

# INFORMATIQUE 2021



Union des Professeurs  
de classes préparatoires  
Scientifiques



# Sommaire

<b>Option informatique</b>	<b>1</b>
<b>X – ENS A (XULCR) (4h) [i213m1e]</b>	
Facteurs dans les mots binaires . . . . .	1
<b>Mines-Ponts (3h) [i21mmoe]</b>	
Le jeu du solitaire . . . . .	13
<b>Centrale-Supélec (4h) [i21cmoe]</b>	
Arbres couvrants et pavages . . . . .	25
<b>CCINP (4h) [i21pmoe]</b>	
Partie I (informatique commune) : étude des partitions non croisées - Partie II (option informatique) : logique et étude du problème Horn-Sat - Partie III (option informatique) : étude des classes sylvestres . . . . .	34
<b>CAPES externe d'informatique - Épreuve 1 [i21c31e]</b>	
Points proches dans le plan, composantes connexes et biconnexes . . . . .	46
<b>CAPES externe d'informatique - Épreuve 2 [i21c32e]</b>	
Réseaux de communication, le Web, développement d'applications . . . . .	59
<b>Informatique commune</b>	<b>74</b>
<b>X – ENS B (XELCR) - MP, PC, PSI (2h) [i213m2e]</b>	
Gestion d'un allocateur dynamique de mémoire . . . . .	74
<b>Mines-Ponts (2h) - MP-PC-PSI [i21mmce]</b>	
Bases de données, mouvement brownien, marche auto-évitante . . . . .	86
<b>Centrale-Supélec - MP, PC, PSI, TSI (3h) [i21cice]</b>	
Lancer de rayons . . . . .	97
<b>CCINP - PSI (3h) [i21psue]</b>	
Autour des montres multisports . . . . .	105
<b>CCINP - TSI (3h) [i21piue]</b>	
Optimisation de rendement d'une entreprise de livraison . . . . .	125
<b>Épreuves de modélisation</b>	<b>145</b>
<b>CCINP - PC (4h) [d21ppue]</b>	
Fuite de matière d'un réservoir rempli de CO <sub>2</sub> gazeux . . . . .	145
<b>CCINP - PSI (4h) [d21psue]</b>	
Régulation d'un système de climatisation à débit d'air variable . . . . .	161

<b>CCINP - TSI (3h) [d21piue]</b>	
Système dish-stirling . . . . .	181
<b>CCINP - TPC (4h) [d21pcue]</b>	
Fuite de matière d'un réservoir rempli de CO <sub>2</sub> gazeux . . . . .	197
<b>Banque PT (4h) [d21dtue]</b>	
Système autofocus d'appareil numérique (informaique en partie 3) . . . . .	213

# En guise d'introduction

*Heureux soient les fêlés car ils laisseront passer la lumière.*

*Apocryphe de Michel Audiard*

Le recueil 2021 comporte trois parties. La première contient les épreuves de l'option informatique de nos classes auxquelles s'ajoutent les épreuves du CAPES d'informatique. La deuxième contient les épreuves d'informatique commune. Enfin, la troisième contient des épreuves de modélisation qui comportent au moins une question d'informatique.

Le bulletin des concours Informatique n'est proposé aux adhérents que sous forme électronique. Dans ce recueil, le nom du fichier contenant chaque sujet figure dans le sommaire et en tête de chaque page. Les fichiers sont disponibles sur le site de l'UPS à l'adresse <https://ups-cpge.fr/ups.php?module=Maths&voir=recherche>. Ce bulletin et ses prédecesseurs sont à votre disposition sur le site de l'UPS à l'adresse <http://prepas.org/ups.php?rubrique=146>.

Laurent Sartre - Septembre 2021

[laurent.sartre@prepas.org](mailto:laurent.sartre@prepas.org)



**ECOLE POLYTECHNIQUE  
ECOLES NORMALES SUPERIEURES**

**CONCOURS D'ADMISSION 2021**

**MARDI 13 AVRIL 2021**

**14h00 - 18h00**

**FILIÈRE MP - Epreuve n° 4**

**INFORMATIQUE A (XULCR)**

*Durée : 4 heures*

*L'utilisation des calculatrices n'est pas autorisée pour  
cette épreuve*

*Cette composition ne concerne qu'une partie des candidats de la filière MP, les autres candidats effectuant simultanément la composition de Physique et Sciences de l'Ingénieur.  
Pour la filière MP, il y a donc deux enveloppes de Sujets pour cette séance.*

## Facteurs dans les mots binaires

Ce sujet traite de mots sur un alphabet  $\{0, 1\}$ . On s'intéresse à leurs facteurs, notamment aux facteurs répétés dans un mot et aux mots ayant le maximum de facteurs distincts. L'algorithmique des facteurs est particulièrement utile dans le domaine de la bio-informatique, où l'analyse de mots très longs, à savoir les génomes, requiert des structures de données efficaces.

**Mots.** Un *mot*  $m$  est une suite finie  $m_0m_1\dots m_{\ell-1}$  de lettres de l'alphabet  $\{0, 1\}$ . Sa *longueur*  $\ell$  est notée  $\ell(m)$  et sa lettre d'indice  $i$  est notée  $m_i$  pour  $0 \leq i \leq \ell(m)-1$ . Le *mot vide*, de longueur 0, est noté  $\varepsilon$ . On note  $c^\ell$  le mot de longueur  $\ell$  formé de  $\ell$  caractères  $c$ .

Pour deux mots  $m = m_0m_1\dots m_{\ell(m)-1}$  et  $m' = m'_0m'_1\dots m'_{\ell(m')-1}$ , on définit leur *concaténation* comme le mot  $mm' = m_0m_1\dots m_{\ell(m)-1}m'_0m'_1\dots m'_{\ell(m')-1}$  de longueur  $\ell(mm') = \ell(m) + \ell(m')$ .

Pour  $0 \leq i \leq j \leq \ell(m)$ , on note  $m[i..j[$  le mot  $m_im_{i+1}\dots m_{j-1}$ , qui est appelé *facteur* de  $m$ . On note que le mot vide est un facteur de n'importe quel mot, obtenu pour  $0 \leq i = j \leq \ell(m)$ . Un *préfixe* (resp. *suffixe*) de  $m$  est un facteur de la forme  $m[0..j[$  (resp.  $m[i..\ell(m)[$ ) et on le note  $m[..j[$  (resp.  $m[i..]$ ). On note

- $F(m) = \{m[i..j[ \mid 0 \leq i \leq j \leq \ell(m)\}$  l'ensemble des facteurs de  $m$  ;
- $P(m) = \{m[..j[ \mid 0 \leq j \leq \ell(m)\}$  l'ensemble des préfixes de  $m$  ;
- $S(m) = \{m[i..[ \mid 0 \leq i \leq \ell(m)\}$  l'ensemble des suffixes de  $m$ .

**Arbres binaires.** Un *arbre binaire* est défini récursivement de la manière suivante :

- soit comme l'arbre vide, noté  $V$ , qui ne contient aucun nœud ;
- soit comme un *nœud*, noté  $N(x, g, d)$  et appelé *racine*, où  $x$  est l'information stockée dans le nœud,  $g$  est un arbre binaire appelé *sous-arbre gauche* et  $d$  est un arbre binaire appelé *sous-arbre droit*.

On note  $n(a)$  le nombre de noeuds et  $h(a)$  la hauteur d'un arbre binaire  $a$ , définis par

$$n(V) = h(V) = 0, \quad n(N(x, g, d)) = 1 + n(g) + n(d) \quad \text{et} \quad h(N(x, g, d)) = 1 + \max(h(g), h(d)).$$

**Langage OCaml.** Ce sujet utilise les listes et les tableaux d'OCaml. Une liste est construite à partir de la liste vide  $[]$  et de la construction  $x :: l$  qui renvoie une nouvelle liste dont la tête est l'élément  $x$  et dont la queue est la liste  $l$ . L'appel de `List.rev l` renvoie une nouvelle liste, formée des éléments de la liste  $l$  en ordre inverse.

On peut créer des tableaux avec les deux fonctions `Array.make` et `Array.of_list`. L'appel de `Array.make n x` crée un tableau de taille `n` dont toutes les cases contiennent la valeur `x`. L'appel de `Array.of_list l` crée un tableau contenant, dans l'ordre, les éléments d'une liste `l`. Les cases d'un tableau sont numérotées à partir de 0. La fonction `Array.length` renvoie la taille d'un tableau. Pour un tableau `tab`, on accède à l'élément d'indice `i` avec `tab.(i)` et on le modifie avec `tab.(i) <- v`.

Dans tout le sujet, on représente un mot par un tableau d'entiers, ne contenant que des entiers 0 ou 1. On se donne donc le type suivant :

```
type mot = int array (* ne contient que des 0/1 *)
```

Dans les fonctions demandées, on ne cherchera jamais à vérifier que les tableaux passés en arguments ne contiennent que des 0 et des 1.

**Complexité.** Par *complexité* (en temps) d'un algorithme  $A$  on entend le nombre d'opérations élémentaires (comparaison, addition, soustraction, multiplication, division, affectation, test, etc.) nécessaires à l'exécution de  $A$  dans le cas le pire. Lorsque la complexité dépend d'un ou plusieurs paramètres  $\kappa_0, \dots, \kappa_{r-1}$ , on dit que  $A$  a une complexité en  $\mathcal{O}(f(\kappa_0, \dots, \kappa_{r-1}))$  s'il existe une constante  $C > 0$  telle que, pour toutes les valeurs de  $\kappa_0, \dots, \kappa_{r-1}$  suffisamment grandes (c'est-à-dire plus grandes qu'un certain seuil), pour toute instance du problème de paramètres  $\kappa_0, \dots, \kappa_{r-1}$ , la complexité est au plus  $C \cdot f(\kappa_0, \dots, \kappa_{r-1})$ .

**Dépendances.** Ce sujet est conçu pour être traité linéairement. La partie I est nécessaire pour la partie II, et ces deux premières parties motivent la partie III, elle-même utilisée dans la question 22 de la partie IV. En revanche, les questions 20 et 21 de la partie IV ainsi que la partie V sont indépendantes du reste du sujet.

## Partie I. Arbres de mots

Dans cette partie, on introduit une structure de données, appelée *arbre de mots*, qui représente un ensemble fini de mots. Un arbre de mots est un arbre binaire dont chaque noeud contient un booléen. Un arbre de mots  $a$  représente un ensemble fini de mots, noté  $\mathbb{M}(a)$ , défini comme suit. Si l'arbre  $a$  est vide, alors  $\mathbb{M}(a)$  est l'ensemble vide. Si l'arbre  $a$  est de la forme  $N(b, a_0, a_1)$ , alors  $\mathbb{M}(a)$  est l'ensemble des mots suivants :

- le mot vide  $\varepsilon$  si le booléen  $b$  est `true` ;
- les mots de la forme  $0m$  pour  $m \in \mathbb{M}(a_0)$  ;
- les mots de la forme  $1m$  pour  $m \in \mathbb{M}(a_1)$ .

Autrement dit,  $\mathbb{M}(a)$  est l'ensemble des mots correspondant aux chemins menant de la racine à un noeud contenant `true`, en utilisant le caractère 0 quand on va à gauche et le caractère 1 quand

on va à droite. Ainsi, l'arbre  $a_1$  de la figure 1 représente l'ensemble de mots  $\mathbb{M}(a_1) = \{\varepsilon, 10, 11\}$ . Dans tous les dessins, on utilise T (resp. F) pour le booléen `true` (resp. `false`).

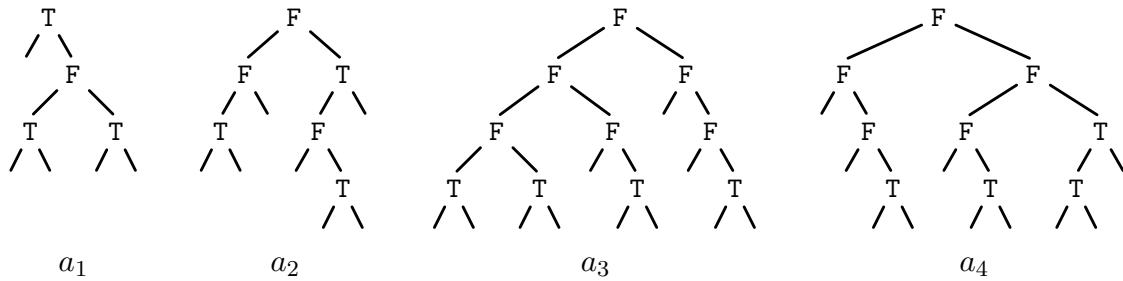


FIGURE 1 – Des arbres de mots.

On note qu'un ensemble de mots peut être représenté par une infinité d'arbres de mots. En effet, il suffit de remplacer un arbre vide par un nœud  $N(\text{false}, V, V)$  pour obtenir un autre arbre de mots représentant le même ensemble de mots. Par exemple, l'arbre vide  $V$  et l'arbre à un nœud  $N(\text{false}, V, V)$  représentent tous les deux l'ensemble vide. On dit qu'un arbre de mots est *réduit* dès lors que tout nœud dont les deux sous-arbres sont vides contient le booléen `true`. Dans toute la suite, on ne manipulera que des arbres de mots réduits.

**Question 1.** Donner l'ensemble de mots représenté par l'arbre  $a_2$  de la figure 1. Dessiner un arbre de mots réduit représentant l'ensemble de mots  $\{\varepsilon, 00, 010, 1, 110, 111\}$ .

**Question 2.** Montrer que, pour tout ensemble fini de mots, il existe un unique arbre de mots réduit qui le représente.

Dans la suite, on note  $\mathbb{A}(M)$  l'arbre de mots réduit de l'ensemble fini de mots  $M$ . On a donc  $\mathbb{M}(\mathbb{A}(M)) = M$  pour tout ensemble fini de mots  $M$  et inversement  $\mathbb{A}(\mathbb{M}(a)) = a$  pour tout arbre de mots réduit  $a$ .

Pour représenter un arbre de mots en OCaml, on se donne le type suivant

```

type am =
| V
| N of bool * am * am
  
```

où `V` représente l'arbre vide et `N` représente un nœud. Par exemple, l'arbre  $a_1$  de la figure 1 est représenté par la valeur suivante :

```
N (true, V, N (false, N (true, V, V), N (true, V, V)))
```

**Question 3.** Écrire une fonction `zeros_puis_1s: int -> am` qui reçoit en argument un entier  $n \geq 0$  et qui renvoie l'arbre de mots réduit représentant tous les mots de la forme  $0^p 1^q$  avec  $p + q = n$ . L'arbre  $a_3$  de la figure 1 donne un exemple d'un tel arbre pour  $n = 3$ .

**Question 4.** Écrire une fonction `k_1s: int -> int -> am` qui reçoit en arguments deux entiers  $k \geq 0$  et  $n \geq 0$  et qui renvoie l'arbre de mots réduit représentant tous les mots de

longueur au plus  $n$  et contenant exactement  $k$  caractères 1. L’arbre  $a_4$  de la figure 1 donne un exemple d’un tel arbre pour  $k = 2$  et  $n = 3$ .

*Indication : Pour s’assurer que l’arbre renvoyé est bien réduit, on pourra se servir de la fonction suivante.*

```
let sN = function
| (false, V, V) -> V
| (b, a0, a1) -> N (b, a0, a1)
```

**Question 5.** Écrire une fonction `compter`: `am` → `int` qui reçoit en argument un arbre de mots  $a$  et renvoie le cardinal de l’ensemble  $\mathbb{M}(a)$ . La complexité en temps doit être linéaire en  $n(a)$ , mais il n’est pas demandé de la justifier.

**Question 6.** Écrire une fonction `chercher`: `am` → `mot` → `bool` qui reçoit en arguments un arbre de mots  $a$  et un mot  $m$  et détermine si  $m$  appartient à l’ensemble  $\mathbb{M}(a)$ . La complexité en temps doit être linéaire en  $\ell(m)$ , mais il n’est pas demandé de la justifier.

**Question 7.** Écrire une fonction `enumerer`: `am` → `mot list` qui reçoit en argument un arbre de mots réduit  $a$  et renvoie l’ensemble de mots  $\mathbb{M}(a)$  sous la forme d’une liste. La complexité en temps doit être linéaire en la taille du résultatat  $\sum_{m \in \mathbb{M}(a)} \ell(m)$ . On justifiera la complexité.

*Indication : On pourra écrire une fonction récursive auxiliaire qui reçoit en arguments*

- une liste `acc` dans laquelle on accumule les mots déjà construits,
- une liste `pref` contenant les lettres rencontrées le long du chemin menant du nœud courant à la racine,
- le sous-arbre  $t$  dont la racine est le nœud courant,

*et qui renvoie la liste `acc` complétée avec tous les mots de  $\mathbb{M}(t)$  auxquels on a ajouté le préfixe donné par la liste `pref`.*

**Question 8.** Écrire une fonction `ajouter`: `am` → `mot` → `am` qui reçoit en arguments un arbre de mots réduit  $a$  et un mot  $m$  et renvoie l’arbre de mots réduit représentant l’ensemble de mots  $\mathbb{M}(a) \cup \{m\}$ . La complexité en temps doit être linéaire en  $\ell(m)$ , mais il n’est pas demandé de la justifier.

Pour la suite, on remarque que les nœuds de l’arbre de mots  $\mathbb{A}(M)$  correspondent à l’ensemble de mots  $\{m[..i] \mid m \in M \text{ et } 0 \leq i \leq \ell(m)\}$ . En effet, les mots donnés par les chemins de la racine aux nœuds de  $\mathbb{A}(M)$  (en utilisant 0 quand on va à gauche et 1 quand on va à droite) sont précisément les préfixes d’au moins un mot de  $M$ .

## Partie II. Arbre des suffixes

Dans cette partie, on considère un mot  $m$  et on définit l’*arbre des suffixes de  $m$* , noté  $\mathbb{AS}(m)$ , comme l’arbre de mots réduit de tous les suffixes de  $m$ , c’est-à-dire  $\mathbb{AS}(m) = \mathbb{A}(S(m))$ .

**Question 9.** Donner l’arbre  $\text{AS}(m)$  pour le mot  $m = 1011$ .

**Question 10.** Quelle est la hauteur de l’arbre  $\text{AS}(m)$ ? En déduire une fonction `retrouver_mot`: `am -> mot` qui retrouve le mot  $m$  à partir de l’arbre  $\text{AS}(m)$  de ses suffixes. La complexité en temps doit être linéaire en  $n(\text{AS}(m))$ , mais il n’est pas demandé de la justifier.

**Question 11.** Montrer que les nœuds de l’arbre  $\text{AS}(m)$  correspondent aux facteurs distincts de  $m$ . En déduire que le nombre de nœuds de l’arbre  $\text{AS}(m)$  est au moins linéaire et au plus quadratique en  $\ell(m)$ . Montrer que ces bornes sont atteintes, c’est-à-dire que pour tout entier  $\ell$ , il existe un mot  $m$  de longueur  $\ell$  tel que l’arbre  $\text{AS}(m)$  a  $\ell + 1$  nœuds (resp. au moins  $C\ell^2$  nœuds où  $C$  est une constante indépendante de  $\ell$  que l’on explicitera).

Des bornes plus précises sur le nombre de facteurs distincts de  $m$  seront étudiées dans la partie IV.

### Partie III. Arbre des facteurs

Dans cette partie, on introduit une structure de données potentiellement plus compacte pour représenter l’arbre des suffixes d’un mot  $m$  fixé, appelée *arbre des facteurs*. Cette structure est basée sur les deux idées suivantes :

- dans l’arbre de suffixes  $\text{AS}(m)$ , un chemin formé de nœuds dont le booléen est `false` et dont l’un des sous-arbres est vide peut être compacté dans le nœud suivant, quitte à se rappeler de la séquence de caractères correspondante (0 quand on va à gauche et 1 quand on va à droite) ;
- une telle séquence de caractères se représente de façon compacte par une paire d’entiers  $(p, q)$  qui désigne le facteur  $m[p..q]$ .

On considère donc un arbre binaire dont chaque nœud contient deux entiers  $p$  et  $q$  tels que  $0 \leq p \leq q \leq \ell(m)$  et un booléen. Un tel arbre  $a$  représente l’ensemble des mots  $\mathbb{M}(a)$  défini comme suit. Si l’arbre  $a$  est vide, alors  $\mathbb{M}(a)$  est l’ensemble vide. Si l’arbre  $a$  est de la forme  $N(p, q, b, a_0, a_1)$ , alors  $\mathbb{M}(a)$  est l’ensemble des mots suivants :

- le mot  $m[p..q[$  si le booléen  $b$  est `true` ;
- les mots de la forme  $m[p..q[0f$  pour  $f \in \mathbb{M}(a_0)$  ;
- les mots de la forme  $m[p..q[1f$  pour  $f \in \mathbb{M}(a_1)$ .

L’arbre  $a$  est un arbre des facteurs du mot  $m$  si l’ensemble  $\mathbb{M}(a)$  est exactement l’ensemble de tous les suffixes de  $m$ . On dit qu’un arbre des facteurs est *réduit* dès lors que tout nœud dont au moins un des sous-arbres est vide contient le booléen `true`. Ainsi, l’arbre de la figure 2 est un arbre des facteurs réduit du mot 10110100. Dans tous les dessins, on représente l’information contenue dans un nœud  $N(p, q, b, a_0, a_1)$  par  $p..q(T)$  si  $b$  est `true` et  $p..q(F)$  si  $b$  est `false`.

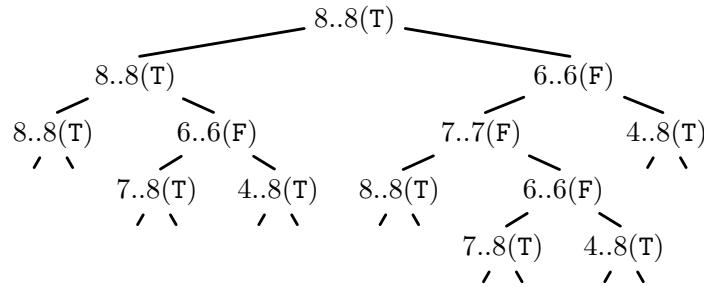


FIGURE 2 – Un arbre des facteurs réduit du mot 10110100.

*Attention : Un arbre des facteurs du mot  $m$  n'est pas un arbre de mots de tous les facteurs de  $m$ , mais une représentation compacte de l'arbre des suffixes de  $m$ . On parle d'arbre des facteurs car il permet de manipuler facilement tous les facteurs de  $m$ , comme on le verra plus loin.*

**Question 12.** Pour chaque arbre  $a$  de la figure 3, donner l'ensemble  $\mathbb{M}(a)$  et indiquer s'il s'agit d'un arbre des facteurs pour le mot  $m = 01001$ , et s'il est réduit.

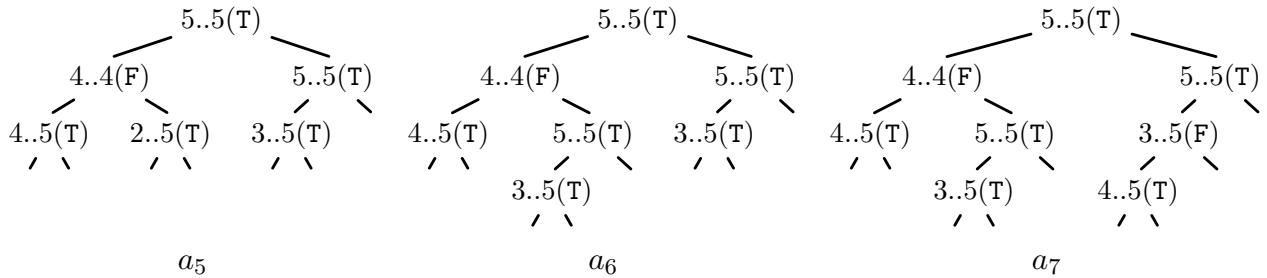


FIGURE 3 – Des arbres.

On peut se convaincre facilement qu'il existe toujours au moins un arbre des facteurs pour tout mot  $m$ . En effet, il suffit de construire l'arbre des suffixes  $\mathbb{AS}(m)$  de  $m$  de la partie II et d'ajouter dans chaque nœud la paire d'entiers  $(0, 0)$ . Pour obtenir un arbre des facteurs réduit, il suffit alors de réduire cet arbre comme expliqué au début de cette partie, en compactant tous les nœuds dont le booléen est `false` et dont un des sous-arbres est vide. On verra plus loin une procédure pour construire directement un arbre des facteurs réduit.

On montre maintenant qu'un arbre des facteurs réduit du mot  $m$  est compact en comparaison de l'arbre des suffixes de la partie II qui pouvait être quadratique comme observé à la question 11.

**Question 13.** Soit  $a$  un arbre des facteurs réduit d'un mot  $m$ . On note que l'arbre  $a$  a  $\ell + 1$  nœuds si  $m$  est  $0^\ell$  ou  $1^\ell$  pour  $\ell \geq 0$ . En supposant que les deux lettres 0 et 1 apparaissent dans le mot  $m$ ,

1. montrer que le nombre de nœuds de  $a$  ayant au moins un sous-arbre vide est au plus  $\ell(m)$ ,
2. en déduire une borne supérieure du nombre de nœuds de  $a$  en fonction de  $\ell(m)$ , et montrer que cette borne est atteinte (c'est-à-dire que pour tout  $\ell > 0$ , il existe un mot de longueur  $\ell$  dont un arbre des facteurs a précisément ce nombre de nœuds).

Pour représenter un arbre des facteurs en OCaml, on se donne le type

```
type af =
| V
| N of int * int * bool * af * af
```

où `V` représente l’arbre vide et `N` représente un nœud. On rappelle que le mot  $m$  est fixé dans cette partie. On pourra supposer qu’il est dans une variable globale `m`.

**Question 14.** Écrire une fonction `nb_facteurs: af -> int` qui reçoit en argument un arbre des facteurs  $a$  du mot  $m$  et renvoie le nombre de facteurs distincts de  $m$ . La complexité en temps doit être linéaire en  $n(a)$ , mais il n’est pas demandé de la justifier.

**Question 15.** Écrire une fonction `plpc: mot -> int -> int -> mot -> int -> int -> int` (pour “plus long préfixe commun”) qui prend en arguments un mot  $m_1$  avec deux indices  $0 \leq p \leq q \leq \ell(m_1)$  et un mot  $m_2$  avec deux indices  $0 \leq i \leq j \leq \ell(m_2)$ , et qui renvoie le plus grand  $k$  tel que  $p + k \leq q$  et  $i + k \leq j$  et  $m_1[p..p+k] = m_2[i..i+k]$ . La complexité en temps doit être linéaire en le résultat, mais il n’est pas demandé de la justifier.

**Question 16.** Écrire une fonction `facteur: af -> mot -> bool` qui reçoit en arguments un arbre des facteurs  $a$  du mot  $m$  et un mot  $f$  et détermine si  $f$  est un facteur de  $m$ . La complexité en temps doit être linéaire en  $\ell(f)$ . On justifiera la complexité.

**Question 17.** Écrire une fonction `facteurs: af -> mot list` qui reçoit en argument un arbre des facteurs réduit  $a$  du mot  $m$  et renvoie l’ensemble des facteurs  $F(m)$  sous la forme d’une liste. Chaque facteur doit apparaître une fois et une seule dans cette liste. La complexité en temps doit être linéaire en la taille du résultat  $\sum_{f \in F(m)} \ell(f)$ . On justifiera la complexité.

On montre maintenant comment construire un arbre des facteurs réduit pour le mot  $m$ . Pour cela, on part de l’arbre vide et on ajoute successivement tous les suffixes du mot  $m$ , dans un ordre arbitraire. Un suffixe  $m[i..]$  est ajouté à l’arbre courant  $a$  de la manière suivante :

- si  $a = V$ , on construit l’arbre  $N(i, \ell(m), \text{true}, V, V)$ ,
- si  $a = N(p, q, b, a_0, a_1)$ , on cherche le plus grand  $k$  tel que  $p + k \leq q$  et  $i + k \leq \ell(m)$  et tel que  $m[p..p+k] = m[i..i+k]$ . On distingue alors deux cas :
  - si  $p + k < q$  alors on construit l’arbre  $N(p, p+k, b', a'_0, a'_1)$  où
    - \* si  $i+k < \ell(m)$ , alors  $b'$  est `false` et  $a'_0$  et  $a'_1$  sont les arbres  $N(p+k+1, q, b, a_0, a_1)$  et  $N(i+k+1, \ell(m), \text{true}, V, V)$ , placés dans le bon ordre ;
    - \* si  $i+k = \ell(m)$ , alors  $b'$  est `true` et  $a'_0$  et  $a'_1$  sont les arbres  $N(p+k+1, q, b, a_0, a_1)$  et  $V$ , placés dans le bon ordre ;
  - si  $p + k = q$ , alors
    - \* si  $i+k < \ell(m)$ , alors on ajoute récursivement le suffixe  $m[i+k+1..]$  dans le bon sous-arbre de  $a$  ;
    - \* si  $i+k = \ell(m)$ , alors on remplace le booléen  $b$  par `true`.

Dans la suite, on admet que cette procédure construit un arbre des facteurs réduit pour le mot  $m$ . On a représenté à la figure 4 les étapes successives de la construction de l'arbre des facteurs pour le mot  $m = 1010$ , en insérant les suffixes dans deux ordres différents. On peut remarquer que la forme de l'arbre des facteurs construit par cette procédure ne dépend pas de l'ordre d'insertion (en revanche, les indices dans les nœuds dépendent de l'ordre d'insertion), mais cette propriété ne sera pas utilisée par la suite.

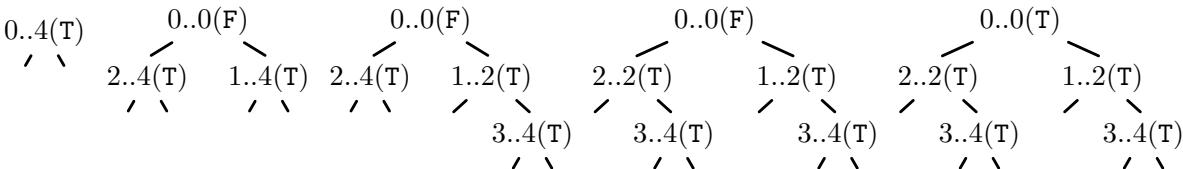
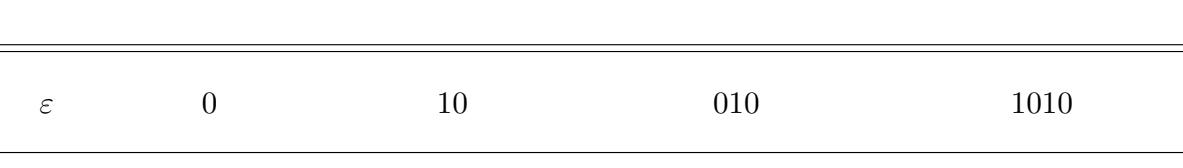
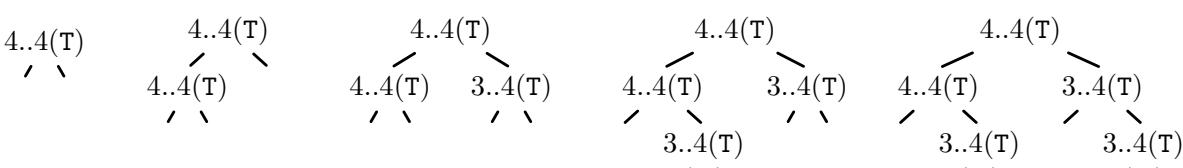
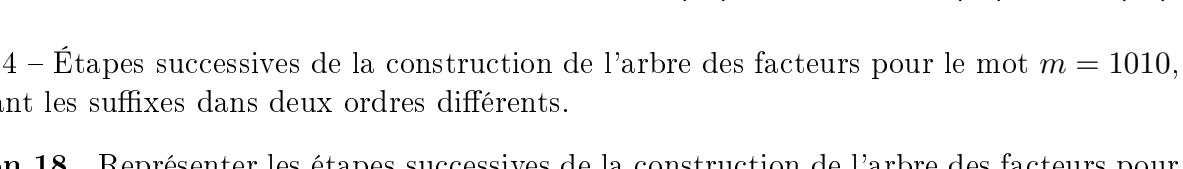
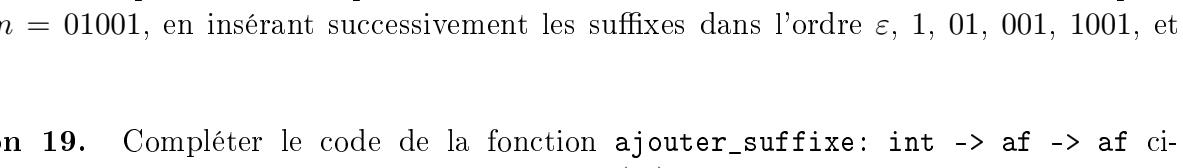
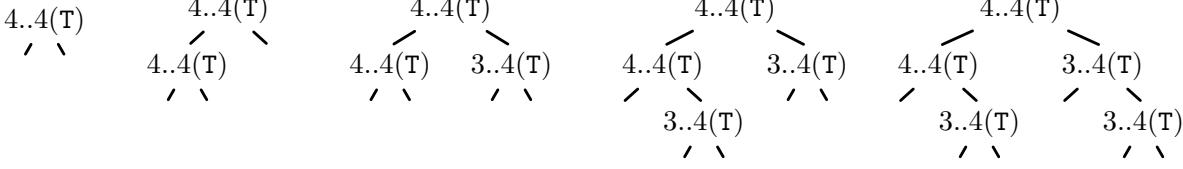
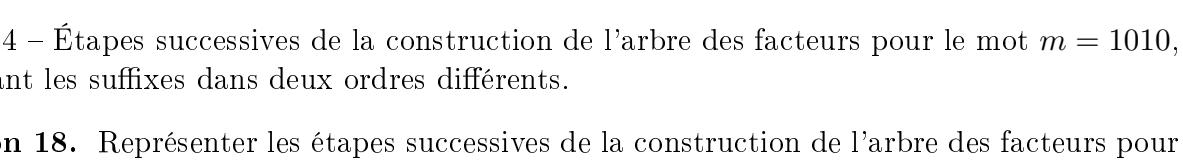
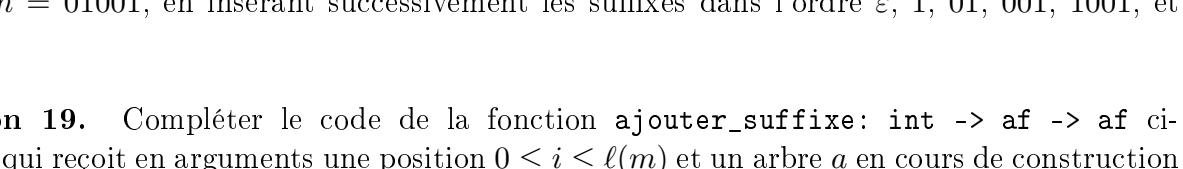
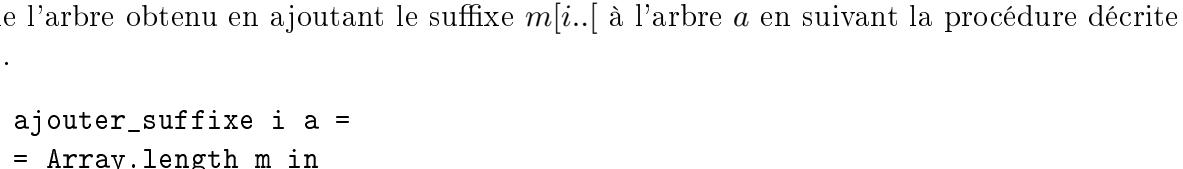
suffixe inséré	1010	010	10	0	$\varepsilon$
arbre obtenu					
suffixe inséré	$\varepsilon$	0	10	010	1010
arbre obtenu					

FIGURE 4 – Étapes successives de la construction de l'arbre des facteurs pour le mot  $m = 1010$ , en insérant les suffixes dans deux ordres différents.

**Question 18.** Représenter les étapes successives de la construction de l'arbre des facteurs pour le mot  $m = 01001$ , en insérant successivement les suffixes dans l'ordre  $\varepsilon, 1, 01, 001, 1001$ , et  $01001$ .

**Question 19.** Compléter le code de la fonction `ajouter_suffixe: int -> af -> af` ci-dessous, qui reçoit en arguments une position  $0 \leq i \leq \ell(m)$  et un arbre  $a$  en cours de construction et renvoie l'arbre obtenu en ajoutant le suffixe  $m[i..]$  à l'arbre  $a$  en suivant la procédure décrite ci-dessus.

```
let rec ajouter_suffixe i a =
  let n = Array.length m in
  match a with
  | V -> N (i, n, true, V, V)
  | N (p, q, b, a0, a1) ->
    let k = plpc m p q m i n in
    if p + k < q then
      (* CAS 1 *)
      ...
    else
      (* CAS 2 *)
      ...
```

On donnera uniquement dans la copie le code correspondant à (\* CAS 1 \*) et (\* CAS 2 \*). La fonction `plpc` est celle décrite à la question 15.

## Partie IV. Application : plus long facteur répété

Dans cette partie, on s'intéresse aux *facteurs répétés* du mot  $m$ , c'est-à-dire aux triplets  $(i, j, k)$  avec  $0 \leq i \leq i + k \leq \ell(m)$  et  $0 \leq j \leq j + k \leq \ell(m)$  tels que  $i \neq j$  et  $m[i..i+k] = m[j..j+k]$ . On cherche la longueur maximale d'un facteur répété dans  $m$ . On commence par un algorithme naïf.

**Question 20.** Écrire une fonction `plfr1: mot -> int` (pour “plus long facteur répété”) qui reçoit en argument un mot  $m$  non vide et renvoie la longueur d'un plus long facteur répété dans  $m$ . La complexité en temps doit être  $O(\ell(m)^3)$ , mais il n'est pas demandé de la justifier.

Pour améliorer la complexité de ce calcul, on utilise maintenant un algorithme de programmation dynamique. Pour  $0 \leq i, j \leq \ell(m)$ , on note  $c(i, j)$  la longueur du plus long suffixe commun de  $m[0..i]$  et  $m[0..j]$ . On rappelle qu'en OCaml, une matrice est un tableau de tableaux. L'appel de `Array.make_matrix p q x` crée une matrice de taille  $p \times q$  dont toutes les cases contiennent la valeur  $x$ . Pour une matrice `mat`, on accède à l'élément  $(i, j)$  avec `mat.(i).(j)` et on le modifie avec `mat.(i).(j) <- v`.

**Question 21.** Écrire une fonction `plfr2: mot -> int` qui reçoit en argument un mot  $m$  non vide et renvoie la longueur d'un plus long facteur répété dans  $m$  en construisant la table des  $c(i, j)$ . La complexité en temps doit être  $O(\ell(m)^2)$ , mais il n'est pas demandé de la justifier.

On suppose maintenant qu'on connaît un arbre des facteurs  $a$  du mot  $m$ . On observe que les plus longs facteurs répétés dans le mot  $m$  correspondent aux noeuds internes de profondeur maximale de l'arbre des facteurs. La profondeur est ici considérée comme étant égale à la longueur du mot représenté par le chemin depuis la racine, c'est-à-dire que chaque noeud  $N(p, q, b, a_0, a_1)$  contribue  $q - p + 1$  à la profondeur.

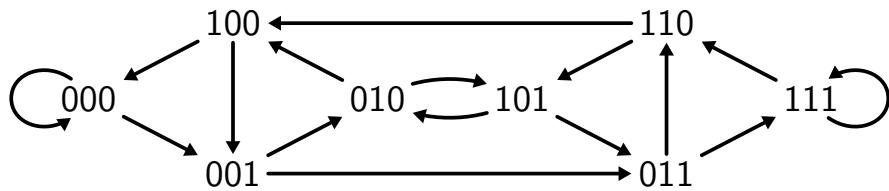
**Question 22.** Écrire une fonction `plfr3: af -> int` qui reçoit en argument un arbre des facteurs réduit  $a$  d'un mot  $m$  non vide et renvoie la longueur d'un plus long facteur répété dans  $m$ . La complexité en temps doit être en  $O(\ell(m))$ . On justifiera la complexité.

## Partie V. Mot contenant un maximum de facteurs distincts

Dans cette partie, on cherche des mots ayant un maximum de facteurs distincts. Pour un mot  $m$  et un entier  $0 \leq k \leq \ell(m)$ , on note  $f_k(m)$  le nombre de facteurs distincts de  $m$  de longueur  $k$ , et  $f(m) = \sum_{k=0}^{\ell(m)} f_k(m)$  le nombre de facteurs distincts de  $m$ .

**Question 23.** Montrer que pour tout mot  $m$  sur l'alphabet  $\{0, 1\}$  et tout entier  $0 \leq k \leq \ell(m)$ , on a

$$f_k(m) \leq \min(2^k, \ell(m) - k + 1).$$

FIGURE 5 – Le graphe  $G_4$ .

On considère maintenant un entier  $k$  fixé. On suppose qu'il existe un mot  $m$  tel que chaque mot de longueur  $k$  sur l'alphabet  $\{0, 1\}$  apparaisse exactement une fois comme facteur de  $m$ . Par exemple, le mot 0001011100 a cette propriété pour  $k = 3$ . On montrera plus tard qu'il tel mot existe toujours.

**Question 24.** Déterminer la longueur d'un tel mot et que ce mot atteint le nombre maximum de facteurs distincts parmi les mots de longueur  $\ell(m)$ .

Il reste donc à montrer qu'il existe un mot  $m$  tel que chaque mot de longueur  $k$  apparaisse exactement une fois comme facteur de  $m$ . Pour cela, on va considérer des cycles eulériens dans un graphe orienté particulier, appelé *graphe de de Bruijn*.

Un graphe orienté  $G$  est dit *fortement connexe* si, pour toute paire de sommets  $v$  et  $w$ , il existe un chemin de  $v$  à  $w$ .

Un graphe orienté  $G$  est dit *eulérien* si, pour tout sommet  $v$ , le degré entrant de  $v$  est égal au degré sortant de  $v$ . Un *circuit eulérien* dans un graphe orienté  $G$  est un circuit passant une et une seule fois par chaque arête de  $G$ .

**Question 25.** Montrer que tout graphe eulérien  $G$  ayant au moins une arête admet un cycle  $C$ , et que le graphe  $G$  privé des arêtes du cycle  $C$  est encore eulérien.

**Question 26.** En déduire qu'un graphe orienté fortement connexe est eulérien si et seulement s'il contient un cycle eulérien.

Finalement, on considère le graphe orienté  $G_k$  dont

- les sommets sont tous les mots à  $k - 1$  lettres sur l'alphabet  $\{0, 1\}$ ,
- les arêtes sont les couples  $(m_1 \dots m_{k-1}, m_2 \dots m_k)$  pour tous les mots  $m_1 \dots m_k$  à  $k$  lettres sur l'alphabet  $\{0, 1\}$ .

Par exemple, le graphe orienté  $G_4$  est représenté à la figure 5.

**Question 27.** Montrer que le graphe  $G_k$  admet un cycle eulérien. En déduire qu'il existe un mot  $m$  tel que chaque mot de longueur  $k$  apparaisse exactement une fois comme facteur de  $m$ .

\* \*  
\*



**A2021 – INFO MP**

**ÉCOLE DES PONTS PARISTECH,  
ISAE-SUPAERO, ENSTA PARIS,  
TÉLÉCOM PARIS, MINES PARIS,  
MINES SAINT-ÉTIENNE, MINES NANCY,  
IMT ATLANTIQUE, ENSAE PARIS,  
CHIMIE PARISTECH - PSL.**

Concours Mines-Télécom,  
Concours Centrale-Supélec (Cycle International).

**CONCOURS 2021****ÉPREUVE D'INFORMATIQUE MP**

**Durée de l'épreuve : 3 heures**

**L'usage de la calculatrice et de tout dispositif électronique est interdit.**

*Cette épreuve concerne uniquement les candidats de la filière MP.*

*Les candidats sont priés de mentionner de façon apparente  
sur la première page de la copie :*

***INFORMATIQUE - MP***

*L'énoncé de cette épreuve comporte 11 pages de texte.*

*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.*

## Préliminaires

L'épreuve est composée d'un problème unique, comportant 30 questions. Après cette section de préliminaires et une section de présentation du *jeu du solitaire*, le problème est divisé en deux sections indépendantes, pages 4 et 8. Dans la première section, nous étudions le jeu du solitaire sur un tablier de forme classique par des méthodes de théorie des graphes. Dans la seconde section, nous étudions le jeu du solitaire sur un tablier en bande unidimensionnelle par des méthodes de théorie des automates. Pour répondre à une question, un candidat pourra réutiliser le résultat d'une question antérieure, même s'il n'est pas parvenu à établir ce résultat.

## Concernant la programmation

Il faudra coder des fonctions à l'aide du langage de programmation OCaml, en reprenant l'entête de fonction fournie par le sujet, sans nécessairement reprendre la déclaration des types. Pour écrire une fonction, on pourra faire appel à d'autres fonctions définies dans les questions précédentes ; on 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. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile de tester si les hypothèses sont bien vérifiées dans le code de la fonction.

Dans tout l'énoncé, un même identificateur écrit dans deux polices de caractère 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  $n$ ).

## Aide à la programmation en OCaml

**Opérations sur les listes :** Sans qu'il ne soit imposé de coder dans un style de programmation les utilisant, on pourra s'appuyer sur les fonctions suivantes :

- append, de type  $'a\ list \rightarrow 'a\ list \rightarrow 'a\ list$ , qui concatène deux listes en une seule liste ;
- filter, de type  $('a \rightarrow \text{bool}) \rightarrow 'a\ list \rightarrow 'a\ list$ , telle que  $\text{filter}\ p\ l$  renvoie la liste des éléments  $x$  de la liste  $\ell$  tels que le prédicat  $p(x)$  vaut true, en respectant l'ordre de la liste ;
- flatten, de type  $'a\ list\ list \rightarrow 'a\ list$ , qui concatène une liste de listes en une seule liste ;
- fold, de type  $('a \rightarrow 'b \rightarrow 'a) \rightarrow 'a \rightarrow 'b\ list \rightarrow 'a$ , telle que  $\text{fold}\ f\ a\ [b_1; \dots; b_n]$  renvoie  $f(\dots(f(f\ a\ b_1)\ b_2)\dots)\ b_n$  ;
- map, de type  $('a \rightarrow 'b) \rightarrow 'a\ list \rightarrow 'b\ list$ , telle que  $\text{map}\ f\ [a_1; \dots; a_n]$  renvoie la liste  $[f\ a_1; \dots; f\ a_n]$ .

**Opérations sur les couples :** On rappelle que

- la fonction `fst`, de type `'a * 'b -> 'a`, renvoie le premier terme d'un couple ;
- la fonction `snd`, de type `'a * 'b -> 'b`, renvoie le second terme d'un couple.

**Opérateurs logiques sur les entiers :** Nous supposons que les entiers naturels, de type `int` en OCaml, sont systématiquement codés à l'aide de 62 bits. Dans un texte mathématique, on pourra signaler la représentation binaire par une barre horizontale. Pour tous entiers  $n$  et  $k$  avec  $0 \leq n < 2^{62}$  et  $0 \leq k < 62$ , nous notons  $[n]_k$  le  $k^{\text{e}}$  bit de poids le plus faible de la représentation binaire de  $n$ , ce qui permet d'écrire  $n = \overline{[n]_{61}[n]_{60} \cdots [n]_0}$ .

Le tableau ci-dessous rappelle le résultat de quelques-uns des opérateurs logiques de OCaml qui agissent bit à bit sur deux entiers naturels  $a$  et  $b$ . Tous ces opérateurs renvoient un élément de type `int`. Dans ce tableau, les symboles  $\wedge$ ,  $\vee$ ,  $\oplus$  et  $\neg$  désignent respectivement le « et », le « ou », le « ou exclusif » entre deux booléens et la négation d'un booléen.

Opérateur	Nom	Résultat exprimé sur 62 bits
<code>a land b</code>	Et logique bit à bit	$([a]_{61} \wedge [b]_{61}) \cdots ([a]_1 \wedge [b]_1)([a]_0 \wedge [b]_0)$
<code>a lor b</code>	Ou logique bit à bit	$([a]_{61} \vee [b]_{61}) \cdots ([a]_1 \vee [b]_1)([a]_0 \vee [b]_0)$
<code>a lxor b</code>	Xor logique bit à bit	$([a]_{61} \oplus [b]_{61}) \cdots ([a]_1 \oplus [b]_1)([a]_0 \oplus [b]_0)$
<code>lnot a</code>	Non logique bit à bit	$(\neg[a]_{61}) \cdots (\neg[a]_1)(\neg[a]_0)$
<code>a lsl b</code>	Décalage vers la gauche	$[a]_{61-b} \cdots [a]_1 \underbrace{[a]_0 0 \cdots 0}_{b \text{ zéros à droite}}$
<code>a lsr b</code>	Décalage vers la droite	$\underbrace{0 \cdots 0}_{b \text{ zéros à gauche}} [a]_{61} \cdots [a]_{b+1} [a]_b$

Par exemple, lorsque  $a = 6$  et  $b = 3$ , les écritures sur 62 bits de  $a$  et  $b$  sont  $a = \overline{0 \cdots 0110}$  et  $b = \overline{0 \cdots 0011}$  et

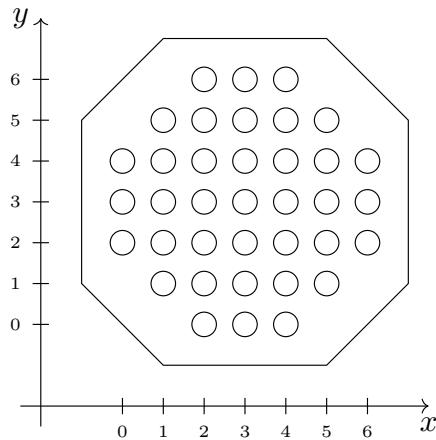
- le calcul de `a land b` vaut l'entier 2, dont l'écriture sur 62 bits est  $\overline{0 \cdots 0010}$  ;
- le calcul de `a lor b` vaut l'entier 7, dont l'écriture sur 62 bits est  $\overline{0 \cdots 0111}$  ;
- le calcul de `a lxor b` vaut l'entier 5, dont l'écriture sur 62 bits est  $\overline{0 \cdots 0101}$  ;
- le calcul de `a lsl b` renvoie 48, dont l'écriture sur 62 bits est  $\overline{0 \cdots 0110000}$  ;
- le calcul de `a lsr b` renvoie 0 puisque les deux bits non nuls de  $a$  sont sortis de la représentation.

L'entier  $2^k$ , avec  $0 \leq k < 62$ , peut commodément se définir en OCaml par `1 lsl k`.

## Le jeu du solitaire

Le *jeu du solitaire* se joue sur un tablier percé d'encoches disposées en quadrillage et remplissant une certaine forme géométrique.

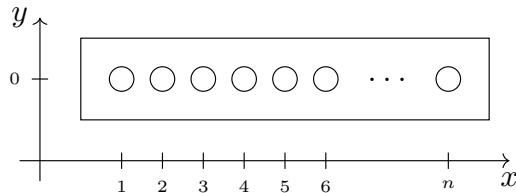
Parmi les tabliers les plus fréquents, le *tablier européen* est formé de 37 encoches organisées en octogone



dont on peut décrire les coordonnées par

$$\{(x, y) \in \mathbb{N}^2; 0 \leq x \leq 6, 0 \leq y \leq 6 \text{ et } |x - 3| + |y - 3| \leq 4\}.$$

D'autres tabliers existent tels que le tablier rectangulaire de dimension  $n \times 1$ , où  $n \geq 1$  est un entier,



et dont on peut décrire les coordonnées par

$$\{(x, y) \in \mathbb{N}^2; 1 \leq x \leq n \text{ et } y = 0\}.$$

Sur un tablier, toute encoche de coordonnées  $(x, y)$  possède au plus quatre encoches *voisines*, à savoir les encoches de coordonnées  $(x + 1, y)$ ,  $(x - 1, y)$ ,  $(x, y + 1)$  et  $(x, y - 1)$  si elles existent.

Nous appelons *motif* tout sous-ensemble d'encoches dans le tablier. Nous disons qu'un motif est *ponctuel* s'il est de cardinal 1.

En début de jeu, des fichets (ou des billes) viennent se loger dans les encoches d'un motif initial. En cours de jeu, un *coup simple* consiste à déplacer l'un des fichets en sautant par-dessus l'une des encoches voisines pour atteindre l'encoche immédiatement après. Il peut être joué à condition que l'encoche voisine en question soit occupée et que le fiche déplacé retombe dans une encoche inoccupée. À l'issue d'un coup simple, le fiche planté dans l'encoche au-dessus de laquelle a eu lieu le saut est retiré. Par exemple,



En cours de jeu, un *coup composé* consiste à jouer zéro, un ou plusieurs coups simples en déplaçant le même fiche. Par exemple,



Une suite de motifs obtenus en jouant une suite de coups s'appelle une *partie*. L'objectif de la joueuse ou du joueur est de transformer un motif en un autre motif par une suite de coups.

## 1 Partie jouée sur le tablier européen en un minimum de coups

Dans toute la section 1 du sujet, une joueuse joue sur le tablier européen. Elle souhaite transformer un motif initial en un motif final par un minimum de coups composés.

- 1 – À titre préliminaire, nommer le type de parcours de graphe le plus approprié pour déterminer un chemin entre deux sommets qui minimise le nombre d'arcs traversés. Quelle politique de mise en attente de sommets caractérise ce parcours ?

### 1.1 Numérotation du tablier

Nous numérotions l'encoche de coordonnées  $(x, y) \in \mathbb{N}^2$  dans le tablier européen par l'entier  $z = 8y + x \in \mathbb{N}$  (voir figure 1, page 5). Nous notons l'ensemble des numéros d'encoches du tablier européen par

$$\mathcal{E} = \{8y + x ; (x, y) \in \mathbb{N}^2 \text{ avec } 0 \leq x, y \leq 6 \text{ et } |x - 3| + |y - 3| \leq 4\} \subseteq \mathbb{N}.$$

**Indication OCaml :** En OCaml, on peut retrouver les coordonnées  $(x, y)$  d'une encoche à partir de son numéro  $z$  en écrivant  $z \bmod 8$  pour obtenir l'abscisse  $x$  et  $z/8$  pour obtenir l'ordonnée  $y$ . On peut calculer la valeur absolue d'un entier avec `abs`.

- 2 – Écrire une fonction `numero_interieur (z:int) : bool`, qui teste si un entier naturel  $z$  est le numéro d'une encoche du tablier européen  $\mathcal{E}$ .

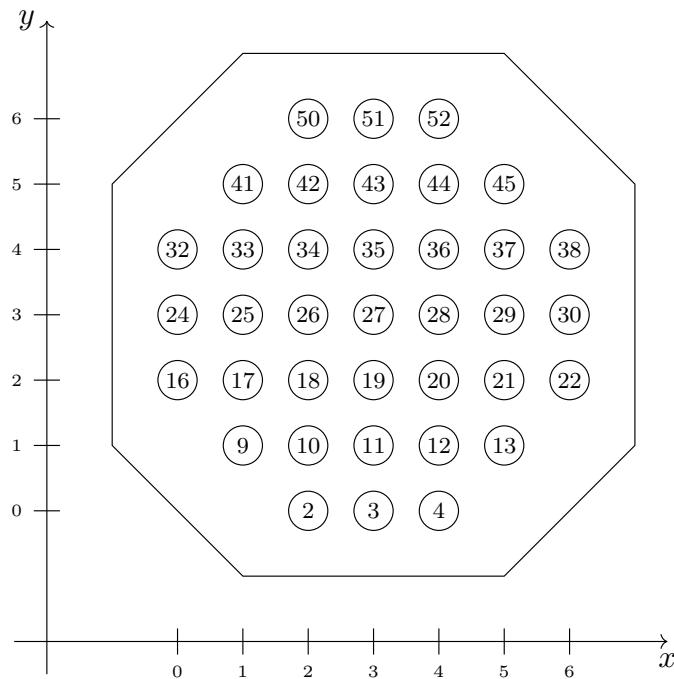


FIGURE 1 – Numérotation des encoches du tablier européen

- 3 – En énumérant les entiers naturels inférieurs à 52 et à l'aide de la fonction `numero_interieur` introduite à la question 2, définir une constante globale `numeros_europeens`, de type `int list`, égale à la liste des numéros d'encoches du tablier européen.
- 4 – En supposant qu'elles existent dans le tablier, donner par une expression mathématique le numéro des quatre encoches voisines de l'encoche de numéro  $z$ .

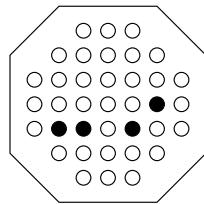
## 1.2 Motifs

Pour représenter un motif  $M \subseteq \mathcal{E}$ , nous utilisons un entier naturel  $m$  dont nous exploitons une partie des bits de l'écriture binaire comme suit : pour tout numéro d'encoche  $z \in \mathcal{E}$ , nous fixons

$$[m]_z = \begin{cases} 1 & \text{si } z \in M \\ 0 & \text{si } z \notin M \text{ ou si } z \notin \mathcal{E} \end{cases}.$$

Lorsqu'un motif est ponctuel, c'est-à-dire qu'il ne contient qu'une seule encoche, sa représentation est simplement une puissance de 2.

Par exemple, le motif à quatre encoches suivant



se représente par le nombre entier

$$m_0 = 2^{29} + 2^{20} + 2^{18} + 2^{17} = 538312704 \in \mathbb{N}.$$

**Indication OCaml :** Pour représenter les motifs et les motifs ponctuels, nous définissons les types `motif` et `ponctuel` par la déclaration suivante :

```
type motif = int;;
type ponctuel = motif;;
```

L'exemple ci-dessus se déclare à l'aide des opérateurs logiques en OCaml par

```
let m0 = (1 lsl 29) lor (1 lsl 20) lor (1 lsl 18) lor (1 lsl 17);;
```

- 5 – Écrire une fonction `numero_vers_ponctuel (z:int) : ponctuel` qui renvoie le motif ponctuel  $\{z\} \subseteq \mathcal{E}$  formé d'une seule encoche de numéro  $z$ .
- 6 – Écrire une fonction `numeros_vers_motif (l:int list) : motif` qui renvoie le motif formé des encoches de numéro appartenant à la liste  $\ell$ .
- 7 – En s'appuyant sur la constante `numeros_europeens` introduite à la question 3, définir une constante globale `motif_europeen`, de type `motif`, égale au motif formé de toutes les encoches du tablier européen.
- 8 – Démontrer que la fonction suivante

```
let est_ponctuel (m:motif) : bool =
  (m>0) && ((m land (m-1)) = 0) ;;
```

renvoie `true` si le motif  $m$  est un motif ponctuel et `false` sinon.

- 9 – Écrire, à l'aide des opérateurs logiques sur les entiers, une fonction `inclus (m:motif) (p:ponctuel) : bool` qui renvoie `true` si le motif ponctuel  $p$  est inclus dans le motif  $m$  ou `false` dans le cas opposé. Donner une preuve mathématique du résultat.

- 10 – Écrire, à l'aide des opérateurs logiques sur les entiers, une fonction `voisin_g (p:ponctuel) : ponctuel` qui calcule la translation par une encoche vers la gauche du motif ponctuel  $p$ . Si le motif ponctuel  $p$  sort du tablier, la valeur de retour est le motif vide 0.

Dans la suite du problème, nous supposerons avoir codé des fonctions similaires `voisin_d`, `voisin_h` et `voisin_b` qui calculent les décalages d'un motif ponctuel respectivement vers la droite, vers le haut et vers le bas. Nous déclarons une constante globale par `let voisins = [voisin_g; voisins_d; voisins_h; voisins_b];;`.

### 1.3 Coups simples et composés

- 11 – Écrire une fonction `coup_simple ((m, p) : motif * ponctuel) : (motif * ponctuel) list` qui prend en entrée un motif quelconque  $m$  et un motif ponctuel  $p$  contenu dans  $m$  et qui construit la liste des couples  $(m', p')$  où  $m'$  est un motif obtenu à partir de  $m$  à la suite du déplacement par un coup simple du fichez initialement placé dans  $p$  et où  $p'$  est le motif ponctuel repérant le même fichez après son déplacement.
- 12 – Écrire une fonction `coup_compose ((m, p) : motif * ponctuel) : (motif * ponctuel) list` qui prend en entrée un motif quelconque  $m$  et un motif ponctuel  $p$  contenu dans  $m$  et qui construit la liste des couples  $(m', p')$  où  $m'$  est un motif obtenu à partir de  $m$  à la suite du déplacement par un coup composé du fichez initialement placé dans  $p$  et où  $p'$  est le motif ponctuel repérant le même fichez après son déplacement.
- 13 – Écrire une fonction `mouvements (m:motif) : motif list` dont la valeur de retour est la liste de tous les motifs que l'on peut obtenir en jouant un coup composé depuis le motif  $m$ , n'importe quel fichez pouvant être déplacé.

### 1.4 Partie de longueur minimale

**Indication OCaml :** Nous utilisons dans cette sous-section une structure de données de dictionnaire, avec modification en place, permettant d'associer des clés de type `motif` et des valeurs de type `int`. Nous notons le type de cette structure `dico`. Pour utiliser ces dictionnaires, nous disposons des fonctions suivantes :

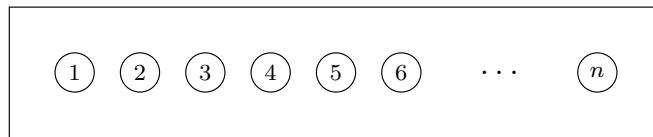
- la fonction `create`, de type `unit -> dico`, qui permet de créer un dictionnaire vide ;
- la fonction `add`, de type `dico -> motif -> int -> unit`, telle que `add d m n` permet d'ajouter au dictionnaire  $d$  une association entre la clé  $m$  et l'entier  $n$  (si une association de clé  $m$  existe déjà, elle est remplacée) ;
- la fonction `mem`, de type `dico -> motif -> bool`, telle que `mem d m` renvoie le booléen `true` si la clé  $m$  est membre du dictionnaire  $d$  ou `false` sinon.
- la fonction `find`, de type `dico -> motif -> int`, telle que `find d m` renvoie la valeur associée à la clé  $m$  si la clé est membre du dictionnaire  $d$  ou une erreur sinon.

- 14 – Écrire une fonction `add_and_mem (d:dico) (s:int) (m:motif) : bool` qui ajoute l'association entre le motif  $m$  et l'entier  $s$  dans le dictionnaire  $d$  si le motif  $m$  est initialement absent du dictionnaire  $d$ . La valeur de retour de la fonction est `true` si le dictionnaire a été modifié et `false` sinon.
- 15 – Écrire une fonction `strate (d:dico) (s:int) (l:motif list) : motif list` qui, pour tout motif  $m$  que l'on peut obtenir après un coup composé à partir d'un motif de la liste  $\ell$ , ajoute l'association entre le motif  $m$  et l'entier  $s$  au dictionnaire  $d$  si le motif  $m$  n'est pas présent dans le dictionnaire  $d$ . La valeur de retour de la fonction est la liste des motifs que la fonction ajoute.

- 16 – Dans cette question, nous supposons qu'il est possible de passer d'un motif  $m_i$  à un motif  $m_f$  par une suite de coups. Écrire une fonction `partie_minimale (mi:motif) (mf:motif)` : int qui calcule le nombre minimal de coups composés à jouer pour passer du motif initial  $m_i$  au motif final  $m_f$ . On pourra utiliser un dictionnaire qui associe des motifs au nombre de coups minimal pour les atteindre depuis le motif  $m_i$ .
- 17 – Peut-on considérer que la réponse donnée à la question 16 observe le parcours nommé à la question 1 ? Introduire un graphe puis argumenter.

## 2 Reconnaissance des motifs résolubles sur le tablier unidimensionnel

Dans toute la section 2 du sujet, une joueuse joue sur des tabliers rectangulaires de dimension  $n \times 1$ , où  $n$  est un entier naturel pouvant varier d'une partie à une autre.



Dans cette section, le but de la joueuse est de faire disparaître tous les fichets sauf un par une suite de coups, autrement dit de jouer une partie qui amène le motif initial à un motif ponctuel. Lorsque ceci est possible, nous disons que le motif initial est *résoluble* et que la partie est *gagnante*.

Nous introduisons l'alphabet binaire  $\Sigma = \{0, 1\}$  et notons  $\Sigma^*$  l'ensemble des mots sur  $\Sigma$ . Pour tout mot  $w = \sigma_1 \dots \sigma_n \in \Sigma^*$ , nous notons  $[w]_k$  le  $k^{\text{e}}$  symbole  $\sigma_k$  de  $w$ .

Le *mot d'un motif*  $M$  du tablier rectangulaire de dimension  $n \times 1$  est le mot  $w$  formé de  $n$  symboles de l'alphabet  $\Sigma$  tel que, pour  $k$  variant entre 1 et  $n$ , le symbole  $[w]_k$  vaut 1 si la  $k^{\text{e}}$  encoche à partir de la gauche appartient au motif  $M$  et vaut 0 sinon. Par exemple, sur ce tablier, le motif  $\bullet\circ\bullet\bullet\circ$  est décrit par le mot  $10110 \in \Sigma^*$ .

Dans toute cette section, nous ne considérons que des coups simples. Une partie est par conséquent une suite finie  $W$  de mots  $w_0, w_1, \dots, w_m$  de même longueur telle que pour tout indice  $i$  compris entre 0 et  $m - 1$ , le mot  $w_{i+1}$  est obtenu à partir du mot  $w_i$  en remplaçant dans le mot  $w_i$  un facteur 110 par les symboles 001 ou bien en remplaçant un facteur 011 par les symboles 100. Dans le premier cas, nous parlons de coup vers la droite ; dans le second cas, nous parlons de coup vers la gauche. Une partie est gagnante si, de plus, le dernier mot  $w_m$  de la partie ne comprend le symbole 1 qu'une seule fois.

L'objectif de cette section est d'étudier le langage  $\mathcal{R} \subseteq \Sigma^*$  des mots de longueur quelconque de motifs résolubles.

## 2.1 Mots de motifs ponctuels

Nous notons  $\mathcal{P} \subseteq \Sigma^*$  le langage des mots de longueur quelconque de motifs ponctuels.

- 18 – Donner, sans justification, une expression rationnelle qui décrit le langage  $\mathcal{P}$ .
- 19 – Dessiner, sans justification, un automate fini qui reconnaît le langage  $\mathcal{P}$ .
- 20 – Le langage  $\mathcal{P}$  est-il un langage local ?

## 2.2 Bascules de l'état d'une encoche

Soient  $W = (w_0, w_1, \dots, w_m) \in (\Sigma^n)^{m+1}$  une partie en  $m$  coups simples joués sur le tablier de dimension  $n \times 1$  et  $k$  un entier compris entre 1 et  $n$ . Dans cette sous-section, nous souhaitons démontrer qu'il existe une constante  $\alpha$  indépendante des entiers  $n$ ,  $m$  ou  $k$  telle la suite des  $(m + 1)$  symboles  $[w_0]_k, [w_1]_k, \dots, [w_m]_k$  ne change jamais de valeur à plus de  $\alpha$  reprises.

Nous notons  $\phi = \frac{\sqrt{5}-1}{2}$  l'une des racines du polynôme  $X^2 + X - 1$ . On pourra utiliser sans preuve l'inégalité  $\sum_{j=1}^{+\infty} \phi^j \leq 2$ . Pour tout mot  $w \in \Sigma^n$  de longueur  $n$ , nous introduisons le variant  $V_k(w)$  par la somme

$$V_k(w) = \sum_{j=1}^n \phi^{|j-k|} [w]_j \in \mathbb{R}.$$

- 21 – Montrer qu'il existe un réel  $S$ , indépendant des entiers  $n$  et  $k$ , tel que, pour tout mot  $w \in \Sigma^n$ , on ait l'inégalité  $V_k(w) \leq S$ .
- 22 – Montrer que la suite des  $(m + 1)$  variants  $V_k(w_0), V_k(w_1), \dots, V_k(w_m)$  est une suite de réels décroissante.
- 23 – Montrer que si, pour un indice  $i$  compris entre 0 et  $m - 1$ , le symbole  $[w_i]_k$  vaut 1 et le symbole  $[w_{i+1}]_k$  vaut 0, alors on a  $V_k(w_{i+1}) \leq V_k(w_i) - 1$ .
- 24 – En déduire qu'il existe une constante  $\alpha$ , indépendante des entiers  $n$ ,  $m$  et  $k$ , telle que la suite  $[w_0]_k, [w_1]_k, \dots, [w_m]_k$  ne change jamais de valeur à plus de  $\alpha$  reprises.

## 2.3 Trace d'une partie

À partir d'une partie  $W = (w_0, w_1, \dots, w_m) \in (\Sigma^n)^{m+1}$  en  $m$  coups simples joués sur le tablier de dimension  $n \times 1$ , nous construisons la *trace*  $T$  de la partie  $W$  en suivant la procédure suivante. Pour plus de clarté, un exemple est présenté en colonne de droite avec la partie en deux coups  $(w_0, w_1, w_2)$  où  $w_0 = 110101$ ,  $w_1 = 001101$  et  $w_2 = 010001$ .

<p><b>ÉTAPE 1 :</b> Nous organisons les mots de <math>W</math> sous forme d'un tableau de <math>n</math> colonnes et <math>m + 1</math> lignes en positionnant le mot <math>w_i</math> dans la ligne <math>i</math>. Les lignes sont numérotées de bas en haut.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td><math>w_2</math></td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td><math>w_1</math></td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr> <td><math>w_0</math></td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table>	$w_2$	0	1	0	0	0	1	$w_1$	0	0	1	1	0	1	$w_0$	1	1	0	1	0	1
$w_2$	0	1	0	0	0	1																
$w_1$	0	0	1	1	0	1																
$w_0$	1	1	0	1	0	1																
<p><b>ÉTAPE 2 :</b> Pour tout indice de ligne <math>i</math> compris entre 0 et <math>m - 1</math>, pour tout indice de colonne <math>j</math> compris entre 1 et <math>n</math>, quand les symboles <math>[w_i]_j</math> et <math>[w_{i+1}]_j</math> sont égaux, nous effaçons le symbole <math>[w_{i+1}]_j</math> inscrit en position <math>(i + 1, j)</math>, de sorte qu'il ne reste que le mot <math>w_0</math> dans la ligne 0 et les 3 symboles qui ont été modifiés dans les autres lignes.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td><td>1</td><td>0</td><td>0</td><td></td><td></td><td></td></tr> <tr> <td>0</td><td>0</td><td>1</td><td></td><td></td><td></td><td></td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> </table>		1	0	0				0	0	1					1	1	0	1	0	1	
	1	0	0																			
0	0	1																				
1	1	0	1	0	1																	
<p><b>ÉTAPE 3 :</b> Pour tout indice <math>i</math> compris entre 0 et <math>m - 1</math>, lorsqu'un coup vers la gauche a été joué entre le mot <math>w_i</math> et le mot <math>w_{i+1}</math>, nous ajoutons la flèche <math>\leftarrow</math> en indice des symboles de la ligne <math>i + 1</math>; lorsqu'un coup vers la droite a été joué, nous ajoutons la flèche <math>\rightarrow</math> en indice.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td><td><math>1_{\leftarrow}</math></td><td><math>0_{\leftarrow}</math></td><td><math>0_{\leftarrow}</math></td><td></td><td></td><td></td></tr> <tr> <td><math>0_{\rightarrow}</math></td><td><math>0_{\rightarrow}</math></td><td><math>1_{\rightarrow}</math></td><td></td><td></td><td></td><td></td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> </table>		$1_{\leftarrow}$	$0_{\leftarrow}$	$0_{\leftarrow}$				$0_{\rightarrow}$	$0_{\rightarrow}$	$1_{\rightarrow}$					1	1	0	1	0	1	
	$1_{\leftarrow}$	$0_{\leftarrow}$	$0_{\leftarrow}$																			
$0_{\rightarrow}$	$0_{\rightarrow}$	$1_{\rightarrow}$																				
1	1	0	1	0	1																	
<p><b>ÉTAPE 4 :</b> Nous numérotons dans chaque ligne différente de la ligne 0 les trois symboles restants par les marques I, II et III en exposant, en allant de gauche à droite.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td><td><math>1^I_{\leftarrow}</math></td><td><math>0^{II}_{\leftarrow}</math></td><td><math>0^{III}_{\leftarrow}</math></td><td></td><td></td><td></td></tr> <tr> <td><math>0^I_{\rightarrow}</math></td><td><math>0^{II}_{\rightarrow}</math></td><td><math>1^{III}_{\rightarrow}</math></td><td></td><td></td><td></td><td></td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> </table>		$1^I_{\leftarrow}$	$0^{II}_{\leftarrow}$	$0^{III}_{\leftarrow}$				$0^I_{\rightarrow}$	$0^{II}_{\rightarrow}$	$1^{III}_{\rightarrow}$					1	1	0	1	0	1	
	$1^I_{\leftarrow}$	$0^{II}_{\leftarrow}$	$0^{III}_{\leftarrow}$																			
$0^I_{\rightarrow}$	$0^{II}_{\rightarrow}$	$1^{III}_{\rightarrow}$																				
1	1	0	1	0	1																	
<p><b>ÉTAPE 5 :</b> Nous abaissons l'ensemble des symboles, colonne par colonne, pour les regrouper dans les alvéoles les plus bas.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td><td><math>1^I_{\leftarrow}</math></td><td><math>0^{II}_{\leftarrow}</math></td><td></td><td></td><td></td><td></td></tr> <tr> <td><math>0^I_{\rightarrow}</math></td><td><math>0^{II}_{\rightarrow}</math></td><td><math>1^{III}_{\rightarrow}</math></td><td><math>0^{III}_{\leftarrow}</math></td><td></td><td></td><td></td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> </table>		$1^I_{\leftarrow}$	$0^{II}_{\leftarrow}$					$0^I_{\rightarrow}$	$0^{II}_{\rightarrow}$	$1^{III}_{\rightarrow}$	$0^{III}_{\leftarrow}$				1	1	0	1	0	1	
	$1^I_{\leftarrow}$	$0^{II}_{\leftarrow}$																				
$0^I_{\rightarrow}$	$0^{II}_{\rightarrow}$	$1^{III}_{\rightarrow}$	$0^{III}_{\leftarrow}$																			
1	1	0	1	0	1																	
<p><b>ÉTAPE 6 :</b> Si <math>x</math> est un symbole marqué par l'exposant I ou II et est initialement issu de la ligne <math>i</math> et si <math>y</math> est son successeur dans le mot <math>w_i</math>, nous adjoignons à <math>x</math> le numéro de ligne dans lequel <math>y</math> se trouve après abaissement.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td><td><math>(1^I_{\leftarrow}, 2)</math></td><td><math>(0^{II}_{\leftarrow}, 1)</math></td><td></td><td></td><td></td><td></td></tr> <tr> <td><math>(0^I_{\rightarrow}, 1)</math></td><td><math>(0^{II}_{\rightarrow}, 1)</math></td><td><math>1^{III}_{\rightarrow}</math></td><td><math>0^{III}_{\leftarrow}</math></td><td></td><td></td><td></td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> </table>		$(1^I_{\leftarrow}, 2)$	$(0^{II}_{\leftarrow}, 1)$					$(0^I_{\rightarrow}, 1)$	$(0^{II}_{\rightarrow}, 1)$	$1^{III}_{\rightarrow}$	$0^{III}_{\leftarrow}$				1	1	0	1	0	1	
	$(1^I_{\leftarrow}, 2)$	$(0^{II}_{\leftarrow}, 1)$																				
$(0^I_{\rightarrow}, 1)$	$(0^{II}_{\rightarrow}, 1)$	$1^{III}_{\rightarrow}$	$0^{III}_{\leftarrow}$																			
1	1	0	1	0	1																	

La trace de la partie  $W$  est la suite des colonnes du tableau obtenue par la procédure ainsi décrite.

- 25 – Soit  $T$  la trace d'une partie. Décrire une construction qui produit une partie de trace  $T$  à partir de la trace  $T$ . (Il est suggéré de raisonner par récurrence.)
- 26 – Montrer que l'on peut majorer le nombre de lignes non vides dans la trace d'une partie par une constante indépendante de la dimension  $n$  du tablier ou de la longueur de la partie  $m$ . En déduire qu'une colonne de la trace d'une partie ne peut prendre qu'un nombre fini de valeurs distinctes.

Nous appelons  $\Delta$  l'ensemble, fini, des valeurs colonnes que peuvent prendre les colonnes d'une trace de partie. Nous appelons  $\mathcal{T}$  le langage des mots sur  $\Delta$  qui représentent la trace d'une partie.

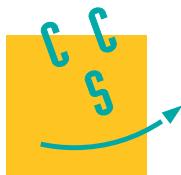
- 27 – Donner une caractérisation de l'ensemble des facteurs de longueur 2 de mots du langage  $\mathcal{T} \subseteq \Delta^*$ .
- 28 – Montrer qu'il existe un automate local qui reconnaît exactement le langage  $\mathcal{T} \subseteq \Delta^*$ .
- 29 – Montrer qu'il existe un automate fini qui reconnaît le langage  $\mathcal{R} \subseteq \Sigma^*$  des mots de longueur quelconque de motifs résolubles.

Dans la question qui suit, le terme *complexité* désigne un ordre de grandeur asymptotique de la complexité en temps et dans le pire des cas.

- 30 – On s'intéresse au problème de déterminer si un motif sur le tablier unidimensionnel est résoluble ou non. Montrer que l'on peut déduire de l'automate construit à la question 29 un algorithme de complexité optimale pour résoudre ce problème.

Les résultats démontrés dans la section 2 ont été établis par B. Ravikumar pour un tablier rectangulaire de dimension  $n \times n_0$  où  $n_0$  est un entier fixé et  $n$  est un entier pouvant varier.

FIN DE L'ÉPREUVE



# Option informatique

MP

2021

CONCOURS CENTRALE-SUPÉLEC

4 heures

Calculatrice autorisée

## *Arbres couvrants et pavages*

Le seul langage de programmation autorisé dans cette épreuve est Caml.

Toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max` ou `incr` ainsi que les opérateurs comme `/` ou `mod`) peuvent être librement utilisés.

On utilisera également le générateur pseudo-aléatoire `int` du module `Random`. Quand `n` est un entier supérieur ou égal à 1, l'appel `Random.int n` renvoie un entier dans l'intervalle  $[0, n]$  (compris entre 0 inclus et `n` exclu) en simulant une loi uniforme. Les appels successifs à cette fonction fournissent des résultats indépendants.

Les candidats ne devront faire appel à aucun autre module.

En Caml, les tableaux d'objets d'un certain type '`a`' sont représentés par le type '`a array`'. L'expression `tab.(i)` permet d'accéder à l'élément situé en `i`-ème position du tableau `tab`. Dans le texte, en dehors du code Caml, cet élément sera noté  $Tab[i]$ . Plus généralement, les objets mathématiques seront notés  $G$ ,  $s$ ,  $Adj\dots$  et représentés en Caml par `g`, `s`, `adj\dots` sans que cela soit systématiquement explicité.

Dans ce problème, un graphe  $G$  est un couple  $(S, A)$  où :

- $S$  est un ensemble fini dont les éléments sont les *sommets* de  $G$  ;
- $A = (a_0, a_1, \dots, a_{m-1})$  est la suite des *arêtes* de  $G$ , une arête étant une partie  $a = \{s, t\}$  de  $S$  de cardinal 2. Les sommets  $s$  et  $t$  sont appelés les *extrémités* de l'arête  $a$  et on dira que  $a$  *relie*  $s$  et  $t$ . Si  $s$  et  $t$  sont reliés par une arête, on dit qu'ils sont *voisins* ou *adjacents*.

Ainsi, les graphes sont non orientés et il n'y a pas d'arête reliant un sommet à lui-même. Par contre, à partir de la partie V, il est possible que plusieurs arêtes aient les mêmes extrémités.

Par convention, nous noterons  $n$  (respectivement  $m$ ) le nombre de sommets (respectivement d'arêtes) du graphe et nous supposerons que  $S = \{0, 1, \dots, n - 1\} = S_n$ .

Un graphe sera représenté par le type

```
type graphe = int list array
```

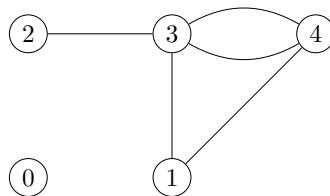
Si  $G$  est un graphe représenté par `g`, alors le nombre de sommets  $n$  du graphe est donné par la longueur du tableau `g`. De plus, si  $s \in S$ , alors `g.(s)` est une liste contenant les indices des sommets voisins de  $s$ , dans un ordre quelconque, chaque sommet apparaissant autant de fois qu'il existe d'arêtes vers  $s$ .

Les complexités temporelles demandées sont des complexités *dans le pire des cas* et seront exprimées sous la forme  $\mathcal{O}(f(n, m))$ , où  $f$  est une fonction la plus simple possible.

Nous utiliserons la représentation graphique habituelle des graphes. Le graphe  $G$ , défini par l'instruction

```
let g = [| [] ; [3; 4] ; [3] ; [1; 2; 4; 4] ; [1; 3; 3] |];;
```

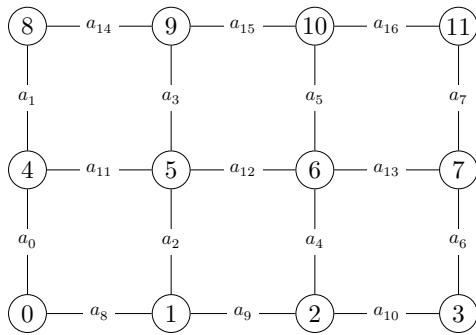
peut être représenté par le graphique de la figure 1.



**Figure 1** Exemple de graphe

Pour  $p, q$  entiers naturels non nuls, nous noterons  $G_{p,q}$  le graphe, appelé *graphe quadrillage*, dont les sommets sont placés sur  $p$  colonnes et  $q$  lignes, chaque sommet étant relié à ses voisins directs. On convient de numérotter les sommets de gauche à droite et de bas en haut.

Par ailleurs, l'ordre des arêtes ayant une importance en partie V, on convient de numérotter les  $p(q-1)+q(p-1)$  arêtes en commençant par les arêtes verticales, de bas en haut et de gauche à droite, puis les arêtes horizontales, de gauche à droite et de bas en haut, comme le montre la figure 2, où  $p = 4$  et  $q = 3$ . Le *rang* d'une arête  $a$  de  $G_{p,q}$  est l'indice  $i$  tel que  $a = a_i$ .

Figure 2 Le graphe  $G_{4,3}$ 

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

- Si  $s \in S_n$ , le *tableau d'adjacence* de  $s$  est le tableau, dans un ordre quelconque, des voisins de  $s$ , chaque sommet apparaissant autant de fois qu'il existe d'arêtes vers  $s$ . On note  $Adj$  le tableau de taille  $n$  tel que, pour tout  $s \in \{0, 1, \dots, n - 1\}$ ,  $Adj[s]$  est le tableau d'adjacence du sommet  $s$ .  $Adj$  est donc représenté par une variable `adj` de type `int array array`.
- Un *chemin* dans  $G$  est une suite  $c = (s_0, s_1, \dots, s_{j-1}, s_j, \dots, s_{k-1}, s_k)$  où pour tout  $j$  compris entre 1 et  $k$ ,  $s_{j-1}$  et  $s_j$  sont des sommets voisins. On dira que  $c$  est un chemin de  $s_0$  à  $s_k$  de longueur  $k$ . Par convention, pour  $s$  sommet de  $G$ , il existe un chemin de longueur nulle de  $s$  à  $s$ .
- La *composante connexe* d'un sommet  $s$  de  $G$ , notée  $C_s$ , est l'ensemble des sommets  $t$  de  $G$  tels qu'il existe un chemin de  $s$  à  $t$ .
- On dit que  $G$  est *connexe* si pour tous sommets  $s$  et  $t$  de  $G$ , il existe un chemin de  $s$  à  $t$ .
- Un *cycle* dans  $G$  est un chemin de longueur  $k \geq 2$  d'un sommet à lui-même et dont les arêtes sont deux à deux distinctes. On dit que  $G$  est *acyclique* s'il ne contient aucun cycle.
- Un *arbre* est un graphe connexe acyclique.

## I Quelques fonctions auxiliaires

**Q 1.** Écrire une fonction

```
nombre_aretes : graphe -> int
```

qui, appliquée à `g` représentant un graphe  $G = (S, A)$ , renvoie le cardinal de  $A$ .

**Q 2.** Écrire une commande Caml permettant de créer le tableau des tableaux d'adjacence  $Adj$  associé au graphe  $G_{3,2}$ .

On rappelle que la fonction `Array.of_list : 'a list -> 'a array` prend en argument une liste  $[x_0; x_1; \dots; x_{k-1}]$  d'éléments de type `'a` et renvoie le tableau  $[[x_0; x_1; \dots; x_{k-1}]]$ .

**Q 3.** Écrire une fonction

```
adjacence : graphe -> int array array
```

qui, appliquée à `g` représentant un graphe  $G$ , renvoie `adj` représentant le tableau des tableaux d'adjacence  $Adj$  associé à  $G$ . La fonction `adjacence` devra être de complexité  $\mathcal{O}(m + n)$  et on demande de justifier cette complexité.

**Q 4.** Écrire une fonction

```
rang : int * int -> int * int -> int
```

telle que `rang (p, q) (s, t)` renvoie le rang de l'arête  $\{s, t\}$  dans le graphe quadrillage  $G_{p,q}$ . Par exemple, `rang (4, 3) (6, 7)` renvoie 13. On suppose que  $s$  et  $t$  sont adjacents et  $s < t$  et on ne demande pas de le vérifier.

**Q 5.** Écrire de même sa fonction réciproque

```
sommets : int * int -> int -> int * int
```

telle que `sommets (p, q) i` renvoie le couple  $(s, t)$  tel que  $a_i = \{s, t\}$  et  $s < t$  dans le graphe  $G_{p,q}$ .

**Q 6.** Écrire une fonction

```
quadrillage : int -> int -> graphe
```

telle que l'instruction `quadrillage p q` renvoie le graphe représentant  $G_{p,q}$ .

## II Caractérisation des arbres

Soit  $G = (S_n, A)$  un graphe à  $n$  sommets et  $m$  arêtes, représenté par  $\mathbf{g}$ . Les arêtes de  $G$  sont donc numérotées  $A = (a_0, a_1, \dots, a_{m-1})$ .

### II.A – Propriétés sur les arbres

**Q 7.** Montrer que les composantes connexes de  $G$  forment une partition de  $S_n$ , c'est-à-dire que :

- $\forall s \in S_n, C_s \neq \emptyset$  ;
- $S_n = \bigcup_{s \in S_n} C_s$  ;
- pour tous sommets  $s$  et  $t$ , soit  $C_s = C_t$ , soit  $C_s \cap C_t = \emptyset$ .

**Q 8.** Montrer que si  $s$  et  $t$  sont deux sommets de  $G$  tels que  $t \in C_s$ , il existe un plus court chemin de  $s$  à  $t$  et que les sommets d'un plus court chemin sont deux à deux distincts.

Pour  $k \in \{0, 1, \dots, m\}$ ,  $G_k$  désigne le graphe  $(S_n, (a_0, a_1, \dots, a_{k-1}))$  obtenu en ne conservant que les  $k$  premières arêtes de  $G$ .

**Q 9.** On suppose que  $G$  est un arbre. Montrer que pour tout  $k \in \{0, 1, \dots, m-1\}$ , les extrémités de  $a_k$  appartiennent à deux composantes connexes différentes de  $G_k$ . En déduire que  $m = n - 1$ .

**Q 10.** Montrer que les trois propriétés suivantes sont équivalentes :

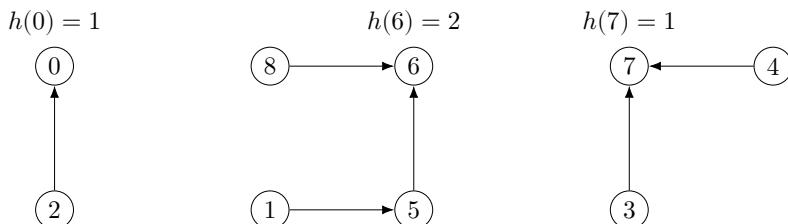
- (i)  $G$  est un arbre ;
- (ii)  $G$  est connexe et  $m = n - 1$  ;
- (iii)  $G$  est acyclique et  $m = n - 1$ .

### II.B – Manipulation de partitions

Nous souhaitons écrire une fonction qui teste si un graphe est un arbre. Nous allons pour cela utiliser une structure de données permettant de manipuler les partitions de l'ensemble  $S_n$  : si  $\mathcal{P} = \{X_1, X_2, \dots, X_k\}$  est une partition de  $S_n$ , on choisit dans chaque partie  $X_i$ , un élément particulier  $r_i$ , appelé *représentant* de  $X_i$ . Notre structure de données doit nous permettre :

- de calculer, pour  $s \in S_n$ , le représentant de la partie  $X_i$  contenant  $s$  ; cet élément sera également appelé *représentant de s* ;
- pour deux entiers  $s$  et  $t$  représentant des parties distinctes  $X_i$  et  $X_j$ , de transformer  $\mathcal{P}$  en réunissant  $X_i$  et  $X_j$ ,  $s$  ou  $t$  devenant le représentant de la partie  $X_i \cup X_j$ .

Nous représenterons une partition  $\mathcal{P} = \{X_1, \dots, X_k\}$  de  $S_n$  par une *forêt* : chaque  $X_i$  est représenté par un arbre dont les noeuds sont étiquetés par les éléments de  $X_i$  et de racine le représentant  $r_i$  de  $X_i$ , les arcs étant orientés vers la racine. Nous noterons  $h(r_i)$  la hauteur de l'arbre  $X_i$ , c'est-à-dire la longueur de sa plus longue branche. Ainsi,  $\mathcal{P}_9 = \{\{0, 2\}, \{1, 5, 6, 8\}, \{3, 4, 7\}\}$  est une partition de  $S_9$  et peut par exemple être représentée par la forêt de la figure 3.



**Figure 3** Une représentation de  $\mathcal{P}_9 = \{\{0, 2\}, \{1, 5, 6, 8\}, \{3, 4, 7\}\}$

Le calcul du représentant d'un entier  $s \in S_n$  se fait donc en remontant jusqu'à la racine de l'arbre (ce qui justifie l'orientation des arcs). Dans l'exemple, les représentants de 2, 1 et 7 sont respectivement 0, 6 et 7.

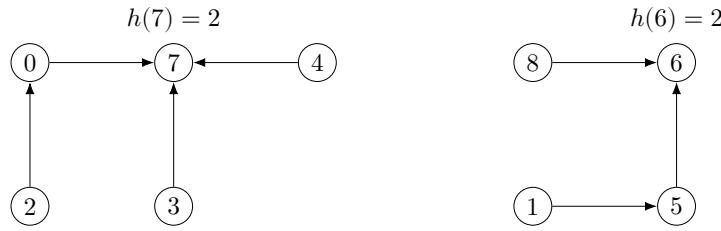
Une partition  $\mathcal{P}$  de  $S_n = \{0, 1, \dots, n-1\}$  sera représentée par un tableau d'entiers  $P$  de taille  $n$  de sorte que pour tout  $s \in \{0, 1, \dots, n-1\}$ ,  $P[s]$  est le père de  $s$  si  $s$  n'est pas un représentant, et est égal à  $-1 - h(s)$  si  $s$  est un représentant. La partition  $\mathcal{P}_9$  peut ainsi être représentée par le tableau

`[| -2; 5; 0; 7; 6; -3; -2; 6 |]`

La réunion de deux parties  $X_i$  et  $X_j$  de représentants  $s$  et  $t$  distincts se fait selon la méthode suivante :

- si  $h(s) > h(t)$ ,  $s$  est choisi pour représentant de la partie  $X_i \cup X_j$  et devient le père de  $t$  ;
- si  $h(s) \leq h(t)$ ,  $t$  est choisi pour représentant de la partie  $X_i \cup X_j$  et devient le père de  $s$ .

Par exemple, la réunion des parties  $X_1$  et  $X_3$  de la partition  $\mathcal{P}_9$  peut donner la nouvelle forêt représentée figure 4.



**Figure 4** Une représentation de  $\{\{0, 2\} \cup \{3, 4, 7\}, \{1, 5, 6, 8\}\}$

**Q 11.** Écrire une fonction

```
representant : int array -> int -> int
```

qui, appliquée à un tableau représentant une partition de  $S_n$  et à  $s \in S_n$ , renvoie le représentant de  $s$ .

**Q 12.** Écrire une fonction

```
union : int array -> int -> int -> unit
```

qui, appliquée à un tableau représentant une partition de  $S_n$  et à deux représentants  $s$  et  $t$  distincts, modifie la partition en réunissant les arbres associés à  $s$  et  $t$ , selon la méthode expliquée ci-dessus, sans oublier, si nécessaire, de modifier  $h(s)$  ou  $h(t)$ .

On note  $\mathcal{P}_n^{(0)}$  la partition de  $S_n$  où toutes les parties sont des singltons, c'est-à-dire  $\mathcal{P}_n^{(0)} = \{\{0\}, \{1\}, \dots, \{n-1\}\}$ .

**Q 13.** Soit  $\mathcal{P}$  une partition de  $S_n$  construite à partir de  $\mathcal{P}_n^{(0)}$  par des réunions successives selon la méthode précédente. Montrer que si  $s$  est le représentant d'une partie  $X \in \mathcal{P}$ , alors le cardinal de  $X$  vérifie  $|X| \geq 2^{h(s)}$ .

**Q 14.** En déduire les complexités des deux fonctions précédentes dans le pire des cas en fonction de  $n$  pour une partition  $\mathcal{P}$  construite à partir de  $\mathcal{P}_n^{(0)}$ .

**Q 15.** Écrire une fonction

```
est_un_arbre : graphe -> bool
```

qui, appliquée à un graphe  $g$  représentant un graphe  $G$ , renvoie **true** si  $G$  est un arbre et **false** sinon.

### III Algorithme de Wilson : arbre couvrant aléatoire

Si  $G = (S_n, A)$  est un graphe connexe et si  $r \in S_n$ , un *arbre couvrant de  $G$  enraciné en  $r$*  est un arbre  $\mathcal{T} = (S_n, B)$  tel que  $B \subset A$  et dont  $r$  a été choisi pour racine. On convient alors d'orienter les arêtes de l'arbre en direction de la racine  $r$  et de représenter  $\mathcal{T}$  par un tableau *parent* de taille  $n$ , *parent.(s)* étant égal au père de  $s$  dans  $\mathcal{T}$  si  $s \neq r$ , et à  $-1$  si  $s = r$ .

La figure 5 représente deux arbres couvrants du graphe  $G_{3,2}$ , enracinés respectivement en 0 et 2, et leurs représentations.



**Figure 5** Deux arbres couvrants enracinés de  $G_{3,2}$  et leurs représentations

Dans cette partie, nous supposons que  $G = (S_n, A)$  est un graphe connexe, représenté par  $g$ , que  $r$  est un sommet de  $G$  et nous cherchons à construire un arbre couvrant aléatoire de  $G$  enraciné en  $r$ .

Nous allons pour cela faire évoluer dynamiquement un arbre  $\mathcal{T}$ , représenté par un tableau *parent* de longueur  $n$  vérifiant :

- *parent.(r) = -1*;
- si  $s \in S_n$  n'est pas un sommet de  $\mathcal{T}$ , *parent.(s) = -2*;
- si  $s \in S_n$  est un sommet de  $\mathcal{T}$  autre que la racine, *parent.(s)* est le père de  $s$  dans l'arbre  $\mathcal{T}$ .

Nous partons de l'arbre réduit à la racine  $r$ , en posant  $\text{parent.(r)} = -1$  et  $\text{parent.(s)} = -2$  pour tout sommet  $s \neq r$ . Pour tout sommet  $s$ , quand  $s$  n'est pas déjà dans l'arbre, on construit un chemin aléatoire  $c$  sans boucle qui part de  $s$  et qui s'arrête dès qu'il a atteint un sommet de  $\mathcal{T}$ . La méthode de construction est la suivante :

- au début du calcul,  $c$  est réduit à  $s$  ;
- à tout moment,  $c$  est de la forme  $(s = s_0, s_1, \dots, s_{k-1}, s_k)$  où les sommets  $s_0, \dots, s_k$  sont deux à deux distincts et les sommets  $s_0, \dots, s_{k-1}$  ne sont pas des sommets de  $\mathcal{T}$  ;
- tant que l'extrémité  $s_k$  n'est pas un sommet de  $\mathcal{T}$ , on choisit aléatoirement et uniformément un voisin  $u$  de  $s_k$  et on distingue,
  - si  $u \in \{s_0, s_1, \dots, s_k\}$ , on supprime le cycle qui vient d'être formé et  $u$  devient le nouveau point d'arrivée du chemin  $c$  ;
  - sinon,  $c$  devient le chemin  $(s = s_0, s_1, \dots, s_{k-1}, s_k, u)$  ;
- on renvoie le chemin  $(s = s_0, s_1, \dots, s_{k-1}, s_k)$  une fois le calcul terminé.

On représentera un chemin en Caml par le type :

```
type chemin = {debut : int; mutable fin : int; suivant : int array}
```

de telle sorte que si le chemin  $c = (s_0, s_1, \dots, s_{k-1}, s_k)$  est représenté par  $c$ , alors  $c.\text{debut}$  est égal à  $s_0$ ,  $c.\text{fin}$  (qui est un champ mutable) est égal à  $s_k$ , et  $c.\text{suivant}$  est un tableau de taille  $n$  tel que pour  $j \in \{0, \dots, k-1\}$ , la case d'indice  $s_j$  de  $c.\text{suivant}$  contient le sommet  $s_{j+1}$ . Pour tout autre sommet  $t \notin \{s_0, s_1, \dots, s_{k-1}\}$ ,  $c.\text{suivant.(t)}$  pourra prendre une valeur arbitraire.

On rappelle que si  $c$  est un objet de type chemin et  $u$  un entier représentant un sommet, la modification du champ mutable  $c.\text{fin}$  pour y mettre  $u$  peut se faire avec la commande :

```
c.fin <- u
```

**Q 16.** Déterminer, dans le graphe  $G_{3,2}$ , le chemin représenté par l'objet

```
{debut = 1; fin = 4; suivant = [| -5; 2; 5; 3; -1; 4 |]}
```

**Q 17.** Que peut-on dire de la terminaison de cet algorithme ?

**Q 18.** Écrire une fonction

```
marche_aleatoire : int array array -> int array -> int -> chemin
```

telle que l'appel `marche_aleatoire adj parent s` renvoie l'objet  $c$  représentant un chemin de  $s$  à un sommet  $t \in \mathcal{T}$  obtenu comme décrit précédemment. On rappelle que `adj` représente le tableau des tableaux d'adjacence  $Adj$  du graphe  $G$ , que `parent` représente l'arbre (en construction)  $\mathcal{T}$  et on supposera que  $s \in S_n$  n'appartient pas à  $\mathcal{T}$ .

L'algorithme d'évolution de  $\mathcal{T}$ , appelé algorithme de Wilson, est alors le suivant : à partir de l'arbre  $\mathcal{T}$  réduit à  $r$ , pour  $s$  variant de 0 à  $n-1$ , si  $s$  n'est pas dans  $\mathcal{T}$ , on calcule un chemin de  $s$  à un sommet  $t \in \mathcal{T}$  codé par  $c$  grâce à la fonction `marche_aleatoire` et on `greffe c à T`, en ajoutant à  $\mathcal{T}$  les arêtes du chemin  $c$ , orientées de  $s$  vers  $u$ ,  $u$  étant le dernier sommet du chemin.

**Q 19.** Écrire une fonction

```
greffe : int array -> chemin -> unit
```

telle que l'instruction `greffe parent c` modifie `parent` de sorte à représenter l'arbre obtenu après la greffe du chemin  $c$  sur  $\mathcal{T}$ .

**Q 20.** Écrire une fonction

```
wilson : graphe -> int -> int array
```

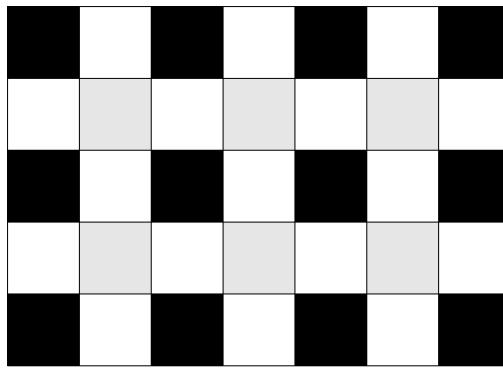
tel que `wilson g r` renvoie un arbre couvrant aléatoire du graphe  $G$  de racine  $r$ .

## IV Arbres couvrants et pavages par des dominos

Soient  $p$  et  $q$  deux entiers strictement supérieurs à 1. Le but des deux dernières parties est de construire des pavages aléatoires par des dominos de taille  $2 \times 1$  d'un échiquier à  $2p-1$  colonnes et  $2q-1$  lignes privé de son coin inférieur gauche (il reste bien un nombre pair de cases à recouvrir). Nous noterons  $E_{p,q}$  cet échiquier, dont les cases sont repérées par des couples de coordonnées entières  $(x, y)$ , avec  $0 \leq x \leq 2(p-1)$  et  $0 \leq y \leq 2(q-1)$ , et  $E'_{p,q}$  l'échiquier  $E_{p,q}$  privé de son coin inférieur gauche, c'est-à-dire de la case  $(0, 0)$ . Les cases de  $E_{p,q}$  sont colorées en noir, gris et blanc, comme dans l'exemple de l'échiquier  $E_{4,3}$  présenté figure 6.

Les cases dont les deux coordonnées sont paires sont colorées en noir, celles dont les deux coordonnées sont impaires sont colorées en gris, les autres sont colorées en blanc.

Si  $\mathcal{P}$  est un pavage de  $E'_{p,q}$ , chaque case noire ou grise de  $E'_{p,q}$  est recouverte par un domino, qui est orienté, à partir de la case noire ou grise considérée, vers le sud, l'ouest, le nord ou l'est.

**Figure 6** L'échiquier  $E_{4,3}$ 

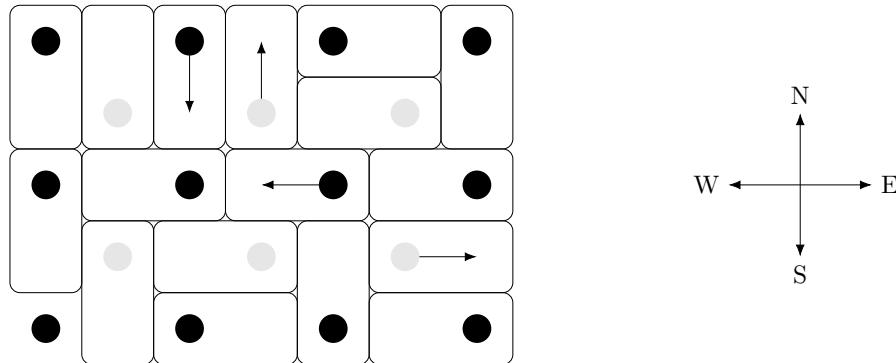
Nous définissons le type

```
type direction = S | W | N | E
```

et nous codons  $\mathcal{P}$  par une matrice `pavage` de taille  $(2p - 1) \times (2q - 1)$ , avec, pour  $i \in \{0, \dots, 2p - 2\}$  et  $j \in \{0, \dots, 2q - 1\}$  :

- si  $i$  et  $j$  ont la même parité et si  $(i, j) \neq (0, 0)$ , `p.(i).(j)` est la direction du domino qui recouvre la case  $(i, j)$  (cette case est soit noire, soit grise) ;
- sinon, `pavage.(i).(j)` prend la valeur N (ces valeurs ne jouent aucun rôle).

Ainsi, si  $p1$  est la matrice associée au pavage  $\mathcal{P}_1$  de  $E'_{4,3}$  représenté figure 7 (pour mieux distinguer les dominos, les cases ont été remplacées par des ronds colorés) on a par exemple  $p1.(2).(4) = S$ ,  $p1.(4).(2) = W$ ,  $p1.(3).(3) = N$ ,  $p1.(5).(1) = E$ , comme indiqué par les sens des flèches.

**Figure 7** Le pavage  $\mathcal{P}_1$  de  $E_{4,3}^*$ 

Les cases noires de l'échiquier  $E_{p,q}$  seront vues comme les sommets du graphe  $G_{p,q}$  (la case noire située en bas à gauche est identifiée au sommet 0 du graphe  $G_{p,q}$ ). Un résultat de Neville Temperley explicite une bijection canonique  $\varphi$  entre les pavages de  $E'_{p,q}$  et les arbres couvrants de  $G_{p,q}$ . La construction de  $\varphi(\mathcal{P})$  à partir d'un pavage  $\mathcal{P}$  s'obtient en ne conservant que les dominos recouvrant une case noire.

Pour toute la suite du sujet, les valeurs de  $p$  et  $q$  seront supposées contenues dans les variables globales `p` et `q`. On suppose de plus définie une variable globale `g`, représentant  $G_{p,q}$ , par l'instruction

```
let g = quadrillage p q;;
```

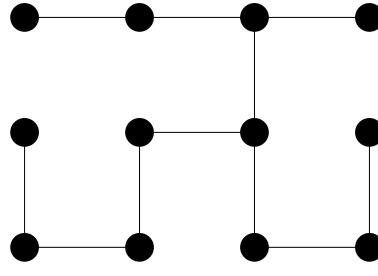
#### **IV.A – Exemples**

**Q 21.** Dessiner l'arbre couvrant *non enraciné*  $\mathcal{T}_1 = \varphi(\mathcal{P}_1)$  de  $G_{4,3}$ .

**Q 22.** Considérons inversement l'arbre couvrant  $\mathcal{T}_2$  de  $G_{4,3}$  décrit figure 8. Par un procédé réciproque à celui utilisé à la question précédente, dessiner le pavage  $\mathcal{P}_2 = \varphi^{-1}(\mathcal{T}_2)$  de  $E'_{4,3}$ .

#### **IV.B – Calcul de l'arbre couvrant associé à un pavage**

**Q 23.** Soit  $\mathcal{P}$  un pavage de  $E'_{p,q}$ , associé à l'arbre couvrant  $\mathcal{T} = \varphi(\mathcal{P})$  de  $G_{p,q}$ . Décrire un procédé permettant de calculer le père d'un sommet  $s \in \{1, \dots, pq - 1\}$  dans l'arbre  $\mathcal{T}$  enraciné en 0. On ne demande pas de démontrer que la structure ainsi obtenue est bien un arbre de racine 0.

Figure 8 L'arbre couvant  $\mathcal{T}_2$ 

**Q 24.** Écrire une fonction

```
coord_noire : int -> int * int
```

telle que l'instruction `coord_noire s` renvoie le couple des coordonnées de la case correspondant au sommet  $s$  du graphe  $G_{p,q}$ . Par exemple, dans le graphe  $G_{4,3}$  représenté figure 2, `coord_noire 6` renverra (4, 2).

**Q 25.** Écrire une fonction

```
sommet_direction : int -> direction -> int
```

telle que `sommet_direction s d` renvoie le sommet  $t$  qui se trouve directement dans la direction  $d$  du sommet  $s$  de  $G_{p,q}$ . Cette fonction renverra -1 si un tel sommet n'existe pas. Par exemple, dans le graphe  $G_{4,3}$  représenté figure 2, `sommet_direction 5 W` renverra 4.

**Q 26.** Écrire une fonction

```
phi : direction array array -> int array
```

qui, appliquée à une matrice pavage représentant un pavage  $\mathcal{P}$  de  $E'_{p,q}$ , renvoie le tableau `parent` représentant l'arbre  $\mathcal{T} = \varphi(\mathcal{P})$  enraciné en 0 (cette représentation est définie au début de la partie III).

## V Utilisation du dual pour la construction d'un pavage

### V.A – Graphe dual de $G_{p,q}$

Le graphe  $G_{p,q}$  est un *graphe planaire*, c'est-à-dire qu'il possède une représentation graphique dans laquelle deux arêtes distinctes ne se coupent pas, sauf peut-être en leurs extrémités. Cette *représentation planaire* sépare le plan en  $(p-1)(q-1)+1$  faces, que l'on va numérotter de gauche à droite et de bas en haut, la face non bornée portant le numéro 0. Le *graphe dual* de  $G_{p,q}$ , noté  $G^*_{p,q}$ , est alors le graphe à  $(p-1)(q-1)+1$  sommets (chaque sommet correspond à une des faces délimitées par la représentation de  $G_{p,q}$ ) et qui possède autant d'arêtes que  $G_{p,q}$  : chaque arête  $a$  de  $G_{p,q}$  donne une arête  $a^*$  entre les deux faces du plan qu'elle sépare. Ce graphe est également planaire. Attention, contrairement au graphe  $G_{p,q}$ , le graphe  $G^*_{p,q}$  peut posséder plusieurs arêtes entre les mêmes sommets. Les sommets de  $G_{p,q}$  ont déjà été identifiés aux cases noires de  $E_{p,q}$  ; les cases grises seront identifiées aux sommets de  $G^*_{p,q}$  distincts de 0 et les cases blanches aux arêtes de  $G_{p,q}$  (et donc également aux arêtes de  $G^*_{p,q}$ , puisque  $a \mapsto a^*$  est une bijection). La figure 9 explicite ces identifications dans le cas du graphe  $G_{4,3}$ . Les sommets de  $G_{4,3}$  sont identifiés par des sommets gris foncés, ceux de  $G^*_{4,3}$  par des sommets gris clairs, les arêtes de  $G_{4,3}$  par des traits pleins et celles de  $G^*_{4,3}$  par des traits hachurés. L'intersection de chaque  $(a, a^*)$  est marquée par un carré symbolisant une case blanche de  $E_{4,3}$ .

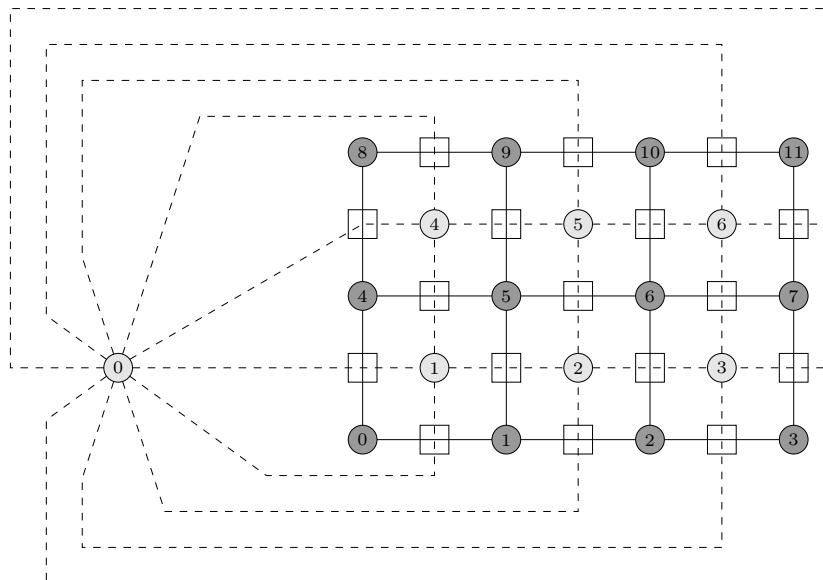
Ainsi, le sommet 6 de  $G_{4,3}$  correspond à la case (4, 2) de  $E_{4,3}$ , le sommet 4 de  $G^*_{4,3}$  correspond à la case (1, 3) et les arêtes  $a_9$  de  $G_{4,3}$  et  $a_9^*$  de  $G^*_{4,3}$  correspondent à la case (3, 0).

Dans la suite du problème, nous notons :

- $n = pq$  le nombre de sommets de  $G_{p,q}$  ;
- $n^* = (p-1)(q-1)+1$  le nombre de sommets de  $G^*_{p,q}$  ;
- $m = p(q-1) + q(p-1)$  le nombre d'arêtes de  $G_{p,q}$  et de  $G^*_{p,q}$  ;
- $A$  l'ensemble des arêtes de  $G_{p,q}$  ;
- $A^*$  l'ensemble des arêtes de  $G^*_{p,q}$ .

Nous supposerons définies pour la suite :

- une fonction `coord_grise : int -> int * int` qui, appliquée à un sommet de  $G^*_{p,q}$  autre que 0, renvoie les coordonnées de la case grise qui correspond à ce sommet ;
- une fonction `numero : int * int -> int` qui, appliquée à un couple  $(x, y)$  représentant une case noire ou grise de l'échiquier  $E_{p,q}$ , renvoie le sommet du graphe  $G_{p,q}$  ou  $G^*_{p,q}$  associé à cette case. On supposera également que dans tous les autres cas, `numero` renvoie la valeur 0, y compris si les coordonnées sont en dehors de l'échiquier. Quand  $p = 4$  et  $q = 3$ , nous avons par exemple : `numero (4, 2) = 6`, `numero (1, 3) = 4`, et `numero (3, 0) = 0`.

Figure 9  $G_{4,3}$  et  $G_{4,3}^*$ 

**Q 27.** En utilisant les fonctions précédentes, écrire une fonction

```
dual : unit -> graphe
```

telle que l'instruction `dual ()` renvoie le graphe représentant  $G_{p,q}^*$ . On respectera la numérotation des sommets de  $G_{p,q}^*$  décrite ci-dessus (figure 9). Par ailleurs, les listes d'adjacence du graphe dual contiendront des sommets en doublet s'il existe plusieurs arêtes entre des mêmes sommets.

On suppose désormais définie une variable globale `g_etoile` par l'instruction

```
let g_etoile = dual () ;
```

#### V.B – Dual d'un arbre couvrant

**Q 28.** Montrer que, quitte à renuméroter les sommets, le dual de  $G_{p,q}^*$  est le graphe  $G_{p,q}$ .

Soit  $B$  une partie de  $A$ . On note  $B^*$  la partie de  $A^*$  définie par

$$\forall i \in \{0, 1, \dots, m-1\}, a_i^* \in B^* \iff a_i \notin B.$$

**Q 29.** Montrer que si le graphe  $(S_n, B)$  possède un cycle, le graphe  $(S_{n^*}, B^*)$  n'est pas connexe.

**Q 30.** En déduire que  $(S_n, B)$  est un arbre couvrant de  $G_{p,q}$  si et seulement si  $(S_{n^*}, B^*)$  est un arbre couvrant de  $G_{p,q}^*$ .

Si  $\mathcal{T} = (S_n, B)$  est un arbre couvrant de  $G_{p,q}$ , nous noterons  $\mathcal{T}^*$  l'arbre couvrant  $(S_{n^*}, B^*)$  de  $G_{p,q}^*$ .

Pour construire l'arbre  $\mathcal{T}^*$ , nous utiliserons une représentation différente des arbres que celle introduite en partie III : on pourra représenter un arbre couvrant  $\mathcal{T} = (S, B)$  enraciné en  $r$  par un couple `(r, b)` où `b` est un tableau de booléens de longueur  $m$  représentant  $B$ , c'est-à-dire tel que `b.(i)` prend la valeur `true` si et seulement si  $a_i \in B$ .

La figure 10 représente les deux arbres couvrants du graphe  $G_{3,2}$ , présenté en figure 5, enracinés respectivement en 0 et 2, et leurs représentations par un couple.



```
(0, [| true; true; false; true; true; false; true |])      (2, [| true; true; true; true; false; false |])
```

Figure 10 Deux arbres couvrants enracinés de  $G_{3,2}$  et leurs représentations

**Q 31.** Écrire une fonction

```
vers_couple : int array -> int * bool array
```

telle que l'instruction `vers_couple parent`, où `parent` est un tableau représentant un arbre enraciné de  $G_{p,q}$ , renvoie le couple `(r, b)` décrit ci-dessus.

**Q 32.** Détails en français un algorithme permettant de construire `parent` à partir du couple  $(r, B)$  et de la représentation du graphe  $G_{p,q}$  par listes d'adjacences. Cet algorithme est-il applicable à un arbre couvrant du graphe  $G_{p,q}^*$ ? Justifier.

**Q 33.** Écrire une fonction

```
vers_parent : int * bool array -> int array
```

telle que l'instruction `vers_parent (r, b)`, où  $b$  désigne un ensemble  $B$  tel que  $(S_n, B)$  est un arbre de  $G_{p,q}$  enraciné en  $r$ , renvoie le tableau `parent` représentant cet arbre.

**Q 34.** Déterminer les complexités de ces deux fonctions de conversion en fonction de  $n$  et  $m$ .

On supposera écrite une fonction `vers_parent_etoile : int * bool array -> int array`, ayant un fonctionnement similaire à `vers_parent`, qui prend en argument un couple  $(r, b\_etoile)$  correspondant à un arbre couvrant  $\mathcal{T}^* = (S_{n^*}, B^*)$  de  $G_{p,q}^*$  et renvoie un tableau `parent_etoile` décrivant l'arbre en utilisant la représentation décrite partie III.

**Q 35.** Écrire une fonction

```
arbre_dual : int array -> int array
```

qui, appliquée au tableau `parent` représentant un arbre couvrant  $\mathcal{T}$  de  $G_{p,q}$  enraciné en 0, renvoie le tableau `parent_etoile` représentant l'arbre couvrant  $\mathcal{T}^*$  de  $G_{p,q}^*$  enraciné en 0.

#### V.C – Calcul du pavage associé à un arbre couvrant

Soit  $\mathcal{T} = (S_n, B)$  un arbre couvrant de  $G_{p,q}$ .

**Q 36.** Décrire un procédé de construction, à partir des arbres  $\mathcal{T}$  et  $\mathcal{T}^*$  enracinés en 0, du pavage  $\mathcal{P} = \varphi^{-1}(\mathcal{T})$  de  $E'_{p,q}$ . On expliquera en détail comment traiter les arêtes d'un sommet de  $G_{p,q}^*$  vers le sommet 0. On justifiera brièvement que l'on obtient bien un pavage.

**Q 37.** Écrire une fonction

```
pavage_aleatoire : unit -> direction array array
```

telle que l'instruction `pavage_aleatoire ()` renvoie une matrice de taille  $(2p - 1)(2q - 1)$  représentant un pavage aléatoire de  $E'_{p,q}$  par des dominos.

**Q 38.** Comment adapter cette méthode à la construction de pavages aléatoires d'un échiquier à  $2p$  colonnes et  $2q - 1$  lignes (respectivement à  $2p$  colonnes et  $2q$  lignes)?

• • • FIN • • •

SESSION 2021

MP7IN

**ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH****ÉPREUVE SPÉCIFIQUE - FILIÈRE MP****INFORMATIQUE****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.*

**RAPPEL DES CONSIGNES**

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

**Les calculatrices sont interdites.**

**Le sujet est composé de trois parties indépendantes.**

## Partie I - Étude des partitions non croisées

Dans cette partie, on introduit et on étudie de façon élémentaire les partitions non croisées, objets combinatoires apparaissant dans divers domaines des mathématiques, notamment dans la théorie des probabilités libres et des matrices aléatoires.

Le langage de programmation utilisé dans cette partie est le langage Python.

Dans la suite, pour tout couple  $(i, n) \in \mathbb{N}^2$ , les notations  $\llbracket n \rrbracket$  et  $\llbracket i, n \rrbracket$  désignent respectivement les ensembles  $\{1, 2, \dots, n\}$  et  $\{i, i + 1, \dots, n\}$ . Par convention,  $\llbracket 0 \rrbracket = \emptyset$  et si  $i > n$  alors  $\llbracket i, n \rrbracket = \emptyset$ .

### Définition 1 (Partition)

Soit  $A$  un sous-ensemble non vide de  $\mathbb{N}$ . Une **partition**  $\mathcal{P}$  de  $A$  est un ensemble de parties de  $A$  tel que :

1.  $\forall P \in \mathcal{P}, P \neq \emptyset$ ;
2.  $\forall (P, Q) \in \mathcal{P}^2, (P \neq Q) \implies (P \cap Q = \emptyset)$ ;
3.  $\bigcup_{P \in \mathcal{P}} P = A$ .

Par exemple,  $\{\{2, 4, 5\}, \{7, 9\}, \{8\}\}$  est une partition de l'ensemble  $\{2, 4, 5, 7, 8, 9\}$ .

### Définition 2 (Classe)

Soit  $\mathcal{P}$  une partition d'un sous-ensemble  $A$  non vide de  $\mathbb{N}$ . Soit  $i$  un élément de  $A$ . La **classe** de  $i$  est l'unique élément  $P$  de  $\mathcal{P}$  tel que  $i \in P$ . Elle est notée  $\text{Cl}(i)$ .

Par exemple, pour  $\mathcal{P} = \{\{2, 4, 5\}, \{7, 9\}, \{8\}\}$ , on a  $\text{Cl}(4) = \{2, 4, 5\}$ .

### Définition 3 (Partition non croisée)

Soit  $\mathcal{P}$  une partition d'un sous-ensemble  $A$  non vide de  $\mathbb{N}$ . On dit que  $\mathcal{P}$  est **non croisée** si pour tout quadruplet  $(a, b, c, d) \in A^4$  tel que  $a < b < c < d$  on a :

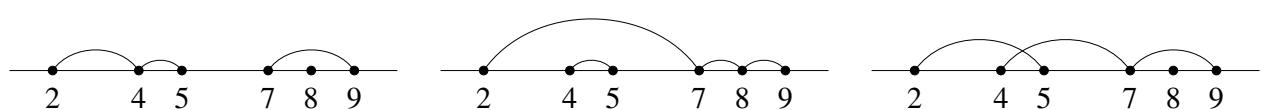
$$(\text{Cl}(a) = \text{Cl}(c), \text{Cl}(b) = \text{Cl}(d)) \implies (\text{Cl}(a) = \text{Cl}(b) = \text{Cl}(c) = \text{Cl}(d)).$$

Par exemple,  $\mathcal{P} = \{\{2, 4, 5\}, \{7, 9\}, \{8\}\}$  est bien une partition non croisée de  $\{2, 4, 5, 7, 8, 9\}$ . En effet, aucun quadruplet  $(a, b, c, d) \in \{2, 4, 5, 7, 8, 9\}^4$  tel que  $a < b < c < d$  ne vérifie la condition  $\text{Cl}(a) = \text{Cl}(c), \text{Cl}(b) = \text{Cl}(d)$ .

De même,  $\mathcal{Q} = \{\{2, 7, 8, 9\}, \{4, 5\}\}$  en est bien une. En effet,  $(a, b, c, d) = (2, 7, 8, 9)$  est l'unique quadruplet tel que  $a < b < c < d$  vérifiant  $\text{Cl}(a) = \text{Cl}(c), \text{Cl}(b) = \text{Cl}(d)$  et ces quatre éléments sont bien dans la même classe.

Par contre,  $\mathcal{R} = \{\{2, 5\}, \{4, 7, 9\}, \{8\}\}$  n'en est pas une. En effet,  $\text{Cl}(2) = \text{Cl}(5)$  et  $\text{Cl}(4) = \text{Cl}(7)$  mais  $\text{Cl}(2) \neq \text{Cl}(4)$ .

Le terme « non croisée » découle naturellement des représentations picturales des partitions (**figure 1**).



**Figure 1** - Représentations picturales de  $\mathcal{P}, \mathcal{Q}, \mathcal{R}$

Dans la suite, on s'intéresse uniquement aux partitions d'un sous-ensemble fini de  $\mathbb{N}$ . On les représente en Python à l'aide de listes de listes croissantes d'entiers triées dans l'ordre lexicographique.

Par exemple, les partitions  $\mathcal{P} = \{\{2, 4, 5\}, \{7, 9\}, \{8\}\}$  et  $\mathcal{R} = \{\{2, 5\}, \{4, 7, 9\}, \{8\}\}$  sont respectivement représentées par les listes  $P = [[2, 4, 5], [7, 9], [8]]$  et  $R = [[2, 5], [4, 7, 9], [8]]$ .

### I.1 - Exemples et fonctions élémentaires (Informatique Pour Tous)

**Q1.** Justifier brièvement que  $\mathcal{P} = \{\{1, 7\}, \{2\}, \{3, 4, 5\}, \{6\}\}$  est une partition non croisée de  $\llbracket 7 \rrbracket$  et que  $\mathcal{Q} = \{\{1, 6\}, \{2\}, \{3, 4, 5, 7\}\}$  n'en est pas une.

**Q2.** Parmi les ensembles suivants, indiquer sans justification lesquels sont des partitions de  $\llbracket 5 \rrbracket$ .  
Parmi les partitions, préciser sans justification lesquelles sont non croisées.

1.  $\mathcal{P}_1 = \{\{1, 3\}, \{2, 4, 5\}\},$
2.  $\mathcal{P}_2 = \{\{1, 3\}, \{1, 2\}, \{4, 5\}\},$
3.  $\mathcal{P}_3 = \{\{1, 3\} \{2, 4\}\},$
4.  $\mathcal{P}_4 = \{\{1, 4, 5\}, \{2, 3\}\}.$

**Q3.** Décrire l'ensemble des partitions non croisées de  $\llbracket 4 \rrbracket$ .

Pour **Q4**, **Q5** et **Q6**, on se fixe un ensemble fini  $A \subset \mathbb{N}$ . Cet ensemble est représenté en Python par la liste croissante d'entiers  $A$ , définie au préalable. On suppose également que pour ces questions, la variable  $P$  désigne une liste de listes croissantes d'entiers triée dans l'ordre lexicographique.

On définit la fonction Python `mystere(P)` par :

```
def mystere(P) :
    L = [False for _ in range(len(A))]
    for i in range(len(P)) :
        if P[i] == [] :
            return False
        else :
            for j in range(len(P[i])) :
                if P[i][j] not in A :
                    return False
                else :
                    # A.index(a) renvoie l'indice de a dans A
                    k = A.index(P[i][j])
                    if L[k] :
                        return False
                    else :
                        L[k] = True
    return not (False in L)
```

**Q4.** Uniquement dans cette question, on suppose que  $A = [1, 2, 3, 4, 5]$ .  
Explicit sans justification les valeurs de :

1. `mystere([[1,3],[1,5],[2,4]])`
2. `mystere([[1,3,4],[2]])`
3. `mystere([[1,3,5],[2,4]])`
4. `mystere([[],[1,2,3,4,5]])`.

**Q5.** Écrire une fonction Python `classe(P, i)` prenant en arguments une variable  $P$  représentant une partition  $\mathcal{P}$  de  $A$  et un entier  $i \in A$  qui renvoie une liste  $L$  représentant  $C_i(i)$ .

On définit la fonction Python au code incomplet `est_nc(P)` suivante :

```
def est_nc (P) :
    # On rappelle que A est une liste triée dans l'ordre croissant
    N = len(A)
    for i in range(N) :
        for j in range(i+1,N) :
            for k in range(j+1,N) :
                for l in range(k+1,N) :
                    . . .
```

- Q6.** Recopier et compléter le code de la fonction `est_nc(P)`, sachant qu'elle prend en argument une variable `P` représentant une partition  $\mathcal{P}$  de  $A$  et qu'elle renvoie `True` si  $\mathcal{P}$  est non croisée et `False` sinon.

## I.2 - Nombre de partitions non croisées

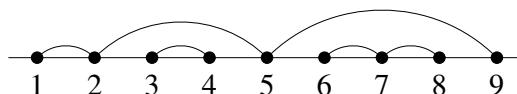
Pour tout  $n \geq 1$ , on note  $C_n$  le nombre de partitions non croisées d'un ensemble  $A \subset \mathbb{N}$  ayant exactement  $n$  éléments et  $\text{NC}(A)$  l'ensemble des partitions non croisées de  $A$ . Par convention,  $C_0 = 1$  et  $\text{NC}(\emptyset) = \{\emptyset\}$ .

### Définition 4 (Partition non croisée emboîtée)

Soient  $A$  un ensemble fini de  $\mathbb{N}$  de cardinal  $n$  et  $\mathcal{P}$  une partition non croisée de  $A$ . On dit que  $\mathcal{P}$  est **emboîtée** si le minimum  $m$  et le maximum  $M$  de  $A$  sont dans la même classe ( $\text{Cl}(m) = \text{Cl}(M)$ ). L'ensemble des partitions non croisées et emboîtées de  $A$  et son cardinal sont respectivement notés  $\text{NCE}(A)$  et  $D_n$ .

Par exemple,  $\mathcal{S} = \{\{1, 2, 5, 9\}, \{3, 4\}, \{6, 7, 8\}\}$  est une partition non croisée et emboîtée de  $\llbracket 9 \rrbracket$ .

Le terme « emboîtée » découle naturellement des représentations picturales des partitions (**figure 2**).



**Figure 2 - Représentations picturale de  $\mathcal{S}$**

- Q7.** Montrer que pour tout  $n \in \mathbb{N}$ , on a  $C_n = D_{n+1}$ .

- Q8.** Soit  $n \in \mathbb{N}$ . On définit l'application  $\Phi_n$  comme suit :

$$\begin{aligned} \Phi_n : \bigcup_{i=1}^{n+1} \text{NCE}(\llbracket i \rrbracket) \times \text{NC}(\llbracket i+1, n+1 \rrbracket) &\rightarrow \text{NC}(\llbracket n+1 \rrbracket) \\ (\mathcal{Q}_1, \mathcal{Q}_2) &\mapsto \mathcal{Q}_1 \cup \mathcal{Q}_2. \end{aligned}$$

En admettant que  $\Phi_n$  est une application bijective, montrer que  $C_{n+1} = \sum_{k=0}^n C_k C_{n-k}$ .

- Q9.** (Informatique Pour Tous) Écrire une fonction Python `calcul_C(n)` qui prend en argument un entier naturel  $n$  et qui renvoie la valeur  $C_n$ .

## Partie II - Logique et étude du problème Horn-Sat

Dans cette partie,  $\wedge, \vee, \neg$ , désignent respectivement les connecteurs de conjonction, de disjonction et de négation. Les symboles V, F désignent respectivement les valeurs de vérité VRAI et FAUX du calcul propositionnel.

Dans la suite, on s'intéresse au problème Horn-Sat, qui peut être décrit de la façon suivante : étant donnée une formule de Horn  $P$ , existe-t-il une interprétation  $I$  telle que  $[P]_I = \text{V}$  ?

L'objectif est d'expliciter un algorithme permettant de résoudre ce problème.

### Définition 5 (Formule propositionnelle)

Soit  $X$  un ensemble de symboles appelés variables propositionnelles. Les **formules propositionnelles** sur  $X$  sont alors définies de façon inductive comme suit :

- V, F sont des formules propositionnelles ;
- tout élément  $x$  de  $X$  est une formule propositionnelle ;
- si  $P$  et  $Q$  sont des formules propositionnelles, alors  $\neg P, (P \wedge Q), (P \vee Q)$  sont des formules propositionnelles.

Dans la suite, les variables propositionnelles et formules propositionnelles considérées sont construites à l'aide d'un ensemble  $X$  qui ne sera pas explicité.

### Définition 6 (Littéral)

Soit  $x$  une variable propositionnelle. Un **littéral** est la formule  $x$  ou la formule  $\neg x$ . Le littéral  $x$  (respectivement  $\neg x$ ) est un littéral positif (respectivement négatif).

### Définition 7 (Clause disjonctive)

Soient  $n \in \mathbb{N}, x_1, x_2, \dots, x_n$  des littéraux. La formule  $x_1 \vee x_2 \vee \dots \vee x_n$  est appelée **clause disjonctive** à  $n$  littéraux (ou de taille  $n$ ). Par convention, () désigne la **clause disjonctive vide**, c'est-à-dire la clause sans littéral.

Une **clause de Horn** est une clause disjonctive contenant au plus un littéral positif.

Une **clause unitaire** est une clause composée d'un unique littéral.

### Définition 8 (Forme normale conjonctive)

Soit  $P$  une formule propositionnelle. On dit que  $P$  est une **forme normale conjonctive** s'il existe un entier  $n \in \mathbb{N}, C_1, C_2, \dots, C_n$  des clauses disjonctives vérifiant  $P = C_1 \wedge C_2 \wedge \dots \wedge C_n$ . Par convention, une forme normale conjonctive sans clause est représentée par  $\emptyset$ .

### Définition 9 (formule de Horn)

Soit  $P$  une formule propositionnelle. On dit que  $P$  est une **formule de Horn** s'il existe  $n \in \mathbb{N}, C_1, C_2, \dots, C_n$  des clauses de Horn vérifiant  $P = C_1 \wedge C_2 \wedge \dots \wedge C_n$ .

### Définition 10 (Interprétation)

Soient  $n \in \mathbb{N}$  et  $x_1, x_2, \dots, x_n$  des variables propositionnelles. Une **interprétation** est une application  $I : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{V}, \text{F}\}$ .

### Définition 11 (Évaluation)

Soient  $P$  une formule propositionnelle,  $x_1, x_2, \dots, x_n$  les variables propositionnelles apparaissant dans  $P$  et  $I$  une interprétation sur  $\{x_1, x_2, \dots, x_n\}$ . L'**évaluation** de  $P$  suivant l'interprétation  $I$  que l'on note  $[P]_I$  est définie par récurrence de la façon suivante :

- si  $P \in \{\text{V, F}\}$ , alors  $[P]_I = P$ ;
- si  $P \in \{x_1, x_2, \dots, x_n\}$ , alors  $[P]_I = I(P)$ ;
- si  $P = \neg Q$  où  $Q$  est une formule propositionnelle, alors  $[P]_I = \neg[Q]_I$ ;
- si  $P = A \circ B$  où  $A, B$  sont des formules propositionnelles et  $\circ \in \{\wedge, \vee\}$ , alors  $[P]_I = [A]_I \circ [B]_I$ .

Par convention,  $[()_I = \text{F}$  et  $[\emptyset]_I = \text{V}$ .

#### Définition 12 (Satisfiable)

Soit  $P$  une formule propositionnelle. On dit que  $P$  est **satisfiable** s'il existe une interprétation  $I$  telle que  $[P]_I = \text{V}$ .

**Q10.** Pour chacune des formules suivantes, montrer qu'elle est satisfiable ou qu'elle est non satisfiable.

1.  $P_1 = (x \vee y) \wedge (\neg x \vee \neg y) \wedge (x \vee \neg y)$ ,
2.  $P_2 = (x) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee z) \wedge (z)$ ,
3.  $P_3 = ()$ ,
4.  $P_4 = (x \vee \neg y \vee \neg t) \wedge (z \vee \neg t \vee \neg x \vee \neg y)$ .

**Q11.** Parmi les formules de la **Q10**, lesquelles sont des formules de Horn ? On ne justifiera pas la réponse.

#### Définition 13 (Propagation unitaire)

Soit  $P$  une forme normale conjonctive contenant une clause unitaire  $(x)$ . On construit une formule  $P'$  à partir de  $P$  de la façon suivante :

1. supprimer de  $P$  toutes les clauses où  $x$  apparaît ;
2. enlever toutes les occurrences du littéral  $\neg x$ .

Cette procédure de simplification est appelée **propagation unitaire**.

Par exemple, si  $P = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x)$ , on a alors  $P' = (y \vee z)$ . Si  $P = (x) \wedge (\neg x)$ , on a alors  $P' = ()$ .

Dans la suite, on note  $\Pi(P)$  une formule sans clause unitaire obtenue en itérant la propagation unitaire sur  $P$ . On admet que si  $\Pi(P)$  ne contient pas de clause vide  $()$  alors  $\Pi(P)$  est unique et que si  $\Pi(P)$  contient au moins une clause vide alors toute formule sans clause unitaire obtenue par itération de la propagation unitaire sur  $P$  contient au moins une clause vide.

**Q12.** On pose :

$$\begin{aligned} P &= (x_1) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_1 \vee x_3) \\ &\quad \wedge (x_3 \vee \neg x_4 \vee x_5) \wedge (\neg x_1 \vee x_2 \vee x_5). \end{aligned}$$

Calculer  $\Pi(P)$ . On pourra donner le résultat directement sans détailler les calculs.

**Q13.** Soit  $P$  une formule de Horn. Montrer que  $\Pi(P)$  est une formule de Horn.

**Q14.** Soit  $P$  une forme normale conjonctive. Montrer que  $P$  est satisfiable si et seulement si  $\Pi(P)$  est satisfiable.

**Q15.** Soit  $P$  une forme normale conjonctive. Montrer que si  $()$  apparaît dans  $\Pi(P)$ , alors  $P$  n'est pas satisfiable.

**Q16.** Soit  $C$  une clause de Horn. Montrer que si  $C$  n'est ni la clause vide ni une clause unitaire positive, alors  $C$  est satisfiable.

**Q17.** Soit  $P$  une formule de Horn ne contenant ni de clause vide ni de clause unitaire positive. Montrer que  $P$  est satisfiable.

**Q18.** Soit  $P$  une formule de Horn. Montrer que  $P$  n'est pas satisfiable si et seulement si  $()$  apparaît dans  $\Pi(P)$ .

## Partie III - Étude des classes sylvestres

Étant donné un arbre binaire de recherche  $T$ , sa classe sylvestre est l'ensemble des mots qui donnent l'arbre  $T$  après insertion dans l'arbre vide. L'objectif de cette partie est de donner une caractérisation et une description de cet ensemble. On commence par rappeler la structure des arbres binaires de recherche ainsi que leurs propriétés usuelles. Puis, on introduit la notion de S-équivalence sur les mots qui permet de caractériser la classe sylvestre d'un arbre  $T$ . Enfin, à l'aide du produit de mélange, on présente un algorithme calculant la classe sylvestre d'un arbre donné.

Dans toute la suite,  $\Sigma$  désigne un alphabet fini totalement ordonné. Le symbole  $\varepsilon$  désigne le mot vide.

### III.1 - Algorithme d'insertion dans un arbre binaire

#### Définition 14 (Arbre binaire)

Un **arbre binaire**  $T$  (étiqueté par les éléments de  $\Sigma$ ) est récursivement soit :

- l'arbre vide que l'on note  $\circ$  ;
- un triplet  $(T_g, r, T_d)$  où  $r$  est un élément de  $\Sigma$ ,  $T_g$  et  $T_d$  des arbres binaires. Les éléments  $r$ ,  $T_g$  et  $T_d$  sont respectivement appelés racine, sous-arbre gauche et sous-arbre droit de  $T$ .

#### Définition 15 (Arbre binaire de recherche)

Un **Arbre Binaire de Recherche** (abrégé en **ABR**)  $T$  est récursivement soit :

- l'arbre vide ;
- un triplet  $(T_g, r, T_d)$  où  $r$  est un élément de  $\Sigma$ ,  $T_g$  et  $T_d$  des **ABR**. De plus, toute valeur apparaissant dans  $T_g$  est strictement inférieure à  $r$  et toute valeur apparaissant dans  $T_d$  est supérieure ou égale à  $r$ .

#### Définition 16 (Insertion dans un arbre binaire)

L'**insertion** d'un élément  $a$  de  $\Sigma$  dans un arbre binaire  $T$  est une opération que l'on note  $T \leftarrow a$  définie récursivement comme suit :

- si  $T = \circ$ , alors  $T \leftarrow a = (\circ, a, \circ)$  ;
- si  $T = (T_g, r, T_d)$  et  $r \leq a$ , alors  $T \leftarrow a = (T_g, r, T_d \leftarrow a)$  ;
- si  $T = (T_g, r, T_d)$  et  $r > a$ , alors  $T \leftarrow a = (T_g \leftarrow a, r, T_d)$ .

On définit récursivement alors l'insertion d'un mot  $w$  dans un arbre binaire  $T$  noté également  $T \leftarrow w$  comme suit :

- si  $w = \varepsilon$ , alors  $T \leftarrow w = T$  ;
- si  $w = av$  avec  $a \in \Sigma$ , alors  $T \leftarrow w = (T \leftarrow a) \leftarrow v$ .

Dans le cas où  $T$  est un **ABR**, on admet que  $T \leftarrow a$  et  $T \leftarrow w$  sont également des **ABR**.

Étant donné un mot  $w$  sur  $\Sigma$ , l'**arbre binaire de recherche associé** à  $w$  est l'arbre  $\circ \leftarrow w$ .

#### Définition 17 (Classe sylvestre)

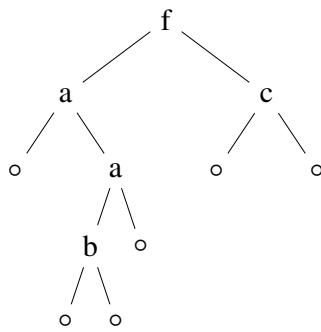
Soit  $T$  un arbre binaire de recherche. La **classe sylvestre** de  $T$  que l'on note  $Syl(T)$  est l'ensemble des mots  $w$  vérifiant :  $(\circ \leftarrow w) = T$ .

Par exemple, si  $T = ((\circ, a, \circ), b, (\circ, c, \circ))$ , on a  $Syl(T) = \{bac, bca\}$ .

Et si  $T = \circ$ , alors  $Syl(T) = \{\varepsilon\}$ .

Pour simplifier la représentation des arbres binaires, on peut utiliser des graphes.

Par exemple, l'arbre  $T = ((\circ, a, ((\circ, b, \circ), a, \circ)), f, (\circ, c, \circ))$  est représenté dans la **figure 3**.

**Figure 3 - Représentation de  $T$** 

Dans la suite, les lettres de  $\Sigma$  sont représentées en Ocaml par des `char` et les mots sur l'alphabet  $\Sigma$  par des `char list`. Ainsi, le mot  $w = "arbre"$  est représenté par la liste  $w = ['a'; 'r'; 'b'; 'r'; 'e']$ . On représente les arbres binaires en Ocaml à l'aide de la structure `arbre` suivante :

```
type 'a arbre = Vide | Noeud of ('a arbre) * 'a * ('a arbre).
```

Ainsi, l'arbre  $T = (\circ, a, (\circ, b, \circ))$  est représenté en Ocaml par :

```
Noeud(Vide, 'a', Noeud(Vide, 'b', Vide)).
```

**Q19.** Représenter le graphe de l'**ABR** associé au mot "fantastique", l'ordre sur les lettres étant l'ordre alphabétique.

**Q20.** Écrire une fonction récursive en Ocaml de signature :

```
insertion_lettre : char -> char arbre -> char arbre
```

et telle que `insertion_lettre a t` est l'arbre binaire obtenu en insérant la lettre `a` dans l'arbre binaire `t`.

**Q21.** Écrire une fonction récursive en Ocaml de signature :

```
insertion_mot : char list -> char arbre -> char arbre
```

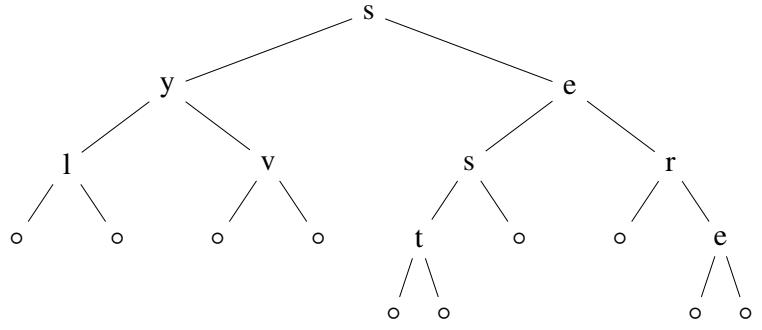
et telle que `insertion_mot w t` est l'arbre binaire obtenu en insérant le mot `w` dans l'arbre binaire `t`.

#### Définition 18 (Lecture préfixe)

Soit  $T$  un arbre binaire. La **lecture préfixe** de  $T$  que l'on note  $w_T$  est le mot défini récursivement par :

- si  $T = \circ$ , alors  $w_T = \varepsilon$ ;
- si  $T = (T_g, r, T_d)$ , alors  $w_T = rw_gw_d$  où  $w_g$  et  $w_d$  sont les lectures préfixes respectives de  $T_g$  et de  $T_d$ .

**Q22.** Expliciter  $w_T$  pour l'arbre binaire  $T$  représenté ci-dessous :



**Figure 4 -** Représentation de  $T$  de Q22

**Q23.** Écrire une fonction récursive en Ocaml de signature :

```
prefixe : char arbre -> char list
```

et telle que `prefixe t` est le mot qui correspond à la lecture préfixe de l'arbre `t`.

**Q24.** Soit  $T$  un **ABR**, soit  $w_T$  la lecture préfixe de  $T$ . Montrer que  $T$  est l'**ABR** associé à  $w_T$ .

### III.2 - Une relation d'équivalence sur les mots

#### Définition 19 (S-adjacence)

Soient  $u$  et  $v$  deux mots sur  $\Sigma$ . On dit que  $u$  est **S-adjacent** à  $v$  (ou que  $u$  et  $v$  sont S-adjacents) s'il existe trois lettres  $a < b \leq c$  de  $\Sigma$  et trois mots  $f_1, f_2, f_3$  sur  $\Sigma$  vérifiant :

$$(u = f_1bf_2acf_3 \text{ et } v = f_1bf_2caf_3) \text{ ou } (u = f_1bf_2caf_3 \text{ et } v = f_1bf_2acf_3).$$

#### Définition 20 (S-équivalence)

Soient  $u$  et  $v$  deux mots sur  $\Sigma$ . On dit que  $u$  est **S-équivalent** à  $v$  (ou que  $u$  et  $v$  sont S-équivalents) s'il existe  $n \in \mathbb{N}$ ,  $(w_0, w_1, \dots, w_n) \in (\Sigma^*)^{n+1}$  vérifiant :

$$u = w_0, v = w_n, \forall i \in \{0, 1, \dots, n-1\}, w_i \text{ est S-adjacent à } w_{i+1}.$$

**Q25.** Montrer que la relation "être S-équivalent à" est bien une relation d'équivalence sur  $\Sigma^*$ .

**Q26.** Soient  $a, b, c$  trois lettres de  $\Sigma$  vérifiant  $a < b \leq c$ . Montrer que pour tout arbre binaire de recherche  $T$  contenant au moins la lettre  $b$ , on a :  $T \leftarrow ac = T \leftarrow ca$ . On pourra raisonner par récurrence sur le nombre de nœuds de  $T$ .

**Q27.** Montrer que si deux mots  $u$  et  $v$  sont S-équivalents, alors les **ABR** ( $\circ \leftarrow u$ ) et ( $\circ \leftarrow v$ ) sont égaux.

**Q28.** Soit  $w$  un mot de longueur  $n \geq 1$  sur  $\Sigma$ , soit  $r$  la première lettre de  $w$ . On veut montrer qu'il existe un mot  $w' = ruv$  où  $u$  (respectivement  $v$ ) est un mot dont toutes les lettres sont plus petites strictement (respectivement plus grandes au sens large) que  $r$  et tels que  $w$  et  $w'$  sont S-équivalents. On note  $A$  l'ensemble des mots S-équivalents à  $w$ .

Pour tout  $a = a_0 a_1 \dots a_{n-1} \in A$ , on pose :

$$N(a) = \{(i, j) \in \llbracket 1, n-1 \rrbracket^2 \mid i < j, a_j < r \leq a_i\}.$$

- (a) Justifier que l'ensemble  $A$  est fini.
- (b) Justifier que tous les mots de  $A$  commencent par  $r$ .
- (c) Soit  $a \in A$ . Montrer que si  $N(a) \neq \emptyset$ , alors il existe  $k \in \llbracket 1, n-2 \rrbracket$  tel que  $(k, k+1) \in N(a)$ .
- (d) Soit  $a \in A$ . Montrer que si  $N(a) = \emptyset$ , alors il existe  $u$  (respectivement  $v$ ) un mot dont toutes les lettres sont plus petites strictes (respectivement plus grandes ou égales) que  $r$  et tels que  $a = ruv$ .
- (e) Soit  $a$  un élément de  $A$  tel que le nombre d'éléments de  $N(a)$  soit le plus petit possible. Montrer par l'absurde que  $N(a) = \emptyset$  et en déduire qu'il existe  $u, v$  vérifiant les conditions de la question tels que  $ruv$  et  $w$  sont S-équivalents.

**Q29.** Montrer que tout mot  $w$  est S-équivalent à  $w_T$ , où  $w_T$  est la lecture préfixe de  $T = (\circ \leftarrow w)$ . On pourra raisonner par récurrence sur la longueur du mot  $w$  et exploiter les résultats de la **Q28**.

**Q30.** En déduire que deux mots sont S-équivalents si et seulement si ils sont des éléments d'une même classe sylvestre.

### III.3 - Construction de classes sylvestres

Pour construire des classes sylvestres, il est utile d'utiliser le produit de mélange.

#### Définition 21 (Mélange)

Soient  $u$  et  $v$  deux mots sur  $\Sigma$ . Le **mélange** de  $u$  et  $v$ , noté  $u \sqcup v$ , est l'ensemble récursivement défini comme suit :

- si  $v = \varepsilon$ , alors  $u \sqcup v = \{u\}$ ;
- si  $u = \varepsilon$ , alors  $u \sqcup v = \{v\}$ ;
- si  $u = au'$  et  $v = bv'$  où  $a$  et  $b$  sont des lettres,  $u'$  et  $v'$  des mots, alors :

$$u \sqcup v = \{aw \mid \exists w \in u' \sqcup v\} \cup \{bw \mid \exists w \in u \sqcup v'\}.$$

#### Définition 22 (Mélange de langages)

Soient  $L$  et  $M$  deux langages. Le **mélange des langages**  $L$  et  $M$ , noté  $L \sqcup M$ , est défini par :

$$L \sqcup M = \bigcup_{u \in L, v \in M} u \sqcup v.$$

Si au moins un des deux langages est égal à l'ensemble  $\emptyset$  ( $\neq \{\varepsilon\}$ ), alors on a  $L \sqcup M = \emptyset$ .

Étant donné un **ABR**  $T = (T_g, r, T_d)$ , on peut montrer que :

$$\text{Syl}(T) = \{rw \mid \exists w \in \text{Syl}(T_g) \sqcup \text{Syl}(T_d)\}.$$

On admet ce résultat pour la suite de cette sous-partie. On note que  $\text{Syl}(\circ) = \{\varepsilon\}$ .

**Q31.** Expliciter sans justification tous les éléments de l'ensemble  $abba \sqcup\sqcup ba$ .

**Q32.** Écrire une fonction récursive en Ocaml de signature :

```
ajout_lettre : char -> char list list -> char list list
```

et telle que `ajout_lettre a liste` est la liste de mots de la forme `a::w` où `w` est un élément de `liste`.

**Q33.** Écrire une fonction récursive en Ocaml de signature :

```
shuffle : char list -> char list -> char list list
```

et telle que `shuffle u v` est une liste contenant exactement tous les mots de  $u \sqcup\sqcup v$  avec répétitions possibles d'un même mot. On devra utiliser la fonction auxiliaire `ajout_lettre` et l'opérateur de concaténation `@`.

**Q34.** Écrire une fonction récursive en Ocaml de signature :

```
shuffle_l : char list list -> char list list -> char list list
```

et telle que `shuffle_l l m` est une liste contenant exactement tous les mots de  $L \sqcup\sqcup M$  où la liste `l` (respectivement `m`) représente le langage  $L$  (respectivement  $M$ ), avec répétitions possibles d'un même mot. Seules les fonctions définies précédemment peuvent être utilisées en fonctions auxiliaires.

**Q35.** Écrire une fonction récursive en Ocaml de signature :

```
sylvestre_T : char arbre -> char list list
```

et telle que `sylvestre_T t` est une liste contenant exactement tous les éléments de la classe `sylvestre` de `t`. Seules les fonctions définies précédemment peuvent être utilisées en fonctions auxiliaires.

**FIN**



**EBE NSI 1****SESSION 2021**

**CAPES  
CONCOURS EXTERNE  
TROISIÈME CONCOURS  
ET CAFEP CORRESPONDANTS**

**Sections :****NUMERIQUE ET SCIENCES INFORMATIQUES**

**PREMIÈRE ÉPREUVE D'ADMISSIBILITÉ**

Durée : 5 heures

*L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique est rigoureusement interdit.*

*Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.*

**NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.**

**Tournez la page S.V.P.**

**A**

**INFORMATION AUX CANDIDATS**

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► Concours externe du CAPES de l'enseignement public :

Concours  
**E<sub>BE</sub>**

Section/option  
**6900E**

Epreuve  
**101**

Matière  
**0540**

► Concours externe du CAFEP/CAPES de l'enseignement privé :

Concours  
**E<sub>BF</sub>**

Section/option  
**6900E**

Epreuve  
**101**

Matière  
**0540**

► Troisième concours du CAPES de l'enseignement public :

Concours  
**E<sub>BV</sub>**

Section/option  
**6900E**

Epreuve  
**101**

Matière  
**0540**

► Troisième concours CAFEP/CAPES de l'enseignement privé :

Concours  
**E<sub>BW</sub>**

Section/option  
**6900E**

Epreuve  
**101**

Matière  
**0540**





Cette épreuve est constituée de deux problèmes indépendants.

Pour ce sujet, vous pourrez utiliser les fonctions de manipulation de listes ou de matrices suivantes :

- Crédation d'une liste de taille  $n$  remplie avec la valeur  $x$  : `li = [x] * n`.
- Obtention de la taille d'une liste `li` : `len(li)`.
- Si `li` est une liste de  $n$  éléments, on peut accéder au  $k^e$  élément (pour  $0 \leq k < \text{len}(li)$ ) avec `li[k]`. On peut définir sa valeur avec `li[k] = x`.
- Un élément `x` peut être ajouté dans une liste `li` à l'aide de `li.append(x)`. On considérera qu'il s'agit d'une opération élémentaire.
- Les matrices sont des listes de listes, chaque sous-liste étant considérée comme une ligne de la matrice. Si `mat` est une matrice, elle possède `len(mat)` lignes et `len(mat[0])` colonnes.
- Crédation d'une matrice de  $n$  lignes et  $p$  colonnes, dont toutes les cases contiennent  $x$  :  
`mat = [[x for j in range(p)] for i in range(n)]`.
- On accède à (resp. modifie) l'élément de `mat` dans la  $i^e$  ligne et  $j^e$  colonne avec `mat[i][j]` (resp. `mat[i][j] = x`).

À moins de les redéfinir explicitement, l'utilisation de toute autre fonction sur les listes (`sort`, `index`, `max`, etc.) est interdite. On rappelle enfin qu'une fonction qui s'arrête sans avoir rencontré l'instruction `return` renvoie `None`.

## Problème 1 : Points proches dans le plan

Ce problème, pouvant par exemple survenir dans le domaine de la navigation maritime, vise à déterminer, dans un nuage de points du plan, la paire de points les plus proches. Il est constitué de trois parties dépendantes.

Formellement, on suppose qu'on dispose de  $n$  points dans le plan  $(M_0, M_1, \dots, M_{n-1})$  dans un ordre quelconque pour le moment. Ils seront représentés en Python par deux listes de flottants de taille  $n$  : `coords_x` et `coords_y`, donnant respectivement les abscisses et les ordonnées des points. On dira ainsi que  $M_i$  est le point d'indice  $i$ , qu'il a pour abscisse  $x_i = \text{coords}_x[i]$  et pour ordonnée  $y_i = \text{coords}_y[i]$ . On supposera que `coords_x` et `coords_y` sont des variables globales, qu'on ne modifiera jamais au cours de l'exécution de l'algorithme.

### 1 Approche exhaustive

On utilise la distance euclidienne définie par  $d(M_i, M_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ .

► **Question 1** Écrire une fonction `distance(i, j)` qui renvoie la distance entre les points  $M_i$  et  $M_j$ . On utilisera la fonction `sqrt` après l'avoir importée.

► **Question 2** Rappeler sommairement comment sont stockés les flottants en mémoire. Quelle conséquence cela peut-il avoir sur le calcul de la distance ? On ignorera par la suite les problèmes d'approximation.

► **Question 3** Écrire une fonction `plus_proche()` qui renvoie, à l'aide d'une recherche exhaustive, le couple d'entiers des indices  $i$  et  $j$  des deux points les plus proches du nuage de points.

► **Question 4** Donner, en la justifiant sommairement, la complexité de la fonction précédente en fonction de  $n$ .

### 2 Quelques outils pour s'améliorer

On souhaite maintenant obtenir la distance entre les deux points les plus proches avec une meilleure complexité. Pour cela nous allons décrire un algorithme utilisant une méthode de type *diviser pour régner*. Cette partie introduit des fonctions utiles pour la mise en œuvre de cet algorithme.

On se donne la fonction suivante :

```
def tri(liste):
    n = len(liste)
    for i in range(n):
        pos = i
        while pos > 0 and liste[pos] < liste[pos-1]:
            liste[pos], liste[pos-1] = liste[pos-1], liste[pos]
            pos -= 1
```

► **Question 5** Que renvoie cette fonction ? Que fait-elle ? Le démontrer soigneusement en exhibant un invariant de boucle.

► **Question 6** Donner, en la démontrant, la complexité de la fonction `tri` en fonction de la taille de la liste donnée en paramètre.

► **Question 7** On souhaite trier une liste contenant des indices de points suivant l'ordre des abscisses croissantes. Que faudrait-il changer à la fonction `tri` ci-dessus pour qu'elle réalise cette opération ?

► **Question 8** Indiquer le nom d'un autre algorithme de tri plus efficace dans le pire des cas, ainsi que sa complexité. On ne demande pas de le programmer.

On admettra que l'on dispose de deux listes de  $n$  entiers `liste_x` (resp. `liste_y`) contenant les indices des points du nuage triés par abscisses croissantes (resp. par ordonnées croissantes). On supposera désormais que deux points quelconques ont des abscisses et des ordonnées distinctes.

Dans toute la suite, un sous-ensemble de points sera décrit par un *cluster*. Un cluster est une matrice de deux lignes contenant chacune les mêmes numéros correspondant aux numéros des points dans le sous-ensemble considéré. Dans la première ligne, les points sont triés par abscisses croissantes ; dans la seconde, ils sont triés par ordonnées croissantes. La figure 1 donne la représentation de deux clusters.

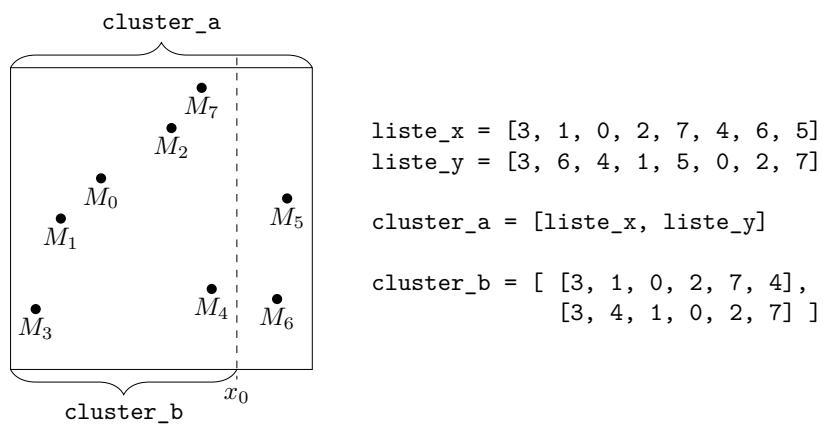


FIGURE 1 – Représentation en Python de deux clusters

Pour être efficace, notre algorithme ne doit pas re-trier les listes des indices de points à chaque étape. Nous allons donc définir une fonction qui permet d'extraire des indices d'un cluster et former ainsi un nouveau cluster plus petit.

► **Question 9** Écrire une fonction `sous_cluster(cl, x_min, x_max)` qui prend en arguments un cluster `cl` et deux flottants `x_min` et `x_max`, et renvoie le sous-cluster des points dont l'abscisse est comprise entre `x_min` et `x_max` (au sens large). Cette fonction doit avoir une complexité linéaire *en la taille du cluster*.

► **Question 10** Écrire une fonction `mediane(cl)` qui prend en entrée un cluster `cl` contenant au moins 2 points et renvoie une abscisse médiane, c'est-à-dire que la moitié (au moins) des points a une abscisse inférieure ou égale à cette valeur, et la moitié (au moins) des points a une abscisse supérieure ou égale à cette valeur. Cette fonction doit avoir une complexité en  $O(1)$ .

### 3 Méthode sophistiquée

Le fonctionnement de l'algorithme utilisant une méthode de type *diviser pour régner* est illustré par la figure 2 :

1. Si le cluster contient deux ou trois points, on calcule la distance minimale en calculant toutes les distances possibles.
2. Sinon, on sépare le cluster en deux parties  $G$  et  $D$  qu'on supposera de tailles égales (éventuellement à un point près) suivant la médiane des abscisses, qu'on notera  $x_0$ .
3. Les deux points les plus proches sont soit tous les deux dans  $G$ , soit tous les deux dans  $D$ , soit un dans  $G$  et un dans  $D$ .
4. On calcule récursivement le couple le plus proche dans  $G$  et le couple le plus proche dans  $D$ . On note  $d_0$  la plus petite des deux distances obtenues.
5. On cherche s'il existe une paire de points  $(M_1, M_2)$  telle que  $M_1$  est dans  $G$ ,  $M_2$  dans  $D$ , et  $d(M_1, M_2) < d_0$ .
6. Si on en trouve une (ou plusieurs), on renvoie la plus petite de ces distances. Sinon, on renvoie  $d_0$ .

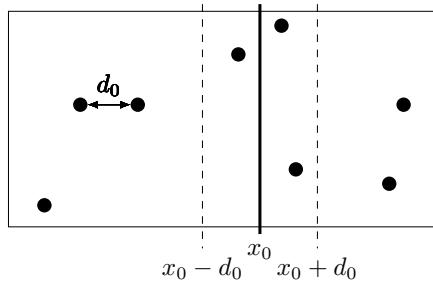


FIGURE 2 – Illustration du diviser pour régner

► **Question 11** Écrire une fonction `gauche(c1)` qui prend en argument un cluster `c1` contenant au moins deux points et renvoie le cluster constitué uniquement de la moitié (éventuellement arrondie à l'entier supérieur) des points les plus à gauche du cluster `c1`.

On suppose qu'on dispose d'une fonction `droite(c1)` qui renvoie le cluster contenant tous les autres points du cluster `c1` n'appartenant pas au cluster renvoyé par la fonction `gauche(c1)`.

► **Question 12** Justifier que l'on peut se contenter de chercher les points  $M_1$  et  $M_2$  de l'étape 5 de l'algorithme dans l'ensemble des points dont l'abscisse appartient à  $I_0 = [x_0 - d_0, x_0 + d_0]$ .

► **Question 13** Écrire une fonction `bande_centrale(c1, d0)` qui prend en argument un cluster `c1` et un réel `d0`, et renvoie le cluster des points dont l'abscisse est dans  $I_0$ . Cette fonction doit avoir une complexité linéaire en la taille du cluster.

► **Question 14** Montrer que deux points  $M_1$  et  $M_2$  (de l'étape 5 de l'algorithme) situés à une distance inférieure à  $d_0$  se trouvent, dans la deuxième ligne du cluster (c'est-à-dire la ligne triée par ordonnées croissantes), séparés d'au plus 6 éléments.

On pourra montrer par l'absurde qu'un rectangle, à préciser, de dimensions  $2d_0 \times d_0$  contient au plus 8 points.

► **Question 15** En déduire une fonction `fusion(c1, d0)` qui prend en entrée un cluster de points dont toutes les abscisses sont dans un intervalle  $[x_0 - d_0, x_0 + d_0]$ , et renvoie la distance minimale entre deux points du cluster si elle est inférieure à  $d_0$ , ou  $d_0$  sinon. Cette fonction doit avoir une complexité linéaire en la taille du cluster `c1`. Vous justifierez cette complexité.

► **Question 16** Écrire une fonction récursive `distance_minimale(c1)` qui prend en argument un cluster et utilise l'algorithme décrit plus haut pour renvoyer la distance minimale entre deux points du cluster.

- **Question 17** Si on note  $n$  la taille du cluster `c1`, et  $C(n)$  le nombre d'opérations élémentaires réalisées par la fonction `distance_minimale(c1)`, justifier que l'on a :

$$C(n) = 2C(n/2) + O(n)$$

- **Question 18** En déduire, en la démontrant, la complexité  $C(n)$ . On pourra se limiter au cas où  $n$  est une puissance de 2.

## Problème 2 : Composantes connexes et biconnexes

### 4 Site Internet et bases de données

On s'intéresse dans cette partie à un site Internet d'échange de supports de cours entre enseignant · e · s de NSI. Chaque personne désirant proposer ou récupérer du contenu doit commencer par se créer un compte sur ce site et peut ensuite accéder à du contenu ou en proposer.

- **Question 19** Expliquer sommairement la différence entre Internet et le web.

- **Question 20** Expliquer deux conséquences du règlement général sur la protection des données (RGPD) sur le site Internet.

Ce site repose sur une base de données contenant en particulier trois tables.

- La table `comptes` possède un enregistrement par utilisateur ou utilisatrice, et ses attributs sont :
  - `id`, un identifiant numérique, unique pour chaque compte ;
  - `nom`, le nom de la personne possédant le compte ;
  - et d'autres informations, concernant le mot de passe, l'adresse mail, des préférences sur le site, etc., que nous ne détaillons pas ici.
- La table `ressources` possède un enregistrement par document téléversé sur le site. Ses attributs sont :
  - `id`, un identifiant numérique, unique pour chaque ressource ;
  - `owner`, l'identifiant de la personne ayant créé la ressource ;
  - `titre`, une chaîne de caractères décrivant la ressource ;
  - `type`, chaîne de caractères pouvant être `cours`, `ds`, `tp` ou `td`.
- La table `chargement` mémorise chaque fois qu'un utilisateur télécharge une ressource sur le site. Ses attributs sont :
  - `date`, date du téléchargement, par exemple '2021-02-28' pour le 28 février 2021 (on peut utiliser des opérations de comparaison classiques avec ce format) ;
  - `id_u`, identifiant de l'utilisateur qui télécharge la ressource ;
  - `id_r`, identifiant de la ressource téléchargée.

Voici un extrait de chacune de ces tables :

comptes			ressources			
<code>id</code>	<code>nom</code>	...	<code>id</code>	<code>owner</code>	<code>titre</code>	<code>type</code>
1	Ada Lovelace	...	4	1	Machine à décalage	<code>cours</code>
4	Alan Turing	...	13	4	Intelligence artificielle	<code>td</code>
...	...	...	...	...	...	...

chargement		
<code>date</code>	<code>id_u</code>	<code>id_r</code>
'1931-06-29'	4	4
'2020-05-30'	27	458
...	...	...

► **Question 21** Écrire une requête SQL permettant de connaître le nombre total de ressources de type cours présentes sur le site.

► **Question 22** Que fait la requête suivante :

```
SELECT ressource.titre, comptes.nom
FROM chargement
JOIN ressources ON ressources.id = chargement.id_r
JOIN comptes ON comptes.id = chargement.id_u
ORDER BY chargement.date DESC
LIMIT 1
```

► **Question 23** Écrire une requête SQL qui permet de déterminer la liste des triplets  $(x, y, n)$ , signifiant que la personne possédant l'identifiant  $x$  a téléchargé  $n$  fois des documents téléversés par la personne possédant l'identifiant  $y$ .

On définit le graphe non-orienté  $G(V, E)$  où  $V$  est l'ensemble des identifiants de comptes sur le site et  $E \subset V \times V$  l'ensemble des paires d'identifiants telles que le premier compte a déjà téléchargé des documents téléversés par l'autre et réciproquement. Ainsi, si  $(x, y) \in E$ , alors on doit avoir  $(y, x) \in E$ .

► **Question 24** Écrire une requête SQL qui renvoie la table des couples  $(x, y)$  de  $E$ .

## 5 Composantes connexes

L'objectif de cette partie est de déterminer les composantes connexes du graphe  $G$  défini à la partie précédente. Dans toute la suite, on notera  $|X|$  le cardinal d'un ensemble  $X$ . On supposera que l'ensemble  $V$  est constitué de sommets numérotés par des entiers consécutifs commençant à 0, c'est-à-dire que  $V = \{0, 1, \dots, |V| - 1\}$ . On dira que deux sommets  $x, y$  de  $V$  sont *voisins* lorsque  $(x, y) \in E$ .

La requête de la question 24 permet de récupérer le résultat sous forme d'une liste de tuple à deux valeurs. On souhaite avoir plutôt une représentation par listes d'adjacences, à savoir une liste de  $|V|$  sous-listes, la  $i^{\text{e}}$  sous-liste contenant les voisins du sommet  $i$ . On illustre ces différentes représentations avec le graphe  $G_{\text{ex}}$  de la figure 3.

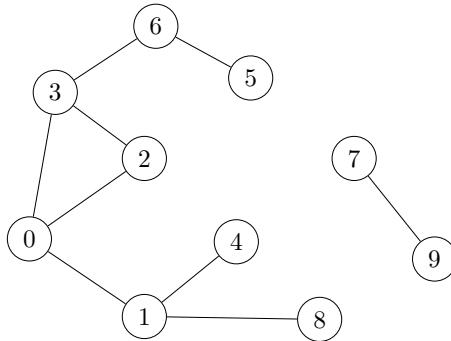


FIGURE 3 – Un graphe  $G_{\text{ex}}$

Ce graphe serait obtenu à la question 24 sous la forme :

```
g_ex_a = [
    (0, 1), (0, 2), (0, 3), (1, 0), (1, 4), (1, 8),
    (2, 0), (2, 3), (3, 0), (3, 2), (3, 6),
    (4, 1), (5, 6), (6, 3), (6, 5),
    (7, 9), (8, 1), (9, 7)
]
```

Sa représentation par listes d'adjacences serait :

```
g_ex_b = [ [1, 2, 3], [0, 4, 8], [0, 3],
           [0, 2, 6], [1], [6], [3, 5] [9], [1], [7]
         ]
```

► **Question 25** Écrire une fonction `adjacences(n, li)` qui prend en argument un entier  $n$  correspondant à  $|V|$  et `li`, une liste de couples correspondant à un ensemble  $E$  (comme par exemple `g_ex_a`) dans un ordre quelconque, et renvoyant la représentation du graphe  $G(V, E)$  sous forme de listes d'adjacences (comme par exemple `g_ex_b`).

On se donne le programme suivant :

```
class Arbre():
    def __init__(self, sommet):
        self.sommet = sommet
        self.children = []

    def add_child(self, child):
        self.children.append(child)

def parcours(listes_adjacences):
    n = len(listes_adjacences)
    deja_vu = [False] * n

    def explorer(i):
        arbre = Arbre(i)
        voisins = listes_adjacences[i]
        for s in voisins:
            if not deja_vu[s]:
                deja_vu[s] = True
                arbre.add_child(explorer(s))
        return arbre

    res = []
    for i in range(n):
        if not deja_vu[i]:
            deja_vu[i] = True
            res.append(explorer(i))
    return res
```

► **Question 26** Quel est le type de la valeur renvoyée par la fonction `parcours`? Appliquer à la main cette fonction sur la liste d'adjacence `g_ex_b` du graphe  $G_{\text{ex}}$  de la figure 3, et représenter la valeur de retour de cette fonction. Quel est le nom de ce parcours ?

► **Question 27** Montrer que la complexité de la fonction `parcours` est en  $O(|V| + |E|)$ . Dans toute la suite, on dira qu'un algorithme ayant cette complexité est *linéaire*.

► **Question 28** Rappeler la définition de la connexité d'un graphe.

► **Question 29** Écrire une fonction `connexe(listes_adjacences)` qui renvoie `True` si le graphe décrit par les listes d'adjacences `listes_adjacences` est connexe et `False` sinon.

► **Question 30** Écrire une fonction `composantes_connexes(p_graphe)` prenant en argument `p_graphe` le graphe obtenu avec la fonction `parcours` et renvoie les composantes connexes sous forme de liste de listes de sommets.

► **Question 31** Quelle est la limitation liée au fait que la fonction `explorer`, programmée en Python, est récursive ?

Dans toute la suite, lorsqu'une fonction est demandée, on pourra utiliser ou non une fonction récursive, au choix des candidat.e.s.

## 6 Graphes biconnexes

On suppose dans cette partie que  $G$  est un graphe connexe. Si  $\forall i \in [0; k] x_i \in V$ , on appelle *chaîne* une suite finie  $(x_0, x_1, \dots, x_k)$  telle que pour tout  $i$ , on ait  $(x_i, x_{i+1}) \in E$ . Cette chaîne est un *circuit élémentaire* lorsque  $x_0 = x_k$ , et c'est de plus un *circuit élémentaire* si tous les sommets  $x_0, \dots, x_{k-1}$  sont distincts deux à deux. On dit que  $G(V, E)$  est *biconnexe* lorsque :

- $|V| = 1$  ;
- $|V| = 2$ ,  $V = \{a, b\}$  et  $(a, b) \in E$  ;
- ou  $|V| \geq 3$  et pour toute paire  $(x, y) \in V^2$ , il existe un circuit élémentaire contenant  $x$  et  $y$ .

► **Question 32** Montrer qu'un graphe biconnexe est également connexe.

► **Question 33** Donner un exemple de graphe connexe mais pas biconnexe.

On dit qu'un sommet  $x$  de  $G$  est un *point d'articulation* lorsque le graphe  $G$  privé du sommet  $x$  (et des arêtes issues de  $x$ ) n'est plus connexe. Notre objectif dans cette partie est de montrer la propriété suivante si  $G(V, E)$  possède au moins 3 sommets :

$G(V, E)$  est biconnexe si et seulement si  $G$  ne possède pas de point d'articulation.

► **Question 34** Sur le graphe  $G'_{\text{ex}}$  de la figure 4, donner les points d'articulations.

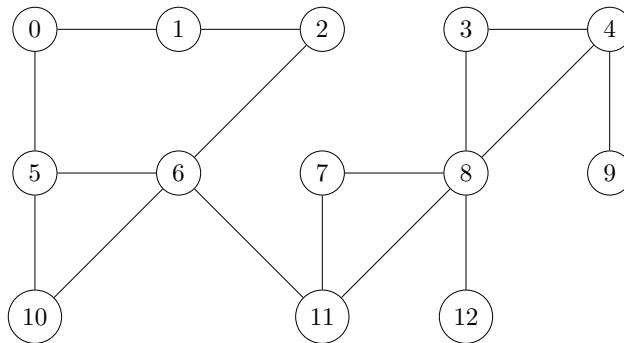


FIGURE 4 – Le graphe connexe  $G'_{\text{ex}}$

Dans toute la suite, on considère un graphe  $G$  ayant au moins 3 sommets.

► **Question 35** Soit  $G(V, E)$  possédant un point d'articulation. Montrer que  $G$  n'est pas biconnexe.

► **Question 36** Inversement, supposons  $G(V, E)$  un graphe sans point d'articulation, et tel que  $|V| \geq 3$ . Considérons deux sommets  $x$  et  $y$ .

1. Justifier qu'il existe une chaîne  $(x_0 = x, x_1, \dots, x_k = y)$  dans le graphe.
2. Montrer qu'il existe un circuit élémentaire contenant  $x_0$  et  $x_1$ .
3. Pour  $i \geq 1$ , on suppose qu'il existe un circuit élémentaire  $C$  contenant  $x_0$  et  $x_i$ . Montrer qu'il existe alors un circuit élémentaire contenant  $x_0$  et  $x_{i+1}$ . On pourra distinguer deux cas selon que  $C$  contient ou non  $x_{i+1}$ .
4. En déduire que  $G$  est biconnexe.

► **Question 37** Expliquer comment on peut déterminer si un sommet particulier est un point d'articulation à l'aide d'un parcours en profondeur.

► **Question 38** En déduire un algorithme qui prend en entrée un graphe connexe décrit par ses listes d'adjacences, et détermine si ce graphe est biconnexe en utilisant la propriété précédente. On ne demande pas de programmer cet algorithme en Python. Quelle serait sa complexité en fonction des caractéristiques  $|E|$  et  $|V|$  du graphe ?

## 7 Algorithme efficace pour déterminer les points d'articulation

Dans cette partie, on détaille comment déterminer tous les points d'articulation d'un graphe  $G(V, E)$  avec une complexité linéaire.

On modifie le programme `parcours` pour lui faire remplir et renvoyer une liste `prefixe` correspondant aux ordres d'appel de la fonction `explorer`. De plus, on suppose désormais que `listes_adjacences` décrit un graphe connexe, et on ne renverra qu'un seul arbre, issu de l'exploration à partir du sommet 0, et la liste `prefixe`. On utilise `nonlocal` pour que la variable `count` soit définie pour toute la fonction `parcours` et pas uniquement au sein de la fonction `explorer`.

```
def parcours(listes_adjacences):
    n = len(listes_adjacences)
    deja_vu = [False] * n
    prefixe = [-1] * n
    count = 1

    def explorer(i):
        nonlocal count
        prefixe[i] = count
        count += 1
        arbre = Arbre(i)
        voisins = listes_adjacences[i]
        for s in voisins:
            if not deja_vu[s]:
                deja_vu[s] = True
                arbre.add_child(explorer(s))
        return arbre

    deja_vu[0] = True
    return explorer(0), prefixe
```

► **Question 39** Donner les valeurs de la liste `prefixe` renvoyée par le programme ci-dessus si on l'applique sur le graphe  $G'_{\text{ex}}$  de la figure 4. On supposera que les voisins sont rangés par ordre croissant de leur numéro dans les listes d'adjacences.

► **Question 40** Soit  $G$  un graphe connexe dans lequel on réalise le parcours avec la fonction ci-dessus, et soit  $(i, j)$  une arête de  $G$  telle que `prefixe[i] < prefixe[j]`. Montrer que  $j$  est un descendant de  $i$  dans l'arbre.

Pour chaque sommet  $i$ , on note  $\mathcal{V}(i)$  le voisinage de  $i$ , c'est-à-dire l'ensemble constitué de  $i$  et de ses voisins. Par extension, pour tout  $A \subset V$ , on notera  $\mathcal{V}(A) = \cup_{i \in A} \mathcal{V}(i)$ . En supposant réalisé un parcours dans l'arbre, on notera de plus  $\mathcal{D}(i)$  l'ensemble des descendants de  $i$  dans l'arbre. Enfin, on définit `ord[i]` par :

$$\text{ord}[i] = \min_{j \in \mathcal{V}(\mathcal{D}(i))} \text{prefixe}[j]$$

► **Question 41** Sur le graphe  $G'_{\text{ex}}$  de la figure 4, donner pour chaque sommet les valeurs de `ord[i]`, en se basant sur les valeurs obtenues à la question 39.

On admettra qu'un sommet  $i$  du graphe qui n'est pas la racine est un point d'articulation si et seulement si un de ses fils  $j$  vérifie `ord[j] = prefixe[i]`.

On supposera de plus qu'on dispose d'une fonction `calcule_ord(listes_adjacences)` qui renvoie la liste des `ord[i]` du graphe décrit par `listes_adjacences`, avec une complexité linéaire.

► **Question 42** Écrire une fonction `points_articulation(listes_adjacences)` qui renvoie la liste des points d'articulation d'un graphe. On fera attention à traiter la racine de l'arbre comme un cas particulier.

On définit une composante biconnexe d'un graphe  $G$  comme un sous-ensemble de sommets maximal (au sens de l'inclusion) qui est biconnexe.

► **Question 43** Sur le graphe  $G'_{\text{ex}}$  de la figure 4, donner la liste des composantes biconnexes.

► **Question 44** Décrire un algorithme qui renvoie les composantes biconnexes d'un graphe avec une complexité linéaire. On ne demande pas de programmer cet algorithme.

**EBE NSI 2****SESSION 2021****CAPES  
CONCOURS EXTERNE  
ET CAFEP CORRESPONDANTS****Section : NUMERIQUE ET SCIENCES INFORMATIQUES****SECONDE ÉPREUVE D'ADMISSIBILITÉ**

Durée : 5 heures

*L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique est rigoureusement interdit.*

*Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.*

**NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.**

Tournez la page S.V.P.

(A)

**INFORMATION AUX CANDIDATS**

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

**► Concours externe du CAPES de l'enseignement public :**

Concours	Section/option	Epreuve	Matière
<b>E B E</b>	<b>6900E</b>	<b>109</b>	<b>0530</b>

**► Concours externe du CAFEP/CAPES de l'enseignement privé :**

Concours	Section/option	Epreuve	Matière
<b>E B F</b>	<b>6900E</b>	<b>109</b>	<b>0530</b>





Préambule : Ce sujet comporte trois parties indépendantes. Pour toutes les questions pédagogiques, vous pourrez vous appuyer sur les programmes de SNT ainsi que de première et de terminale NSI dont des extraits choisis sont donnés en Annexe.

## **Partie 1 - Réseaux de communication**

Les notions portant sur l'Internet et les réseaux de communication font partie des programmes de l'enseignement de SNT en seconde et de NSI en première et terminale. En tant qu'enseignant ou enseignante dans ces matières, vous devrez aborder et expliquer les différentes couches de la pile protocolaire TCP/IP. Dans cet exercice, vous allez explorer certaines de ces couches.

### **1 Généralités sur les réseaux**

1. L'Internet est basé sur une architecture en couches. Les différentes couches de cette architecture constituent la pile protocolaire utilisée par l'Internet. Expliquer pourquoi ce fonctionnement en couches a été retenu. Préciser les éventuels avantages et inconvénients d'un tel système.
2. Que définit un protocole réseau ?
3. Expliquer ce qu'est le modèle client-serveur.
4. Citer deux technologies de communication et donner un ordre de grandeur de leur capacité d'émission (appelé aussi débit d'émission) pour chacune d'entre elles.
5. Vous trouverez un exemple de réseau dans le document 1. Associer à chacun des numéros légendés sur ce schéma l'un des termes suivants : Wi-Fi, Ethernet, Client, Serveur, Fibre optique, Routeur, Commutateur (aussi appelée Switch). Sur votre copie, vous pourrez ne reporter que les numéros et les termes correspondants. Il n'est pas nécessaire de reproduire le schéma.
6. Le programme de première de la spécialité NSI demande de présenter "le rôle des différents constituants du réseau local de l'établissement" (document 5). Inventorier les principaux éléments susceptibles de constituer un réseau local et préciser leur(s) rôle(s).

### **2 Couche transport**

7. Quel est l'objectif d'un protocole de transport ?
8. Expliquer le principe général de fonctionnement du protocole de transport TCP.
9. Comparer le protocole de transport UDP et le protocole de transport TCP. Donner les avantages et les inconvénients de chacun de ces protocoles.
10. Donner un exemple d'application reposant sur le protocole de transport UDP et un exemple d'application reposant sur le protocole de transport TCP.
11. Le protocole de transport TCP est au programme de SNT (document 2). Proposer une activité débranchée permettant de faire comprendre aux élèves les notions de fiabilité d'une transmission et d'absence de garantie temporelle caractérisant le protocole TCP. L'activité proposée devra mettre en évidence les problèmes induits par cette absence de garantie temporelle.

### **3 Couche réseau**

12. Quel est l'objectif d'un protocole de la couche réseau ?

#### **3.1 Adressage IP**

13. Décrire une activité que vous proposez à vos élèves de seconde en SNT pour expliquer le but d'une adresse IP et comment elle est organisée.
14. Expliquer le rôle du protocole DHCP.

### 3.2 Routage

#### Protocoles de routage

15. Le routage est au programme de la spécialité NSI terminale (document 7). Proposer un plan de cours sur ce sujet en 10 lignes maximum, puis décrire une activité que vous réaliserez avec vos élèves.
16. Expliquer le principe du protocole OSPF. Quel algorithme de parcours est utilisé par ce protocole ? Illustrer le fonctionnement de cet algorithme sur un schéma réseau que vous choisirez.

La recherche d'un plus court chemin sur un graphe est une problématique sous-jacente à celle du routage de données. Dans la section suivante, on s'intéresse à l'enseignement de notions relatives aux graphes (programme de la classe de SNT, document 3) et à leur parcours (programme de la classe de terminale NSI, document 10).

#### Utilisation des graphes

17. Proposer un exercice destiné à des élèves de seconde permettant de travailler les notions de rayon, diamètre et centre d'un graphe. Vous rédigerez une correction de cet exercice.
18. Le document 9 est un exercice proposé à des élèves de terminale en spécialité NSI.
  - (a) Proposer une correction et un barème de cet exercice en justifiant votre barème.
  - (b) Relever les erreurs éventuelles de la copie donnée dans le document 11.
  - (c) Proposer des améliorations à cet exercice afin, notamment, de le rendre plus accessible pour un élève ayant un niveau "moyen" en spécialité NSI de terminale.

## 4 Étude du fonctionnement général et des performances d'un réseau

Cette partie concerne le programme de première de la spécialité NSI, donné dans le document 5.

19. Dans ce programme, il est demandé de "simuler ou mettre en œuvre un réseau". Proposer une séquence pédagogique où les élèves auront à mettre en œuvre un réseau à l'aide d'un logiciel de simulation. Vous préciserez le logiciel que vous comptez utiliser. Proposer un schéma du réseau qui serait simulé et décrire les activités que vous proposeriez sur ce scénario réseau.
20. Discuter les avantages et les inconvénients de la simulation pour étudier le fonctionnement ou les performances d'un réseau.
21. Le programme demande de "mettre en évidence l'intérêt du découpage des données en paquets et de leur encapsulation". Citer les facteurs qui ont un impact sur le temps de transfert d'un message de la source à la destination. À partir de ces facteurs, expliquer l'intérêt d'un découpage des données à transmettre en paquets. À quoi peut conduire un découpage excessif (à savoir en un très grand nombre de paquets) ?

## Partie 2 - Le Web

Cette partie porte sur le World Wide Web qui sera abrégé en Web dans la suite de l'énoncé.

## 5 Généralités, langages et URL

Le Web fait partie du programme de SNT comme indiqué dans le document 4.

22. Donner une définition du Web.
23. À quelle date et par qui le Web a-t-il été "inventé" ?
24. Proposer une séquence pédagogique permettant d'introduire le HTML et le CSS en classe de SNT.
25. À la suite de votre enseignement sur HTML et CSS, vous comptez évaluer vos élèves via un QCM. Proposer un QCM comportant 3 questions ayant chacune 4 choix possibles et un seul choix correct. On justifiera le choix des questions et des réponses proposées, en précisant la réponse juste.
26. Expliquer la structure générale d'une URL.

## 6 HTTP et HTTPS

27. À quoi sert le protocole HTTP ?
28. Expliquer le rôle des méthodes GET et POST du protocole HTTP.
29. Proposer une séquence pédagogique permettant d'aborder le protocole HTTP avec des élèves de la spécialité NSI de première (document 6).
30. La sécurisation des communications est au programme de terminale NSI (document 7). Expliquer les principes :
  - du chiffrement symétrique,
  - du chiffrement asymétrique,
  - du protocole HTTPS (on s'intéressera uniquement à la partie "sécurité" de ce protocole).
31. Vous désirez mettre en place une activité permettant d'illustrer le principe de chiffrement symétrique auprès de vos élèves de spécialité terminale NSI. Décrire l'activité que vous allez réaliser et motiver les choix que vous avez faits.

## 7 Moteur de recherche

L'étude des moteurs de recherche est au programme de SNT (document 4).

32. À quoi sert un moteur de recherche ?
33. Expliquer le principe général de fonctionnement de l'algorithme "PageRank" qui est à la base du moteur de recherche de Google.
34. Vous demandez à trois élèves de seconde de réaliser un exposé sur les moteurs recherches. Proposer une grille d'évaluation pour cet exposé.

## Partie 3 - Développement d'applications

Une partie de l'enseignement de NSI est consacrée à l'élaboration de projets conduits par les élèves, comme cela est précisé dans le document 8. On s'intéresse, dans la suite, à deux projets proposés par deux groupes d'élèves de terminale NSI.

## 8 Logiciel d'emprunt de livres

Un groupe d'élève souhaite réaliser un logiciel d'emprunt de livres. Le cahier des charges établi est le suivant :

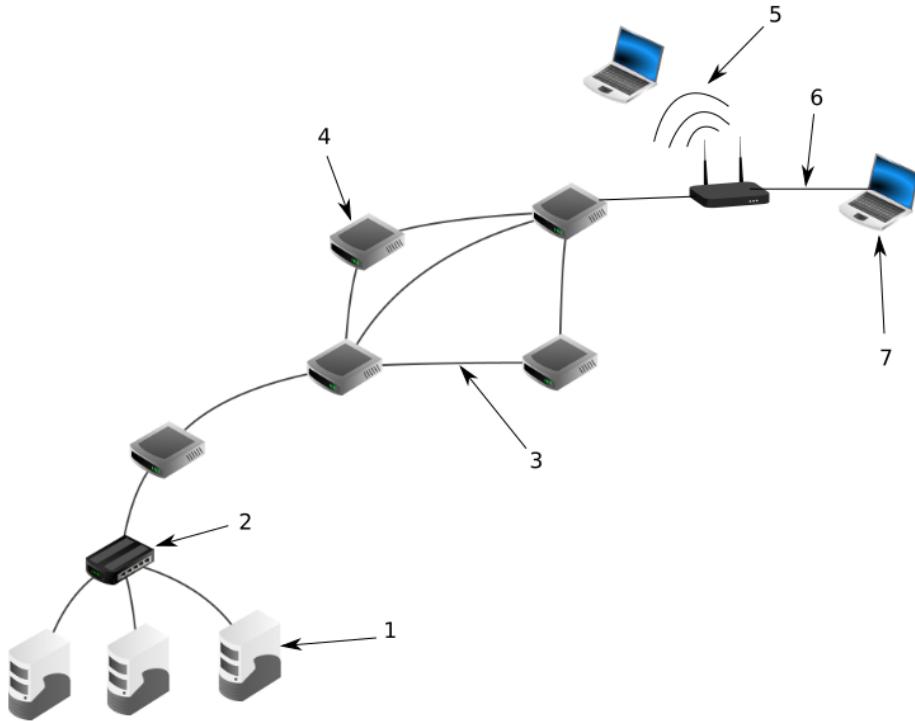
- Les informations relatives aux utilisateurs et aux livres disponibles à l'emprunt doivent être structurées dans une base de données.
  - Le modèle conceptuel de données retenu est celui fourni dans le document 13.
  - Le logiciel en lui-même doit être muni d'une interface graphique.
35. Justifier le choix d'une structure de base de données dans le cadre de ce projet.
  36. Proposer un langage permettant d'implémenter une interface graphique. Recommanderiez-vous l'usage d'une bibliothèque de ce langage à cette fin ? Si oui, laquelle ?
  37. Un élève du groupe choisit de se consacrer à la structuration de la base de donnée et implémente les trois tables fournies dans le document 14. De quoi cet élève n'a-t-il pas tenu compte ? Corriger sa proposition.
  38. Présenter le concept de clé étrangère tel que vous le feriez à des élèves de terminale NSI.
  39. Proposer trois situations que les élèves de ce groupe pourront rencontrer lors de la réalisation de leur projet qui amènent à réaliser :
    - (a) Une requête de sélection de données sur la table "livres".
    - (b) Une requête d'insertion de données sur la table "livres".
    - (c) Une requête de mise à jour de données portant sur une jointure des tables "livres" et "emprunts".Fournir également ces requêtes en langage SQL.
  40. Proposer une grille d'évaluation pour ce projet.

## 9 Jeu du morpion

Un autre groupe d'élèves souhaite réaliser un jeu de morpion. C'est un jeu entre deux joueurs dans lequel chaque joueur ou joueuse inscrit, l'un après l'autre, son symbole sur une grille. Le premier joueur ou la première joueuse qui parvient à aligner trois de ses symboles horizontalement, verticalement ou en diagonale gagne le jeu. Le cahier des charges établi avec les élèves est le suivant :

- Le jeu doit être muni d'une interface graphique qui doit être programmée en langage Python.
  - Le jeu est conçu de telle sorte que le joueur se confronte à l'ordinateur qui jouerait de manière optimale, en se reposant sur l'algorithme min-max décrit dans le document 15.
41. Proposer une activité qui vous permettrait de présenter aux élèves d'une classe de terminale NSI les concepts d'arbre binaire, de parcours en profondeur et en largeur de cet arbre. On pourra se référer à l'extrait du programme fourni dans le document 12.
  42. Expliquer le fonctionnement de l'algorithme min-max comme vous le feriez auprès des élèves du groupe ayant choisi ce projet.
  43. Préciser pourquoi il répond à la problématique d'un jeu optimal de la part de l'ordinateur, et pourquoi il se fait en temps de calcul raisonnable dans le cas du jeu de morpion.
  44. Proposer trois tests permettant de vérifier le bon fonctionnement de l'application réalisée par les élèves au cours de ce projet.
  45. Proposer une grille d'évaluation pour ce projet.

## Annexes exercice 1



Document 1 – Schéma réseau

Contenus	Capacités attendues
Protocole TCP/IP : paquets, routage des paquets	Distinguer le rôle des protocoles IP et TCP. Caractériser les principes du routage et ses limites. Distinguer la fiabilité de transmission et l'absence de garantie temporelle.
Adresses symboliques et serveurs DNS	Sur des exemples réels, retrouver une adresse IP à partir d'une adresse symbolique et inversement.
Réseaux pair-à-pair	Décrire l'intérêt des réseaux pair-à-pair ainsi que les usages illicites qu'on peut en faire.
Indépendance d'internet par rapport au réseau physique	Caractériser quelques types de réseaux physiques : obsolètes ou actuels, rapides ou lents, filaires ou non. Caractériser l'ordre de grandeur du trafic de données sur internet et son évolution.

Document 2 – Extrait programme de SNT (thème "Internet")

Contenus	Capacités attendues
Réseaux sociaux existants	Distinguer plusieurs réseaux sociaux selon leurs caractéristiques, y compris un ordre de grandeur de leurs nombres d'abonnés. Paramétrier des abonnements pour assurer la confidentialité de données personnelles.
Modèle économique des réseaux sociaux	Identifier les sources de revenus des entreprises de réseautage social.
Rayon, diamètre et centre d'un graphe	Déterminer ces caractéristiques sur des graphes simples.
Notion de « petit monde » Expérience de Milgram	Décrire comment l'information présentée par les réseaux sociaux est conditionnée par le choix préalable de ses amis.
Harcèlement numérique	Connaître les dispositions de l'article 222-33-2-2 du code pénal.

Document 3 – Extrait programme de SNT (thème "Les réseaux sociaux")

Contenus	Capacités attendues
Repères historiques	Définir les étapes du développement du Web.
Hypertexte	Maîtriser les renvois d'un texte à différents contenus.
Langages HTML et CSS	Distinguer ce qui relève du contenu d'une page et de son style de présentation. Étudier et modifier une page HTML simple.
URL	Décomposer l'URL d'une page. Reconnaitre les pages sécurisées.
Requête HTTP	Décomposer le contenu d'une requête HTTP et identifier les paramètres passés.
Modèle client/serveur	Inspecter le code d'une page hébergée par un serveur et distinguer ce qui est exécuté par le client et par le serveur.
Moteurs de recherche : principes et usages	Mener une analyse critique des résultats fournis par un moteur de recherche. Comprendre que toute requête laisse des traces.
Paramètres de sécurité d'un navigateur	Maîtriser les réglages les plus importants concernant la gestion des cookies, la sécurité et la confidentialité d'un navigateur.

Document 4 – Extrait programme de SNT (thème "Le Web")

Contenus	Capacités attendues	Commentaires
Modèle d'architecture séquentielle (von Neumann)	Distinguer les rôles et les caractéristiques des différents constituants d'une machine. Dérouler l'exécution d'une séquence d'instructions simples du type langage machine.	La présentation se limite aux concepts généraux. On distingue les architectures monoprocesseur et les architectures multiprocesseur. Des activités débranchées sont proposées. Les circuits combinatoires réalisent des fonctions booléennes.
Transmission de données dans un réseau Protocoles de communication Architecture d'un réseau	Mettre en évidence l'intérêt du découpage des données en paquets et de leur encapsulation. Dérouler le fonctionnement d'un protocole simple de récupération de perte de paquets (bit alterné). Simuler ou mettre en œuvre un réseau.	Le protocole peut être expliqué et simulé en mode débranché. Le lien est fait avec ce qui a été vu en classe de seconde sur le protocole TCP/IP. Le rôle des différents constituants du réseau local de l'établissement est présenté.

Document 5 – Extrait du programme de première de la spécialité NSI (thème "Architectures matérielles et systèmes d'exploitation")

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine  Événements	Identifier les différents composants graphiques permettant d'interagir avec une application Web.  Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.
Interaction avec l'utilisateur dans une page Web	Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page Web.	
Interaction client-serveur.  Requêtes HTTP, réponses du serveur	Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre.  Distinguer ce qui est mémorisé dans le client et retransmis au serveur.  Reconnaitre quand et pourquoi la transmission est chiffrée.	Il s'agit de faire le lien avec ce qui a été vu en classe de seconde et d'expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP.
Formulaire d'une page Web	Analyser le fonctionnement d'un formulaire simple.  Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.

Document 6 – Extrait du programme de première de la spécialité NSI (thème "Interactions entre l'homme et la machine sur le Web")

Contenus	Capacités attendues	Commentaires
Composants intégrés d'un système sur puce.	Identifier les principaux composants sur un schéma de circuit et les avantages de leur intégration en termes de vitesse et de consommation.	Le circuit d'un téléphone peut être pris comme un exemple : microprocesseurs, mémoires locales, interfaces radio et filaires, gestion d'énergie, contrôleur vidéo, accélérateur graphique, réseaux sur puce, etc.
Gestion des processus et des ressources par un système d'exploitation.	Décrire la création d'un processus, l'ordonnancement de plusieurs processus par le système.  Mettre en évidence le risque de l'interblocage ( <i>deadlock</i> ).	À l'aide d'outils standard, il s'agit d'observer les processus actifs ou en attente sur une machine.  Une présentation débranchée de l'interblocage peut être proposée.
Protocoles de routage.	Identifier, suivant le protocole de routage utilisé, la route empruntée par un paquet.	En mode débranché, les tables de routage étant données, on se réfère au nombre de sauts (protocole RIP) ou au coût des routes (protocole OSPF).  Le lien avec les algorithmes de recherche de chemin sur un graphe est mis en évidence.
Sécurisation des communications.	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique).  Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier.  La négociation de la méthode chiffrement du protocole SSL ( <i>Secure Sockets Layer</i> ) n'est pas abordée.

Document 7 – Extrait programme de terminale NSI (thème "Architectures matérielles, systèmes d'exploitation et réseaux")

### Démarche de projet

Un enseignement d'informatique ne saurait se réduire à une présentation de concepts ou de méthodes sans permettre aux élèves de se les approprier en développant des projets.

Un quart au moins de l'horaire total de la spécialité est réservé à la conception et à l'élaboration de projets conduits par les élèves.

Les projets réalisés par les élèves, sous la conduite du professeur, constituent un apprentissage fondamental tant pour l'appropriation des concepts informatiques que pour l'acquisition de compétences. En classe de première comme en classe terminale, ils peuvent porter sur des problématiques issues d'autres disciplines et ont essentiellement pour but d'imaginer des solutions répondant à un problème ; dans la mesure du possible, il convient de laisser le choix du thème du projet aux élèves. Il peut s'agir d'un approfondissement théorique des concepts étudiés en commun, d'une application à d'autres disciplines telle qu'une simulation d'expérience, d'exploitation de modules liés à l'intelligence artificielle et en particulier à l'apprentissage automatique, d'un travail sur des données socioéconomiques, du développement d'un logiciel de lexicographie, d'un projet autour d'un objet connecté ou d'un robot, de la conception d'une bibliothèque implémentant une structure de données complexe, d'un problème de traitement d'image ou de son, d'une application mobile, par exemple de réalité virtuelle ou augmentée, du développement d'un site Web associé à l'utilisation d'une base de données, de la réalisation d'un interpréteur d'un mini-langage, de la recherche d'itinéraire sur une carte (algorithme A\*), d'un programme de jeu de stratégie, etc.

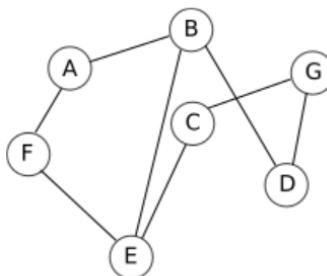
La conduite d'un projet inclut des points d'étape pour faire un bilan avec le professeur, valider des éléments, contrôler l'avancement du projet ou en adapter les objectifs, voire le redéfinir partiellement, afin de maintenir la motivation des élèves.

Les professeurs veillent à ce que les projets restent d'une ambition raisonnable afin de leur permettre d'aboutir.

### Document 8 – Extrait programme de terminale NSI

#### Exercice III (5 points)

Soit le graphe suivant :



1. Donnez une implémentation de ce graphe en langage Python.
2. En partant du sommet A, donnez la liste des sommets parcourus dans le cas d'un parcours « en profondeur d'abord » (attention à l'ordre des sommets).
3. En partant du sommet A, donnez la liste des sommets parcourus dans le cas d'un parcours « en largeur d'abord » (attention à l'ordre des sommets).
4. Écrivez l'algorithme permettant de réaliser un parcours « en largeur d'abord »

### Document 9 – Exercice proposé à des élèves de terminale en spécialité NSI

Contenus	Capacités attendues	Commentaires
Algorithmes sur les arbres binaires et sur les arbres binaires de recherche.	Calculer la taille et la hauteur d'un arbre. Parcourir un arbre de différentes façons (ordres infixé, préfixé ou suffixé ; ordre en largeur d'abord). Rechercher une clé dans un arbre de recherche, insérer une clé.	Une structure de données récursive adaptée est utilisée. L'exemple des arbres permet d'illustrer la programmation par classe. La recherche dans un arbre de recherche équilibré est de coût logarithmique.
Algorithmes sur les graphes.	Parcourir un graphe en profondeur d'abord, en largeur d'abord. Repérer la présence d'un cycle dans un graphe. Chercher un chemin dans un graphe.	Le parcours d'un labyrinthe et le routage dans Internet sont des exemples d'algorithme sur les graphes. L'exemple des graphes permet d'illustrer l'utilisation des classes en programmation.

### Document 10 – Extrait programme de terminale NSI (thème "Algorithmique")

Exercice III

- 1) On peut utiliser un dictionnaire pour implémenter le graphe en python.

$$g = \{ 'A': ['F', 'B'], 'B': ['E', 'D'], 'C': ['E', 'G'], 'D': ['G'], 'E': ['F'] \}$$

- 2) Parcours en profondeur d'abord:

$$A \rightarrow F \rightarrow E \rightarrow C \rightarrow G \rightarrow D \rightarrow B$$

- 3) Parcours en largeur d'abord

$$A \rightarrow B \rightarrow E \rightarrow F \rightarrow D \rightarrow G \rightarrow C$$

4)

## VARIABLE

$G$ : graphe

$s$ : noeud origine

$u$ : noeud

$v$ : noeud

$p$ : pile (initiallement vide.)

## DEBUT

$s.couleur \leftarrow \text{noir}$

$\text{piler}(s, p)$

tant que  $p$  non vide :

$u \leftarrow \text{depiler}(p)$

pour chaque sommet  $v$  adj à  $u$ :

si  $v.couleur$  n'est pas noir:

$v.couleur \leftarrow \text{noir}$

$\text{piler}(v, p)$

fin si

fin pour

fin tant que

Document 11 – Extrait d'une copie d'élève

### Structures de données

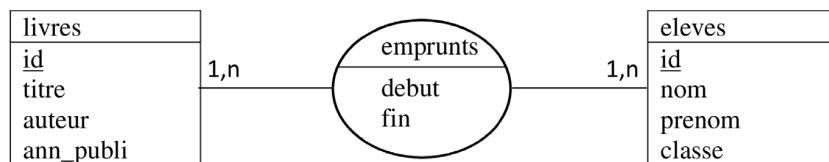
L'écriture sur des exemples simples de plusieurs implémentations d'une même structure de données permet de faire émerger les notions d'interface et d'implémentation, ou encore de structure de données abstraite.

Le paradigme de la programmation objet peut être utilisé pour réaliser des implémentations effectives des structures de données, même si ce n'est pas la seule façon de procéder.

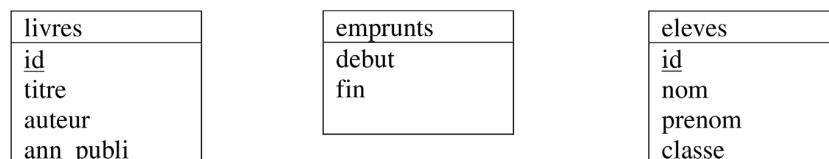
Le lien est établi avec la notion de modularité qui figure dans la rubrique « langages et programmation » en mettant en évidence l'intérêt d'utiliser des bibliothèques ou des API (*Application Programming Interface*).

Contenus	Capacités attendues	Commentaires
Structures de données, interface et implémentation.	Spécifier une structure de données par son interface. Distinguer interface et implémentation. Écrire plusieurs implémentations d'une même structure de données.	L'abstraction des structures de données est introduite après plusieurs implémentations d'une structure simple comme la file (avec un tableau ou avec deux piles).
Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.	Écrire la définition d'une classe. Accéder aux attributs et méthodes d'une classe.	On n'aborde pas ici tous les aspects de la programmation objet comme le polymorphisme et l'héritage.
Listes, piles, files : structures linéaires. Dictionnaires, index et clé.	Distinguer des structures par le jeu des méthodes qui les caractérisent. Choisir une structure de données adaptée à la situation à modéliser. Distinguer la recherche d'une valeur dans une liste et dans un dictionnaire.	On distingue les modes FIFO ( <i>first in first out</i> ) et LIFO ( <i>last in first out</i> ) des piles et des files.
Arbres : structures hiérarchiques. Arbres binaires : noeuds, racines, feuilles, sous-arbres gauches, sous-arbres droits.	Identifier des situations nécessitant une structure de données arborescente. Évaluer quelques mesures des arbres binaires (taille, encadrement de la hauteur, etc.).	On fait le lien avec la rubrique « algorithmique ».
Graphes : structures relationnelles. Sommets, arcs, arêtes, graphes orientés ou non orientés.	Modéliser des situations sous forme de graphes. Écrire les implémentations correspondantes d'un graphe : matrice d'adjacence, liste de successeurs/de prédécesseurs. Passer d'une représentation à une autre.	On s'appuie sur des exemples comme le réseau routier, le réseau électrique, Internet, les réseaux sociaux. Le choix de la représentation dépend du traitement qu'on veut mettre en place : on fait le lien avec la rubrique « algorithmique ».

Document 12 – Extrait du programme de terminale de la spécialité NSI (thème "Structures de données")



Document 13 – Modèle conceptuel de données retenu



Document 14 – Tables implémentées par un élève

Lorsque l'IA doit jouer dans une situation de jeu donnée, son action est déterminée par le choix d'une nouvelle situation, prise au hasard, parmi toutes les situations de jeu atteignables dont l'évaluation par la fonction **min-max**, fournie ci-dessous est maximale. Dans cette fonction, on considérera que :

- La fonction **évaluer** retourne un entier 1 = gagnant, 0 = match nul ou -1 = perdant selon que la situation de fin de jeu fournie en argument est favorable, ne départage pas ou est défavorable au joueur fourni en argument.
- La fonction **coefficient** retourne un entier 1 = IA ou -1 = humain, selon le joueur fourni en argument.
- La fonction **situations-atteignables** retourne, pour une situation de jeu fournie en argument, l'ensemble des situations pouvant être atteintes au tour de jeu suivant.
- Les fonctions **min** et **max** retournent les minimum et maximum des ensembles de valeurs qui leur sont fournies en argument.

```
1. fonction min-max(situation, joueur)
2.   si situation est finale
3.     retourner évaluer(situation, joueur) × coefficient(joueur)
4.   sinon
5.     situationsSuivantes ← situations-atteignables(situation)
6.     si joueur = IA
7.       retourner max{min-max(suivante, humain) pour suivante ∈ situationsSuivantes}
8.     sinon
9.       retourner min{min-max(suivante, IA) pour suivante ∈ situationsSuivantes}
10.    fin si
11.  fin si
12. fin fonction
```

Document 15 – Principe général du fonctionnement de l'algorithme du min-max

**ECOLE POLYTECHNIQUE - ESPCI  
ECOLES NORMALES SUPERIEURES**

**CONCOURS D'ADMISSION 2021**

**JEUDI 15 AVRIL 2021  
16h30 - 18h30**

**FILIERES MP-PC-PSI**

**Epreuve n° 8**

**INFORMATIQUE B (XELCR)**

*Durée : 2 heures*

*L'utilisation des calculatrices n'est pas  
autorisée pour cette épreuve*

## Gestion d'un allocateur dynamique de mémoire

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve. Le langage de programmation sera **obligatoirement Python**.

**Complexité.** La complexité, ou le temps d'exécution, d'une fonction  $P$  est le nombre d'opérations élémentaires (addition, multiplication, affectation, test, etc...) nécessaires à l'exécution de  $P$ . Lorsqu'il est demandé de donner la complexité d'un programme, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

**Rappels concernant le langage Python.** *L'utilisation de toute fonction Python sur les listes autre que celles mentionnées dans ce paragraphe est interdite.*

Si `a` désigne une liste en Python :

- `len(a)` renvoie la longueur de cette liste, c'est-à-dire le nombre d'éléments qu'elle contient ; la complexité de `len` est en  $\mathcal{O}(1)$ .
- `a[i]` désigne le  $i$ -ème élément de la liste, où l'indice  $i$  est compris entre 0 et `len(a) - 1` ; la complexité de cette opération est en  $\mathcal{O}(1)$ .

On pourra aussi utiliser la fonction `range` pour constituer une liste d'entiers et réaliser des itérations.

**Introduction.** Dans ce sujet, on s'intéresse à un environnement de programmation qui a une mémoire limitée, et où l'on veut limiter au maximum la création de nouvelles structures (comme les listes Python) en mémoire. Pour cela, nous souhaitons fournir au programmeur un environnement de gestion dynamique de la mémoire dans lequel il pourra *réservier* des *portions* de mémoire pour y lire et écrire des données, puis *libérer* certaines portions quand il n'en a plus besoin. Pour simplifier, nous ne permettons au programmeur de lire et écrire dans ces portions de mémoire que des valeurs de type caractère, c'est-à-dire des chaînes de caractères de longueur 1. Une portion qui n'est pas (ou n'est plus) réservée est dite *libre*. Chaque portion agit comme un tableau de caractères : elle a une taille, et on peut lire et écrire les caractères aux positions comprises entre 0 et  $t - 1$ , si  $t$  est la taille demandée pour la portion considérée lors de sa réservation.

Ce service est réduit aux cinq fonctions suivantes :

- `p = reserver(n, c)` qui renvoie une nouvelle portion `p` réservée qui était libre précédemment. La portion est de taille `n` et toutes ses `n` cases contiennent le même caractère `c`. La fonction renvoie `None` si la mémoire est trop pleine pour permettre cette réservation.

- `liberer(p)` qui libère une portion `p` qui était réservée précédemment.
- `c = lire(p, i)` renvoie le caractère `c` stocké à la case en position `i` dans la portion `p`.
- `ecrire(p, i, c)` qui met à jour la case en position `i` dans la portion `p` avec le caractère `c`.
- `demarrage()` qui est appelée une seule fois pour initialiser correctement la mémoire, avant tout appel aux quatre fonctions précédentes.

Dans ce sujet, on considère que le programmeur fera un usage licite de ces fonctions en respectant les préconditions suivantes :

- il ne lira et n'écrira qu'en utilisant les fonctions `lire` et `ecrire`, dans les portions actuellement réservées et à des positions comprises entre 0 et  $t - 1$ , si  $t$  est la taille demandée pour la portion considérée lors de sa réservation ;
- il n'utilisera les fonctions `libere`, `lire` et `ecrire` qu'avec des portions `p` obtenues par un appel à `reserver` et qui n'auront pas été explicitement libérées depuis cet appel ;
- il n'utilisera `p = reserver(n, c)` que lorsque l'entier `n` est strictement positif ;
- il n'utilisera les fonctions `libere`, `lire`, `ecrire` et `reserver` que sur une mémoire correctement initialisée avec un appel à `demarrage()`.

Ces propriétés sont appellées *hypothèses de bon usage*.

Pour implémenter ces services, nous créons initialement une liste Python `mem` de taille `TAILLE_MEM` (initialisée avec des 0). Les variables `mem` et `TAILLE_MEM` sont des variables globales. **Aucune autre liste Python ne doit être manipulée dans ce sujet**, exception faite des listes créées avec la fonction `range` pour réaliser des itérations.

```
mem = [0] * TAILLE_MEM
```

On suppose que `mem` vient juste d'être initialisée de cette façon au moment de l'appel de `demarrage()`. Cette liste va contenir les différents contenus des portions qui seront réservées et libérées. Chaque case contiendra soit un caractère placé par le programmeur, soit un entier que nous placerons pour organiser notre structure de données. Grâce aux hypothèses de bon usage, le programmeur n'accédera jamais à des cases contenant des entiers.

Dans tout le sujet, les services de lecture et écriture sont donnés par les fonctions suivantes :

```
def lire(p, i):
    return mem[p+i]

def ecrire(p, i, c):
    mem[p+i] = c
```

Par conséquent, nous désignerons par une portion un indice valide `p` de `mem`, tel que `mem[p] ... mem[p+n-1]` sont les cases réservées par le programmeur lors de l'appel à

`p = reserver(n, c)`. Nous confondons donc la portion `p` avec la première position, accessible au programmeur, des cases ainsi réservées.

Ce sujet est conçu pour être traité linéairement. Les différentes hypothèses et spécifications listées dans cette introduction sont valables et importantes pour les quatre parties du sujet. Chaque partie propose une stratégie différente des fonctions `demarrage`, `reserver` et `liberer` de gestion dynamique de la mémoire. Seule la fonction `initialiser` décrite plus bas est la même dans chaque partie.

## Partie I : Implémentation naïve

Dans cette partie, nous organisons notre mémoire `mem` comme une suite contiguë de portions entre les indices 1 et `TAILLE_MEM-1`. La case `mem[0]` joue un rôle spécifique : elle contient un entier `prochain` tel que les portions réservées jusqu'à présent se trouvent parmi les cases `mem[1] ... mem[prochain-1]`. Lors de la prochaine réservation, la nouvelle portion sera placée à partir de l'indice `prochain`. Ici, une portion `p` de taille `n` est constituée de `n` cases `mem[p], ..., mem[p+n-1]`. Mais dans cette stratégie naïve d'implémentation, la libération de portion n'a pas d'effet sur `mem`, ce qui est correct mais peu efficace en termes de consommation mémoire : aucun *recyclage* de mémoire n'est possible. La fonction de libération est donc simplement<sup>1</sup> :

```
def liberer(p):
    pass
```

La FIGURE 1 présente le contenu du début de `mem` après l'appel des services suivants par le programmeur :

```
p1 = reserver(6, 'a')
p2 = reserver(9, 'b')
p3 = reserver(3, 'c')
ecrire(p1, 0, 'A')
```

19	A	a	a	a	a	a	b	b	b	b	b	b	b	c	c	c	0	0	0	0	0	0	0	0	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

FIGURE 1 – Exemple de contenu de la mémoire - Stratégie naïve.

**Question 1.** Écrire une fonction `initialiser(p, n, c)` qui initialise une portion de taille `n` à l'indice `p` en remplissant chaque case avec le caractère `c`.

**Question 2.** Écrire la fonction `demarrage()` pour cette stratégie.

**Question 3.** Écrire la fonction `reserver(n, c)` pour cette stratégie. Préciser la complexité de `reserver(n, c)` en fonction de `n` et `TAILLE_MEM`.

1. En Python, `pass` est l'instruction qui ne fait rien.

À partir de maintenant, les seules fonctions manipulant *directement* la liste `mem` seront fournies par l'énoncé. Vos solutions doivent donc s'appuyer sur les fonctions auxiliaires qui vous seront fournies et ne pas contenir de lecture (de la forme `a=mem[p]`) ou écriture (de la forme `mem[p]=a`) directes à `mem`.

## Partie II : Réservations de blocs de tailles fixes

Nous cherchons maintenant à permettre la réutilisation de la mémoire libérée. Nous proposons pour cela une nouvelle stratégie d'implémentation. Nous fixons une constante globale `TAILLE_BLOC` et nous placerons les portions à l'intérieur de blocs de taille fixe de `TAILLE_BLOC` cases contiguës dans la liste `mem`. Une portion `p` réservée avec la taille  $n$  (où  $n+1 \leq TAILLE\_BLOC$ ) occupe  $n+1$  cases au début d'un tel bloc. La première case `mem[p-1]` contient un *en-tête* qui vaut 1 si la portion est encore réservée, ou 0 si elle a été libérée. Les cases suivantes sont utilisées pour stocker les caractères écrits par le programmeur. Comme précédemment, la case `mem[0]` est réservée pour pointer sur la prochaine portion libre pour créer un bloc (si aucun recyclage de bloc libéré n'était possible).

La FIGURE 2 présente le contenu du début de `mem` après l'appel des services suivants par le programmeur :

```
p1 = reserver(6, 'a')
p2 = reserver(4, 'b')
p3 = reserver(3, 'c')
liberer(p2)
ecrire(p1, 0, 'A')
```

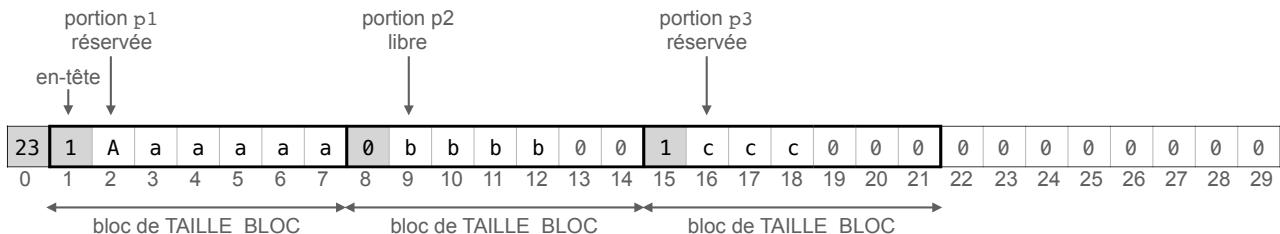


FIGURE 2 – Exemple de contenu de la mémoire avec `TAILLE_BLOC=7` - Partie II.

Pour manipuler les cases d'en-tête des portions et la prochaine portion libre, nous vous fournissons les fonctions suivantes :

<code>def lire_prochain():</code>	<code>def ecrire_prochain(p):</code>
<code>    return mem[0]</code>	<code>    mem[0] = p</code>
<code>def est_reservee(p):</code>	<code>def est_libre(p):</code>
<code>    return mem[p-1] == 1</code>	<code>    return mem[p-1] == 0</code>

```
def marque_reservee(p):
    mem[p-1] = 1
```

```
def marque_libre(p):
    mem[p-1] = 0
```

**Question 4.** Écrire la fonction `demarrage()` pour cette stratégie d'implémentation.

Pour réserver une nouvelle portion, on cherche en priorité à réutiliser un bloc laissé libre par une précédente libération. La portion libre pointée par `lire_prochain()` n'est utilisée que si un tel bloc n'existe pas.

**Question 5.** Écrire la fonction `reserver(n, c)` pour cette stratégie d'implémentation. Renvoyer `None` si la taille `n` est trop grande vis-à-vis de `TAILLE_BLOC`. Préciser la complexité de `reserver(n, c)` en fonction de `n` et `TAILLE_MEM`.

**Question 6.** Écrire la fonction `liberer(p)` pour cette stratégie d'implémentation.

### Partie III : Portions avec en-tête et pied de page

Nous abordons maintenant une autre stratégie d'implémentation qui permettra de ne pas limiter autant la taille de chaque portion. Nous munissons pour cela chaque portion d'un en-tête mais aussi d'un *pied de page*. Pour chaque portion, ces deux cases additionnelles contiennent la même valeur : un entier encodant deux informations sur la portion courante. La première information indique si la portion est réservée ou pas. La deuxième indique la taille réservée à cette portion. Nous imposons que cette taille soit toujours un entier pair. Si un programmeur demande à réserver une portion de taille `n` impaire, nous attribuerons une taille `n + 1` à cette portion, mais le programmeur n'est pas autorisé à consulter cet espace supplémentaire, en vertu des hypothèses de bon usage dictées en début de sujet.

Nous vous fournissons toutes les fonctions nécessitant un accès direct à `mem` :

```
def est_reservee(p):
    return mem[p-1] % 2 == 1
```

```
def est_libre(p):
    return mem[p-1] % 2 == 0
```

```
def marque_reservee(p, taille):
    mot = taille + 1
    mem[p-1] = mot
    mem[p+taille] = mot
```

```
def marque_libre(p, taille):
    mot = taille
    mem[p-1] = mot
    mem[p+taille] = mot
```

```
def lire_taille(p):
    return 2 * (mem[p-1] // 2)
```

```
def lire_taille_precedent(p):
    return 2 * (mem[p-2] // 2)
```

```
def precedent_est_libre(p):
    return mem[p-2] % 2 == 0
```

```
def precedent_est_reserve(p):
    return mem[p-2] % 2 == 1
```

**Question 7.** Expliquer en quelques lignes le fonctionnement des fonctions `est_reservee`, `marque_reservee`, `lire_taille` et `precedent_est_libre`.

Nous utiliserons deux portions spéciales, une *portion prologue* et une *portion épilogue*. Ces deux portions spéciales sont de taille nulle et toujours marquées réservées. Nous les plaçons en début et en fin de la zone de réservation. La portion prologue se situe à une position fixe donnée par la variable globale suivante :

```
PROLOGUE = 2
```

La case `mem[0]` indique la position courante de l'épilogue. L'épilogue est contigu au prologue au démarrage, puis est déplacé vers des indices plus élevés quand la zone des portions réservées doit être agrandie. Nous vous fournirons les fonctions permettant de lire et modifier les informations concernant cette portion spéciale :

```
def lire_position_epilogue():      def ecrire_position_epilogue(p):  
    return mem[0]                  mem[0] = p
```

La FIGURE 3 présente<sup>2</sup> le contenu du début de `mem` après l'appel des services suivants par le programmeur :

```
p1 = reserver(6, 'a')  
p2 = reserver(7, 'b')  
p3 = reserver(1, 'c')  
liberer(p2)  
ecrire(p1, 0, 'A')
```

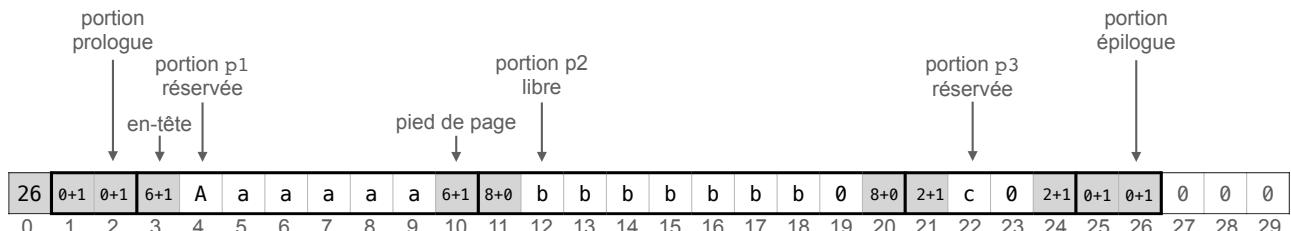


FIGURE 3 – Exemple de contenu de la mémoire - Partie III.

**Question 8.** Écrire la fonction `démarrage()` pour cette stratégie.

A part les deux portions spéciales épilogue et prologue, toutes les portions ont une taille strictement positive. Lors de la réservation d'une nouvelle portion, on réserve en priorité dans la zone de mémoire comprise entre le prologue et l'épilogue, et en dernier recours on déplace l'épilogue. Si une portion libre est suffisamment grande, une réservation dans cette zone la sépare en une portion réservée et une portion libre.

2. Dans les en-têtes et pieds de page de ces figures, la notation  $t + b$  désigne l'encodage d'une paire formée d'une taille  $t$  et d'un bit d'état de réservation  $b \in \{0, 1\}$ .

La FIGURE 4 illustre ce mécanisme en présentant le contenu de la mémoire de la FIGURE 3, après l'appel à `p4 = reserver(2, 'd')`.

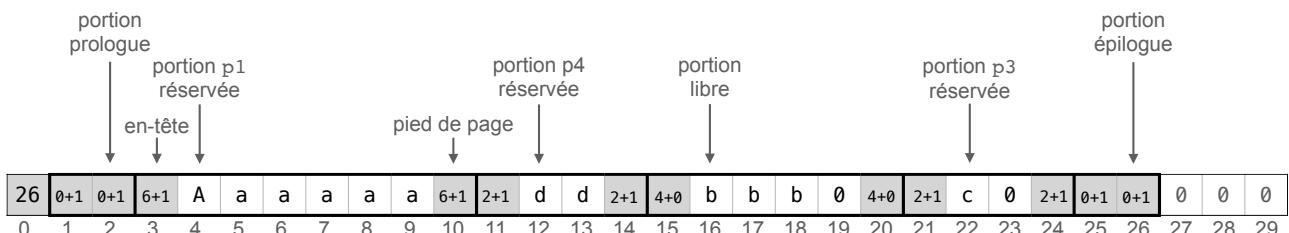


FIGURE 4 – Exemple de contenu de la mémoire après nouvelle réservation - Partie III.

**Question 9.** Écrire la fonction `reserver(n, c)` pour cette stratégie. Préciser la complexité de `reserver(n, c)` en fonction de `n` et `TAILLE_MEM`.

Lors de la libération d'une portion, on étudie les portions adjacentes libres et on réalise si possible une *fusion* afin qu'il n'y ait jamais deux portions adjacentes libres après un appel à la fonction `liberer`.

La FIGURE 5 illustre ce mécanisme en présentant le contenu de la mémoire de la FIGURE 4, après l'appel à `liberer(p3)`.

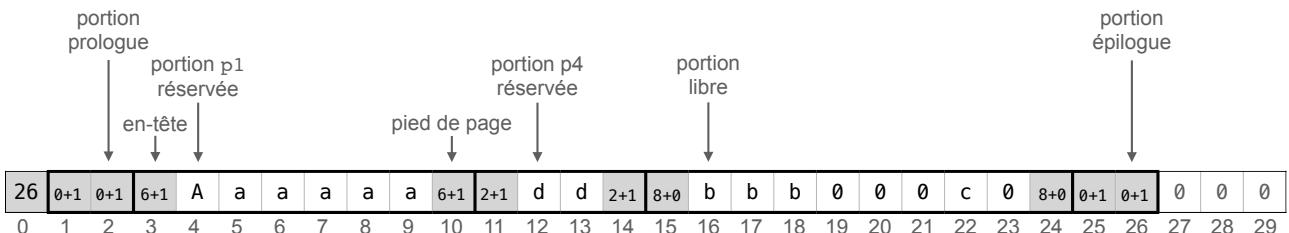


FIGURE 5 – Exemple de contenu de la mémoire après nouvelle libération - Partie III.

**Question 10.** Écrire la fonction `liberer(p)` pour cette stratégie. Justifier la correction et la complexité de `liberer(p)` en fonction de `TAILLE_MEM`, en précisant le nombre maximal de fusions effectuées à chaque appel, et en expliquant l'intérêt des portions prologues et épilogues.

## Partie IV : Chaînage explicite des portions libres

Nous souhaitons maintenant améliorer l'implémentation de la partie précédente pour accélérer la recherche de portions libérées. Nous allons pour cela maintenir une *chaîne des portions libres*. Il s'agit d'une séquence de portions libres organisée de la façon suivante :

- dans chaque portion libre `p` dans la chaîne, on stocke une information dans les cases `mem[p]` et `mem[p+1]` :

- la case `mem[p]` contient la position de la portion libre prédécesseur dans la chaîne ;
- la `mem[p+1]` contient la position de la portion libre successeur dans la chaîne ;
- la position de l'*entrée* de la chaîne est stockée dans la case `mem[1]`. Elle vaut 0 si et seulement si la chaîne est vide.
- si la chaîne n'est pas vide, son entrée est une portion dont le prédécesseur vaut 0 ;
- si la chaîne n'est pas vide, elle contient une portion *de fin* (éventuellement égale à celle d'entrée) dont le successeur vaut 0 ;
- toutes les autres portions de la chaîne ont des prédécesseurs et successeurs non nuls.

Pour manipuler cette structure, nous vous fournissons les fonctions suivantes :

```
def lire_entree_chaine():
    return mem[1]

def ecrire_entree_chaine(p):
    mem[1] = p

def lire_predecesseur(p):
    return lire(p, 0)

def lire_successeur(p):
    return lire(p, 1)

def ecrire_predecesseur(p, predecesseur):
    ecrire(p, 0, predecesseur)

def ecrire_successeur(p, successeur):
    ecrire(p, 1, successeur)

def chaine_est_vide():
    return lire_entree_chaine() == 0
```

Par ailleurs, on redéfinit la portion prologue comme suit :

```
PROLOGUE = 3
```

La FIGURE 6 présente le contenu du début de `mem` après l'appel des services suivants par le programmeur :

```
p1 = reserver(2, 'a')
p2 = reserver(2, 'b')
p3 = reserver(2, 'c')
p4 = reserver(2, 'd')
p5 = reserver(1, 'e')
p6 = reserver(1, 'f')
liberer(p1)
liberer(p5)
liberer(p3)
```

La FIGURE 7 présente la chaîne associée, qui commence par la portion  $p3=13$ , suivi de  $p5=21$  et enfin de  $p1=5$ .

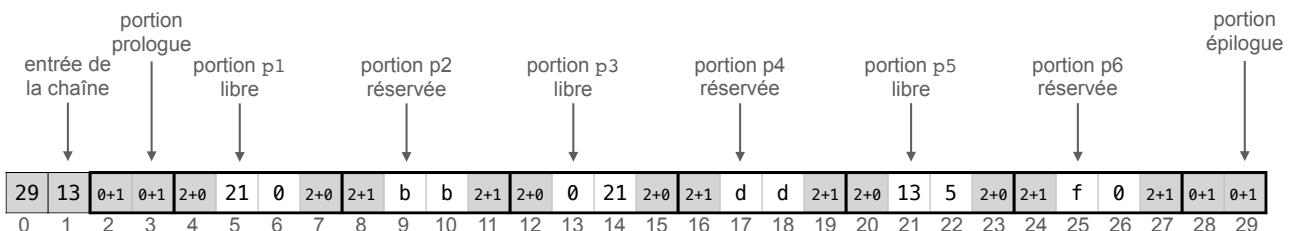


FIGURE 6 – Exemple de contenu de la mémoire - Partie IV.

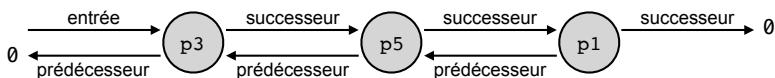


FIGURE 7 – Chaîne associée à la mémoire de la FIGURE 6.

**Question 11.** Écrire la fonction `ajoute_en_entree_de_chaine(p)` qui ajoute la portion libre  $p$  en tête de la chaîne et en fait son entrée. La portion  $p$  n'appartient pas à la chaîne au moment de l'appel.

**Question 12.** Écrire la fonction `supprime_dans_chaine(p)` qui supprime la portion  $p$  de la chaîne. La portion  $p$  est libre et appartient à la chaîne.

**Question 13.** Écrire la fonction `demarrage()` pour cette stratégie.

**Question 14.** Écrire la fonction `reserver(n, c)` pour cette stratégie. Préciser la complexité de `reserver(n, c)` en fonction de  $n$ ,  $TAILLE\_MEM$ , et tout autre quantité que vous jugerez utile pour mettre en valeur le gain en efficacité de cette stratégie d'implémentation. Comme cette fonction est très similaire à la fonction `reserver(n, c)` écrite pour la partie précédente, vous pouvez indiquer seulement les différences entre cette version et la précédente (en numérotant les lignes de la fonction précédente).

La FIGURE 8 présente le résultat de l'opération `liberer(p6)` sur la mémoire de la FIGURE 6.

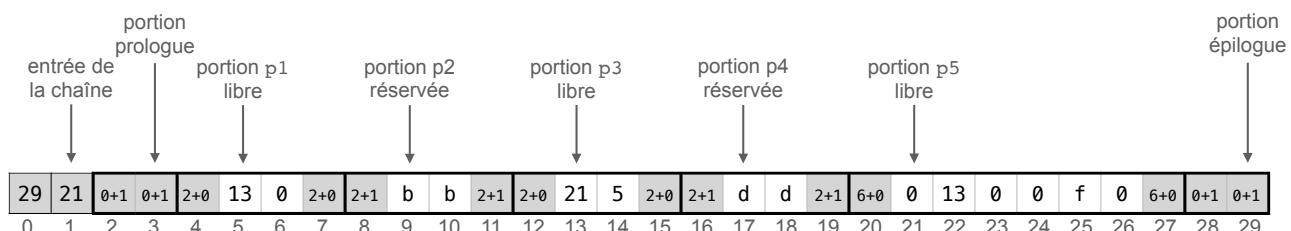


FIGURE 8 – Exemple de contenu de la mémoire après libération de  $p6$  - Partie IV.

Pour libérer une portion, on réalise comme dans la partie précédente une fusion avec les éventuelles portions libres adjacentes en mémoire, mais en les supprimant au préalable de la chaîne, puis en ajoutant en entrée de chaîne la portion libre créée par la fusion.

**Question 15.** Écrire la fonction `liberer(p)` pour cette stratégie. Préciser la complexité de `liberer(p)` en fonction de `TAILLE_MEM`. Là encore, il vous suffit de préciser les différences avec la fonction `liberer(p)` de la partie précédente.

\* \*  
\*



**A2021 – INFO**

**ÉCOLE DES PONTS PARISTECH,  
ISAE-SUPAERO, ENSTA PARIS,  
TÉLÉCOM PARIS, MINES PARIS,  
MINES SAINT-ÉTIENNE, MINES NANCY,  
IMT ATLANTIQUE, ENSAE PARIS,  
CHIMIE PARISTECH - PSL.**

Concours Mines-Télécom,  
Concours Centrale-Supélec (Cycle International).

**CONCOURS 2021****ÉPREUVE D'INFORMATIQUE COMMUNE**

**Durée de l'épreuve : 2 heures**

**L'usage de la calculatrice et de tout dispositif électronique est interdit.**

*Cette épreuve est commune aux candidats des filières MP, PC et PSI.*

*Les candidats sont priés de mentionner de façon apparente  
sur la première page de la copie :*

**INFORMATIQUE COMMUNE**

*L'énoncé de cette épreuve comporte 10 pages de texte.*

*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.*

Les sujets sont la propriété du GIP CCMP. Ils sont publiés sous les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 3.0 France. Tout autre usage est soumis à une autorisation préalable du Concours commun Mines Ponts.



## Marchons, marchons, marchons...

Ce sujet propose d'appliquer des techniques informatiques à l'étude de trois marches de nature différente :

- une marche concrète (partie I - Randonnée)
- une marche stochastique (partie II - Mouvement brownien d'une petite particule)
- une marche auto-évitante (partie III - Marche auto-évitante)

Les trois parties sont indépendantes. L'annexe à la fin du sujet (pages 8 à 10) fournit les canevas de code en Python que vous devrez reprendre dans votre copie pour répondre aux questions de programmation.

### Partie I. Randonnée

Lors de la préparation d'une randonnée, une accompagnatrice doit prendre en compte les exigences des participants. Elle dispose d'informations rassemblées dans deux tables d'une base de données :

- la table **Rando** décrit les randonnées possibles – la clef primaire entière **rid**, son nom, le niveau de difficulté du parcours (entier entre 1 et 5), le dénivelé (en mètres), la durée moyenne (en minutes) :

<b>rid</b>	<b>rnom</b>	<b>diff</b>	<b>deniv</b>	<b>duree</b>
1	La belle des champs	1	20	30
2	Lac de Castellane	4	650	150
3	Le tour du mont	2	200	120
4	Les crêtes de la mort	5	1200	360
5	Yukon Ho!	3	700	210
...	...	...	...	...

- la table **Participant** décrit les randonneurs – la clef primaire entière **pid**, le nom du randonneur, son année de naissance, le niveau de difficulté maximum de ses randonnées :

<b>pid</b>	<b>pnom</b>	<b>ne</b>	<b>diff_max</b>
1	Calvin	2014	2
2	Hobbes	2015	2
3	Susie	2014	2
4	Rosalyn	2001	4
...	...	...	...

Donner une requête SQL sur cette base pour :

- Q1** – Compter le nombre de participants nés entre 1999 et 2003 inclus.
- Q2** – Calculer la durée moyenne des randonnées pour chaque niveau de difficulté.
- Q3** – Extraire le nom des participants pour lesquels la randonnée n°42 est trop difficile.
- Q4** – Extraire les clés primaires des randonnées qui ont un ou des homonymes (nom identique et clé primaire distincte), sans redondance.

L'accompagnatrice a activé le suivi d'une randonnée par géolocalisation satellitaire et souhaite obtenir quelques propriétés de cette randonnée une fois celle-ci effectuée. Elle a exporté les données au format texte CSV (*comma-separated values* – valeurs séparées par des virgules) dans un fichier nommé **suivi\_rando.csv** : la première ligne annonce le format, les suivantes donnent les positions dans l'ordre chronologique.

Voici le début de ce fichier pour une randonnée partant de Valmorel, en Savoie, un bel après-midi d'été :

```
lat(°),long(°),height(m),time(s)
45.461516,6.44461,1315.221,1597496965
45.461448,6.444426,1315.702,1597496980
45.461383,6.444239,1316.182,1597496995
45.461641,6.444035,1316.663,1597496710
45.461534,6.443879,1317.144,1597496725
45.461595,6.4437,1317.634,1597496740
45.461562,6.443521,1318.105,1597496755
...
```

Le module `math` de Python fournit les fonctions `asin`, `sin`, `cos`, `sqrt` et `radians`. Cette dernière convertit des degrés en radians. La documentation donne aussi des éléments de manipulation de fichiers textuels :

```
fichier = open(nom_fichier, mode) ouvre le fichier, en lecture si mode est "r".
ligne = fichier.readline() récupère la ligne suivante de fichier ouvert en lecture avec open.
lignes = fichier.readlines() donne la liste des lignes suivantes.
fichier.close() ferme fichier, ouvert avec open, après son utilisation.
ligne.split(sep) découpe la chaîne de caractères ligne selon le séparateur sep : si ligne vaut "42,43,44",
alors ligne.split(",") renvoie la liste ["42", "43", "44"].
```

On souhaite exploiter le fichier de suivi d'une randonnée – supposé préalablement placé dans le répertoire de travail – pour obtenir une liste `coords` des listes de 4 flottants (latitude, longitude, altitude, temps) représentant les points de passage collectés lors de la randonnée.

À partir du canevas fourni en annexe et en ajoutant les `import` nécessaires :

**Q5** – Implémenter la fonction `importe_rando(nom_fichier)` qui réalise cette importation en retournant la liste souhaitée, par exemple en utilisant certaines des fonctions ci-dessus.

On suppose maintenant l'importation effectuée dans `coords`, avec au moins deux points d'instants distincts.

**Q6** – Implémenter la fonction `plus_haut(coords)` qui renvoie la liste (latitude, longitude) du point le plus haut de la randonnée.

**Q7** – Implémenter la fonction `déniveles(coords)` qui calcule les dénivélés cumulés positif et négatif (en mètres) de la randonnée, sous forme d'une liste de deux flottants. Le dénivelé positif est la somme des variations d'altitude positives sur le chemin, et inversement pour le dénivelé négatif.

On souhaite évaluer de manière approchée la distance parcourue lors d'une randonnée. On suppose la Terre parfaitement sphérique de rayon  $R_T = 6371$  km au niveau de la mer. On utilise la formule de haversine pour calculer la distance  $d$  du grand cercle sur une sphère de rayon  $r$  entre deux points de coordonnées (latitude, longitude)  $(\varphi_1, \lambda_1)$  et  $(\varphi_2, \lambda_2)$  :

$$d = 2 r \arcsin \left( \sqrt{\sin^2 \left( \frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left( \frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

On prendra en compte l'altitude moyenne de l'arc, que l'on complètera pour la variation d'altitude par la formule de Pythagore, en assimilant la portion de cercle à un segment droit perpendiculaire à la verticale.

**Q8** – Implémenter la fonction `distance(c1, c2)` qui calcule avec cette approche la distance en mètres entre deux points de passage. On décomposera obligatoirement les formules pour en améliorer la lisibilité.

**Q9** – Implémenter la fonction `distance_totale(coords)` qui calcule la distance en mètres parcourue au cours d'une randonnée.

## Partie II. Mouvement brownien d'une petite particule

De petites particules en suspension dans un liquide se déplacent spontanément sous l'effet de l'agitation thermique du milieu environnant, donnant ainsi naissance à des trajectoires apparemment chaotiques et peu régulières. Ce phénomène est à la base de la vie : l'agitation incessante des protéines permet aux réactions biochimiques de se produire à l'intérieur de nos cellules. Il a été observé expérimentalement en 1827 sur des grains de pollen en suspension dans l'eau par le botaniste ROBERT BROWN, d'où le nom de mouvement brownien. Son étude théorique par ALBERT EINSTEIN en 1905 a permis au physicien JEAN PERRIN d'estimer la valeur du nombre d'Avogadro dans une série d'expériences menées entre 1907 et 1909.

Dans cette partie, on s'intéresse à un modèle du mouvement brownien proposé par PAUL LANGEVIN en 1908. Dans ce modèle, la particule étudiée est supposée soumise à deux actions de la part du fluide :

- une force de frottement fluide  $\vec{f}_F = -\alpha \vec{v}$  ;
- une force  $\vec{f}_B$  aléatoire simulant l'action désordonnée des molécules d'eau sur la particule.

Le mouvement de cette particule est donc régi par l'équation différentielle :

$$\frac{d\vec{v}}{dt} = -\frac{\alpha \vec{v}}{m} + \frac{\vec{f}_B}{m}$$

Pour faire une simulation en deux dimensions, on prend une particule de masse  $m = 10^{-6}$  kg, on attribue à  $\alpha$  la valeur  $10^{-5}$  kg/s, on suppose enfin que  $\vec{f}_B$  change à chaque pas d'intégration, avec une direction isotrope aléatoire (l'angle suit une loi de probabilité uniforme) et une norme qui est la valeur absolue d'une variable aléatoire suivant une loi de probabilité gaussienne (loi normale) d'espérance  $\mu$  nulle et d'écart-type  $\sigma = 10^{-8}$  N.

On simule le vecteur d'état  $E = (x, y, \dot{x}, \dot{y})$  de la particule en intégrant  $\dot{E} = (\dot{x}, \dot{y}, \ddot{x}, \ddot{y})$  selon la méthode d'Euler.  $E$  et  $\dot{E}$  seront chacun représentés par une liste de 4 flottants.

L'instruction `assert expression` de Python vérifie la véracité d'une expression booléenne et interrompt brutalement l'exécution du programme si ce n'est pas le cas. Elle permet de vérifier très simplement une précondition ou un invariant, comme illustré à la ligne 10 du canevas.

Le module `random` fournit la fonction `uniform(bi, bs)` qui renvoie un flottant aléatoire entre les valeurs `bi` et `bs` incluses en utilisant une densité de probabilité uniforme ainsi que la fonction `gauss(mu, sigma)` qui renvoie un flottant aléatoire en utilisant une densité de probabilité gaussienne d'espérance `mu` et d'écart-type `sigma`.

Le module `math` fournit enfin les fonctions `cos` et `sin` (en radians), la fonction `sqrt` et la constante `pi`. La fonction `abs` est directement disponible.

En utilisant le canevas fourni en annexe et en ajoutant les `import` nécessaires :

**Q10** – Implémenter la fonction `vma(v1, a, v2)` (multiplication-addition sur des vecteurs) avec `v1` et `v2` des listes de flottants et `a` un scalaire flottant, qui renvoie une nouvelle liste de flottants correspondant à :

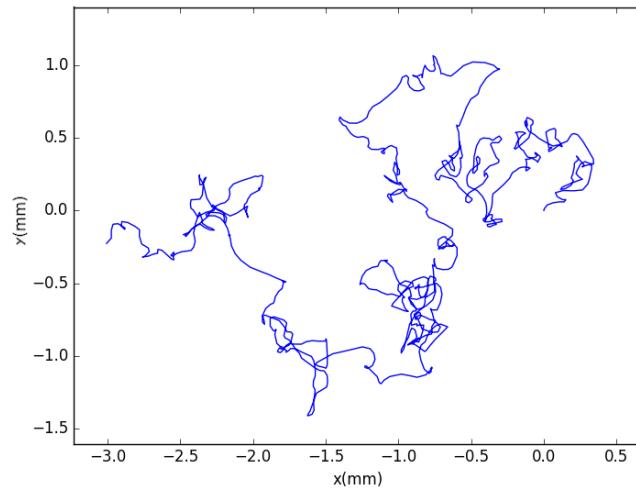
$$\vec{v} = \vec{v}_1 + a \vec{v}_2$$

La fonction vérifiera que les deux listes en entrée sont de même longueur avec `assert`.

**Q11** – Implémenter la fonction `derive(E)` qui renvoie la dérivée du vecteur d'état passé en paramètre d'après l'équation différentielle décrite en introduction de la partie.

**Q12** – Implémenter la fonction `euler(E0, dt, n)` pour résoudre numériquement l'équation différentielle par la méthode d'Euler avec `E0` le vecteur d'état initial, `dt` le pas d'intégration et `n` le nombre de pas de la simulation. La fonction renvoie la liste des  $n + 1$  vecteurs d'état.

À titre d'exemple, voici le tracé d'une trajectoire obtenue par cette méthode :



### Partie III. Marche auto-évitante

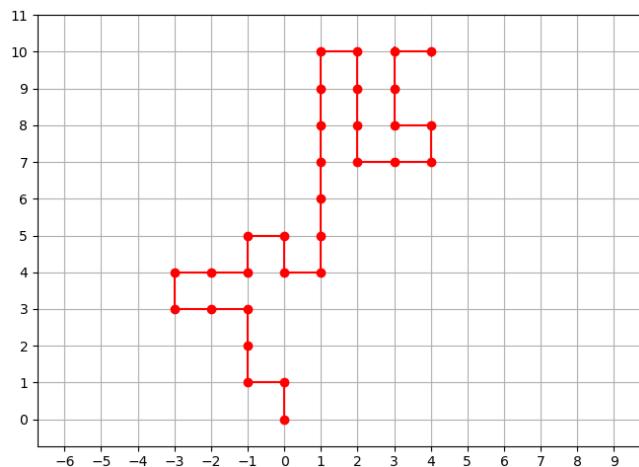
Une marche auto-évitante est un processus au cours duquel un point décrit un chemin auto-évitant, c'est-à-dire qui ne se recoupe jamais lui-même. Ce modèle peut servir lors de la modélisation d'une chaîne polymère. En effet, deux monomères distincts de la chaîne ne peuvent pas se trouver à deux positions identiques pour des raisons d'encombrement stérique.

Dans ce sujet, on appellera chemin auto-évitant (CAE) de longueur  $n$  tout ensemble de  $n + 1$  points  $P_i \in \mathbb{Z}^2$  pour  $0 \leq i \leq n$  tels que :

- $\forall i \quad \|P_{i+1} - P_i\| = 1$
- $\forall(i, j) \quad i \neq j \Rightarrow P_i \neq P_j$

Chaque point du chemin sera représenté par une liste à deux éléments entiers précisant les deux coordonnées (par exemple  $P_i = (5, -4)$  est représenté par la liste  $[5, -4]$ ). Le chemin lui-même est constitué de la liste des points, dans l'ordre dans lequel ils sont atteints. Les codes Python produits dans cette partie devront manipuler exclusivement des coordonnées entières.

Voici un exemple de représentation graphique de CAE de longueur 30 à partir de  $(0, 0)$  :



On s'intéresse dans un premier temps à une méthode naïve pour générer un chemin auto-évitant de longueur  $n$  sur une grille carrée. La méthode adoptée est une approche gloutonne :

1. Le premier point est choisi à l'origine :  $P_0 = (0, 0)$ .
2. En chaque position atteinte par le chemin, on recense les positions voisines accessibles pour le pas suivant et on en sélectionne une au hasard. En l'absence de positions accessibles l'algorithme échoue.
3. On itère l'étape 2 jusqu'à ce que le chemin possède la longueur désirée ou échoue.

Le module `random` fournit la fonction `randrange(a, b)` qui renvoie un entier compris entre  $a$  et  $b - 1$  inclus, ainsi que la fonction `choice(L)` qui renvoie l'un des éléments de la liste  $L$ , dans les deux cas choisis aléatoirement avec une probabilité uniforme.

L'expression `x in L` est une expression booléenne qui vaut `True` si `x` est l'un des éléments de  $L$  et `False` dans le cas contraire. On supposera que la méthode employée pour évaluer cette expression sur une liste est une recherche séquentielle.

La valeur spéciale `None` est utilisée en Python pour représenter une valeur invalide, inconnue ou indéfinie. L'expression booléenne `v is None` indique si la valeur de `v` est cette valeur spéciale.

En utilisant le canevas fourni en annexe et en ajoutant les `import` nécessaires :

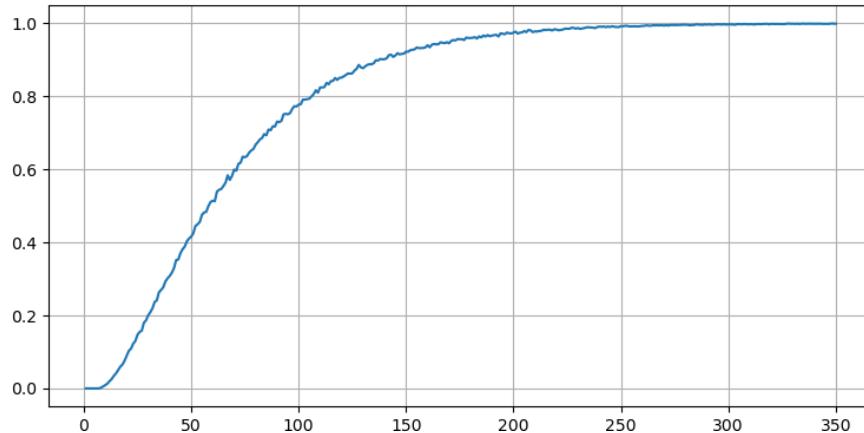
- ❑ **Q13** – Implémenter la fonction `positions_possibles(p, atteints)` qui construit la liste des positions suivantes possibles à partir du point `p`. La liste `atteints` contient les points déjà atteints par le chemin.
- ❑ **Q14** – Mettre en évidence graphiquement un exemple de CAE le plus court possible pour lequel, à une étape donnée, la fonction `positions_possibles` va renvoyer une liste vide. En prenant en compte les symétries, combien de tels chemins distincts existent pour cette longueur minimale ?
- ❑ **Q15** – Implémenter la fonction `genere_chemin_naif(n)` qui construit la liste des points représentant le chemin auto-évitant de longueur `n` et renvoie le résultat, ou bien renvoie la valeur spéciale `None` si à une étape `positions_possibles` renvoie une liste vide.
- ❑ **Q16** – Évaluer avec soin la complexité temporelle asymptotique dans le pire des cas de la fonction `genere_chemin_naif(n)` en fonction de `n`, en supposant que la fonction ne renvoie pas `None`.

Une personne curieuse et patiente a écrit le code suivant :

```

1  from chemin import genere_chemin_naif
2
3  N, M, L, P = 10000, 351, [], []
4
5  for n in range(1, M):
6      nb = 0
7      for i in range(N):
8          chemin = genere_chemin_naif(n)
9          if chemin is None:
10              nb += 1
11      L.append(n)
12      P.append(nb / N)
13
14  import matplotlib.pyplot as plt
15  plt.plot(L, P)
16  plt.grid()
17  plt.show()
```

Après un long moment, elle a obtenu le graphique suivant, volontairement laissé sans étiquettes d'axes :

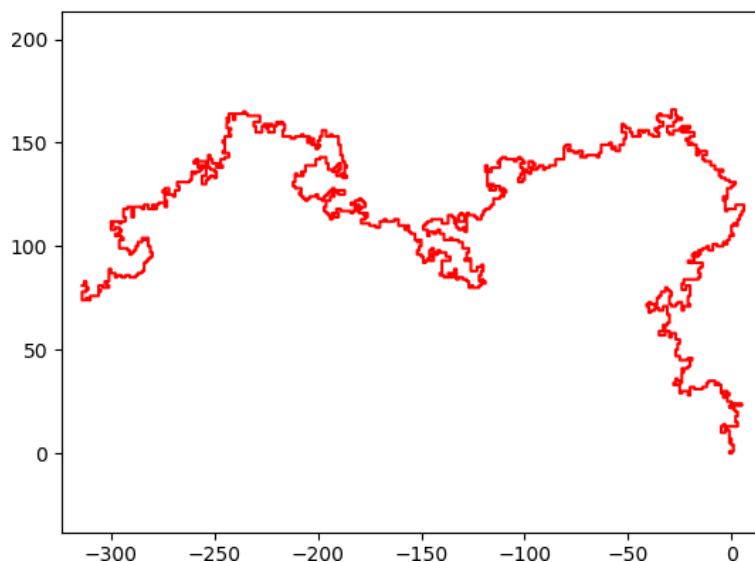


**Q17** – Décrire ce que représente ce graphique et interpréter sa signification pour la méthode naïve.

Afin d'éviter les inconvénients de la méthode précédente, on s'intéresse à une solution différente nommée *méthode du pivot*, proposée par Moti Lal en 1969. Son principe est le suivant :

1. On part d'un chemin auto-évitant arbitraire de longueur  $n$ . Ici, on choisira une initialisation très simple, le chemin droit  $[ [0, 0], [1, 0], [2, 0], \dots, [n, 0] ]$ .
2. On sélectionne au hasard un point, nommé pivot, entre le second et l'avant-dernier point du chemin, et un angle aléatoire de rotation parmi  $\pi$ ,  $\frac{\pi}{2}$  et  $-\frac{\pi}{2}$ .
3. On laisse les points avant le pivot inchangés et on fait subir à l'ensemble des points situés strictement après le pivot une rotation ayant pour centre le pivot et pour angle, l'angle choisi à l'étape 2 ci-dessus.
4. Si le chemin ainsi obtenu est auto-évitant, on le garde. Sinon, on reprend à l'étape 2 la sélection d'un pivot et d'un angle, jusqu'à en trouver une paire qui conviennent.
5. On répète les étapes 2 à 4 un certain nombre de fois. Le choix du nombre minimal de rotations à effectuer pour obtenir un chemin non corrélé au chemin initial est laissé de côté dans ce sujet.

Cette méthode permet de générer de manière efficace des marches auto-évitantes de plusieurs milliers de points, comme la marche ci-dessous de longueur 2000 :



Dans cet algorithme, une étape importante est la vérification qu'un chemin donné est auto-évitant. Pour vérifier cela on pourrait bien sûr s'y prendre comme dans la fonction `positions_posibles`, mais on adopte une méthode différente avec une fonction `est_CAE(chemin)` qui trie les points du chemin puis met à profit ce tri pour vérifier efficacement que le chemin est auto-évitant.

On utilise pour cela la fonction `sorted` qui renvoie une nouvelle liste triée par ordre croissant. Elle fonctionne sur tout type d'éléments disposant d'une relation d'ordre, y compris des listes pour lesquelles l'ordre lexicographique (ordre du premier élément, en cas d'égalité du second, etc.) est appliqué. On suppose de plus que la complexité temporelle asymptotique dans le pire des cas de cette fonction est la meilleure possible.

❑ **Q18** – Rappeler, sans la justifier, la complexité temporelle asymptotique dans le pire des cas attendue de `sorted` en fonction de la longueur de la liste à trier. Donner le nom d'un algorithme possible pour son implémentation.

En utilisant le canevas fourni en annexe et en ajoutant les `import` nécessaires :

❑ **Q19** – Implémenter la fonction `est_CAE(chemin)` qui vérifie si un chemin est auto-évitant en se basant sur `sorted` et renvoie un résultat booléen. Elle devra ne pas être de complexité temporelle asymptotique dans le pire des cas supérieure à la fonction `sorted` : vous prouverez ce dernier point.

❑ **Q20** – Implémenter la fonction `rot(p, q, a)` qui renvoie le point image du point `q` par la rotation de centre `p` et d'angle défini par la valeur de `a` : 0 pour  $\pi$ , 1 pour  $\frac{\pi}{2}$ , 2 pour  $-\frac{\pi}{2}$ .

❑ **Q21** – Implémenter la fonction `rotation(chemin, i_piv, a)` qui renvoie un nouveau chemin identique à `chemin` jusqu'au pivot d'indice `i_piv`, et ayant subi une rotation de `a` autour du pivot (même codage que la fonction précédente) sur la partie strictement après le pivot.

❑ **Q22** – Implémenter la fonction `genere_chemin_pivot(n, n_rot)` permettant de générer un chemin auto-évitant de longueur `n` en appliquant `n_rot` rotations. L'application d'une rotation peut nécessiter plusieurs tentatives.

❑ **Q23** – On considère un pivot, son point précédent et son point suivant. Quel est l'impact prévisible sur les rotations admissibles ? Suggérer un moyen de prendre en compte cette propriété pour améliorer l'algorithme.

*Fin de l'énoncé*

*Annexe canevas de codes Python sur les pages suivantes*

## Annexe : canevas de codes Python

## — Partie I : Randonnée

```

1  # import Python à compléter...
2
3  # importation du fichier d'une randonnée
4  def importe_rando(nom_fichier):
5      # À compléter...
6
7  coords = importe_rando("suivi_rando.csv")
8
9  # donne le point (latitude, longitude) le plus haut de la randonnée
10 def plus_haut(coords):
11     # À compléter...
12
13 print("point le plus haut", plus_haut(coords))
14 # exemple : point le plus haut [45.461451, 6.443064]
15
16 # calcul des dénivélés positif et négatif de la randonnée
17 def deniveles(coords):
18     # À compléter...
19
20 print("dénivelés", deniveles(coords))
21 # exemple : denivelés [4.05999999999945, -1.175999999999309]
22
23 RT = 6371 # rayon moyen volumétrique de la Terre en km
24
25 # distance entre deux points
26 def distance(c1, c2):
27     # À compléter...
28
29 print("premier intervalle", distance(coords[0], coords[1]), "m")
30 # exemple : premier intervalle 16.230964254992816 m
31
32 # distance totale de la randonnée
33 def distance_totale(coords):
34     # À compléter...
35
36 print("distance parcourue", distance_totale(coords), "m")
37 # exemple : distance parcourue 187.9700904658368 m

```

— Partie II : Mouvement brownien

```

1  # import Python à compléter...
2
3  # paramètres physiques
4  MU = 0.0      # N
5  SIGMA = 1E-8   # N
6  M = 1E-6      # kg
7  ALPHA = 1E-5   # kg/s
8
9  # vérification des hypothèses sur les paramètres
10 assert MU >= 0 and SIGMA > 0 and M > 0 and ALPHA > 0
11
12 # multiplication-addition vectorielle
13 def vma(v1, a, v2):
14     # À compléter...
15
16 # dérivée du vecteur d'état
17 def derive(E):
18     # À compléter...
19
20 # intégration par la méthode d'Euler
21 def euler(E0, dt, n):
22     Es = [ E0 ]
23     # À compléter...
24     return Es
25
26 # simulation
27 DT = 0.002    # durée du pas en secondes
28 N = 5000      # nombre de pas
29
30 Es = euler([ 0.0, 0.0, 0.0, 0.0 ], DT, N)
31
32 print("trajectoire", Es)

```

— Partie III : Chemin auto-évitant - méthode naïve

```

1 # import Python à compléter...
2
3 # positions auto-évitantes suivantes possibles
4 def positions_possibles(p, atteints):
5     possibles = []
6     # À compléter...
7     return possibles
8
9 # génération gloutonne d'un chemin de longueur n
10 # renvoie None en cas d'échec
11 def genere_chemin_naif(n):
12     chemin = [ [ 0, 0 ] ] # on part de l'origine
13     # À compléter...
14     return chemin
15
16 N = 10
17 print("chemin", genere_chemin_naif(N))

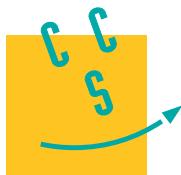
```

— Partie III : Chemin auto-évitant - méthode du pivot

```

1 # import Python à compléter...
2
3 # vérifie si un chemin est CAE
4 def est_CAE(chemin):
5     # À compléter...
6
7 # calcule la rotation de q autour de p selon a :
8 # Pi si a vaut 0, Pi/2 si a vaut 1, -Pi/2 si a vaut 2
9 def rot(p, q, a):
10    # À compléter...
11
12 # renvoie le chemin dont les points après i_pivot
13 # ont subi une rotation a codé comme précédemment
14 def rotation(chemin, i_pivot, a):
15    # À compléter...
16
17 # génère un chemin de longueur n (donc n+1 points)
18 def genere_chemin_pivot(n, n_rot):
19    # À compléter...
20
21 N, A = 1000, 2.3
22 print("chemin", genere_chemin_pivot(N, int(A * N)))

```



CONCOURS CENTRALE-SUPÉLEC

# Informatique

MP, PC, PSI, TSI

3 heures

Calculatrice autorisée

2021

## *Lancer de rayons*

*Et qu'au contraire les chats voient de nuit par le moyen des rayons qui tendent de leurs yeux vers les objets.*

— René Descartes, *Discours de la méthode - La dioptrique*(1637)

Le sujet présente une méthode de génération d'images en deux dimensions représentant des objets dans l'espace, éclairés par des sources lumineuses. Cette technique appelée *lancer de rayons* s'appuie sur les lois de propagation de la lumière, ce qui lui confère une grande qualité de rendu, au prix d'un temps de calcul souvent élevé. L'augmentation de la puissance des processeurs devrait bientôt pouvoir rendre cette technique compatible avec les jeux vidéo.



**Figure 1** Boîte en bois et boules<sup>1</sup>

Ce sujet propose d'illustrer les principales caractéristiques de la méthode, en se limitant à des scènes ne comportant que des sphères et des sources lumineuses ponctuelles.

De nombreuses questions sont indépendantes, il est toutefois demandé de les traiter dans l'ordre.

Les seuls langages informatiques autorisés dans cette épreuve sont Python et SQL. Pour répondre à une question, il est possible, et souvent souhaitable, de faire appel aux fonctions définies dans les questions précédentes.

Les modules `math` et `numpy` ont été rendus accessibles grâce à l'instruction

```
import math, numpy as np
```

Dans tout le sujet, le terme « liste » désigne une valeur de type `list`. Le terme « tableau » désigne une valeur de type `np.ndarray`. Enfin le terme « séquence » désigne une suite finie *indicable* et *itérable*.

— *Indicable* signifie que les éléments sont accessibles par des indices, `seq[0]` désignant le premier élément de la séquence `seq`.

<sup>1</sup> Exemple fourni avec le logiciel libre de lancer de rayons POV-Ray. Fichier `woodbox.pov`, POV-Ray scene file by Dan Farmer, sous licence Creative Commons Attribution-ShareAlike 3.0

— *Itérable* signifie que la séquence peut être parcourue dans une boucle par `for element in seq: ...`

Un tuple d'entiers, une liste d'entiers et un tableau d'entiers sont trois exemples de « séquence d'entiers ».

Les entêtes des fonctions demandées sont annotés pour préciser les types des paramètres et du résultat. Ainsi,

```
def uneFonction(n:int, X:[float], c:str, u) -> (np.ndarray, int):
```

signifie que la fonction `uneFonction` prend quatre paramètres `n`, `X`, `c` et `u`, où `n` est un entier, `X` une liste de nombres à virgule flottante, `c` une chaîne de caractères et le type de `u` n'est pas précisé. Cette fonction renvoie un couple constituée d'un tableau et d'un entier.

Il n'est pas demandé aux candidats d'annoter leurs fonctions, la rédaction pourra commencer par

```
def uneFonction(n, X, c, u):
    ...
```

De façon générale, une attention particulière sera portée à la lisibilité, la simplicité et la clarté du code proposé. L'utilisation d'identifiants significatifs, l'emploi judicieux de commentaires seront appréciés.

Une liste de fonctions potentiellement utiles est fournie à la fin du sujet.

## I Géométrie

Dans tout le sujet, l'espace est muni d'un repère  $(O, \vec{u}_x, \vec{u}_y, \vec{u}_z)$  orthonormé direct.  $\|\vec{v}\|$  désigne la norme euclidienne du vecteur  $\vec{v}$ . Le produit scalaire de deux vecteurs  $\vec{v}_1$  et  $\vec{v}_2$  est noté  $\vec{v}_1 \cdot \vec{v}_2$ , leur produit terme à terme (produit de Hadamard) est noté  $\vec{v}_1 \odot \vec{v}_2 : \vec{v}_1 \odot \vec{v}_2 = v_{1x}v_{2x}\vec{u}_x + v_{1y}v_{2y}\vec{u}_y + v_{1z}v_{2z}\vec{u}_z$ .

Tout point ou vecteur de l'espace est représenté en Python par le tableau de ses trois coordonnées cartésiennes, de type `float`. Pour faciliter la compréhension, un tel tableau est considéré de type `point` quand il désigne un point et de type `vecteur` quand il désigne un vecteur : `np.array([0., 0., 0.])` est considéré de type `point` quand il représente le point  $O$  et de type `vecteur` quand il représente le vecteur nul.

Beaucoup d'opérations classiques sur les vecteurs se transposent simplement dans la syntaxe de `numpy`. Si  $\vec{v}_1$ ,  $\vec{v}_2$  sont 2 vecteurs et  $t$  un réel, représentés respectivement par `v1`, `v2` et `t`, alors  $\vec{v}_1 + \vec{v}_2$ ,  $\vec{v}_1 - \vec{v}_2$ ,  $\vec{v}_1 \odot \vec{v}_2$  et  $t\vec{v}_1$  peuvent être calculés simplement par `v1 + v2`, `v1 - v2`, `v1 * v2` et `t * v1`.

Quand il n'y a pas de confusion possible, un objet mathématique est assimilé à sa représentation en Python. Ainsi, par exemple, « la fonction `f` prend en paramètre le vecteur  $\vec{v}_1$  » signifie que la fonction `f` accepte des valeurs de type `vecteur` représentant le vecteur  $\vec{v}_1$ .

On rappelle que la valeur du cosinus de l'angle formé par deux vecteurs unitaires correspond à la valeur de leur produit scalaire. Ainsi, si  $\vec{u}_1$  et  $\vec{u}_2$  sont deux vecteurs unitaires,  $\cos(\vec{u}_1, \vec{u}_2) = \vec{u}_1 \cdot \vec{u}_2$ . Par ailleurs, si  $\vec{u}$  est un vecteur unitaire et  $\vec{v}$  un vecteur quelconque, le projeté du vecteur  $\vec{v}$  dans la direction  $\vec{u}$  est donné par  $(\vec{u} \cdot \vec{v}) \vec{u}$ .

**Q 1.** Écrire une fonction d'entête

```
def vec(A:point, B:point) -> vecteur:
```

qui prend deux points en paramètre et renvoie le vecteur  $\overrightarrow{AB}$ .

**Q 2.** Écrire une fonction d'entête

```
def ps(v1:vecteur, v2:vecteur) -> float:
```

qui prend en paramètres deux vecteurs  $\vec{v}_1$  et  $\vec{v}_2$  et qui renvoie la valeur de leur produit scalaire  $\vec{v}_1 \cdot \vec{v}_2$ .

**Q 3.** Écrire une fonction d'entête

```
def norme(v:vecteur) -> float:
```

qui prend en paramètre un vecteur  $\vec{v}$  et qui renvoie  $\|\vec{v}\|$ , la valeur de sa norme euclidienne.

**Q 4.** Écrire une fonction d'entête

```
def unitaire(v:vecteur) -> vecteur:
```

qui prend en paramètre un vecteur  $\vec{v}$  non nul et qui renvoie  $\frac{1}{\|\vec{v}\|}\vec{v}$ , le vecteur unitaire correspondant.

On utilise dans ce sujet le modèle du rayon lumineux de l'optique géométrique. Un rayon lumineux issu du point  $S$  est une demi-droite d'origine  $S$ . Ce rayon est représenté en Python par le couple  $(S, \vec{u})$ , où  $S$  est le point de départ du rayon et  $\vec{u}$  le vecteur unitaire donnant la direction de propagation du rayon lumineux. Tout point  $M$  du rayon est tel que  $\overline{SM} = t\vec{u}$ , avec  $t \in \mathbb{R}^+$ . Un tel couple est désigné dans la suite par le type `rayon`.

Ainsi, en définissant `0 = np.array([0., 0., 0.])` et `u = np.array([0, 0.6, 0.8])`, le couple `(0, u)` représente le rayon issu de  $O$  dans la direction  $3\vec{u}_y + 4\vec{u}_z$ .

**Q 5.** Que font les fonctions `pt`, `dir` et `ra` ci-dessous ?

```
1 def pt(r:rayon, t:float) -> point:
2     assert t >= 0
3     (S, u) = r
4     return S + t * u
```

```

5   def dir(A:point, B:point) -> vecteur:
6       return unitaire(vec(A, B))

7   def ra(A:point, B:point) -> rayon:
8       return A, dir(A, B)

```

Comme toutes les fonctions définies dans ce sujet, les fonctions `pt`, `dir` et `ra` peuvent être utilisées dans la suite. Une sphère de centre  $C$  et de rayon  $r > 0$  est l'ensemble des points de l'espace situés à la distance  $r$  du point  $C$ . Ici, elle est représentée en Python par le couple  $(C, r)$ . Un tel couple est désigné par le type `sphère`.

**Q 6.** Écrire une fonction d'entête

```
def sp(A:point, B:point) -> sphère:
```

qui renvoie la sphère de centre  $A$  passant par  $B$ .

**Q 7.** Montrer qu'une droite passant par le point  $A$  de vecteur directeur  $\vec{u}$  et une sphère  $(C, r)$  sont sécantes si et seulement si l'équation d'inconnue  $t$

$$t^2 + 2t(\vec{u} \cdot \overrightarrow{CA}) + \|\overrightarrow{CA}\|^2 - r^2 = 0 \quad (1)$$

possède deux solutions réelles, éventuellement confondues.

**Q 8.** Écrire une fonction d'entête

```
def intersection(r:rayon, s:sphère) -> (point, float) or None:
```

qui renvoie le premier point de la sphère  $s$  frappé par le rayon lumineux  $r$  et la distance entre ce point et l'origine du rayon. La fonction renvoie `None` si le rayon ne coupe pas la sphère. On suppose que l'origine du rayon n'est pas située à l'intérieur de la sphère.

## II Optique

Les couleurs seront représentées par des tableaux de 3 nombres flottants, associés dans la suite au type `couleur`. Les trois composantes codent respectivement le niveau de rouge, vert et bleu de la couleur considérée (composantes RVB). Chaque composante est comprise entre 0 et 1. Plus la valeur est élevée, plus la composante contribue fortement à la couleur. Ainsi,  $(1, 1, 1)$  correspond au blanc,  $(0, 0, 0)$  au noir,  $(0.5, 0.5, 0.5)$  désigne un gris moyen et  $(0.6, 1, 0.6)$  un vert pastel.

Pour toute la suite, on a défini les variables globales

```
noir = np.array([0., 0., 0.])
blanc = np.array([1., 1., 1.])
```

### II.A – Visibilité

**Q 9.** Une source lumineuse ponctuelle  $S$  ne peut être vue d'un point  $P$  d'une sphère  $(C, r)$  que si la source est au-dessus de l'horizon de  $P$ , défini ici comme le plan tangent à la sphère en  $P$ . Donner une condition géométrique pour que la source soit au-dessus de l'horizon.

**Q 10.** Écrire une fonction booléenne, d'entête

```
def au_dessus(s:sphère, P:point, src:point) -> bool:
```

qui détermine si la source située en `src` est au-dessus de l'horizon du point `P` de la sphère `s`.

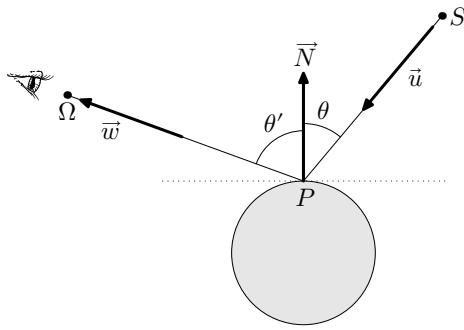
**Q 11.** On considère une scène contenant plusieurs sphères et une source lumineuse. Pour que la source soit visible d'un point  $P$  d'une sphère particulière, il faut que cette source soit au-dessus de l'horizon et qu'aucune autre sphère ne la cache. Écrire une fonction booléenne d'entête

```
def visible(obj:[sphère], j:int, P:point, src:point) -> bool:
```

où le paramètre `obj` est une liste contenant les sphères de la scène et `src` l'emplacement de la source lumineuse. Cette fonction détermine si la source est visible du point `P`, appartenant à la sphère `obj[j]`.

### II.B – Diffusion

On considère un point  $P$ , à la surface d'une sphère, éclairé sous l'incidence  $\theta$  par une source lumineuse  $S$  ponctuelle de couleur  $C_s = (R_s, V_s, B_s)$ . On note  $\vec{N}$  le vecteur unitaire normal à la sphère en  $P$ , dirigé vers l'extérieur de la sphère, et  $\vec{u}$  le vecteur unitaire du rayon lumineux qui éclaire  $P$  en provenance de la source. On considère que le point  $P$  diffuse la lumière de la source de manière isotrope dans tout le demi-espace délimité par le plan tangent à la sphère en  $P$  qui contient la source (figure 2). Autrement dit, le point  $P$  diffuse la lumière de la source dans toutes les directions  $\vec{w}$  telles que  $\vec{w} \cdot \vec{N} \geqslant 0$  (les vecteurs  $\vec{u}$ ,  $\vec{w}$  et  $\vec{N}$  ne sont pas forcément coplanaires) et la lumière diffusée ne dépend pas de la direction d'observation  $\vec{w}$ , en particulier elle ne dépend pas de  $\theta'$ .

**Figure 2** Parcours de la lumière de la source à l'œil

La faculté d'un objet à diffuser la lumière est modélisée par ses coefficients de diffusion  $k_{dr}$ ,  $k_{dv}$  et  $k_{db}$  qui mesurent son aptitude à réémettre les composantes rouge, verte et bleue. Chaque coefficient est un nombre à virgule flottante compris entre 0 et 1. On représente les coefficients de diffusion d'un objet par un triplet  $k_d = (k_{dr}, k_{dv}, k_{db})$  de type **couleur**. La couleur  $C_d = (R_d, V_d, B_d)$  de la lumière diffusée par le point  $P$  éclairé par la source  $S$  suit alors la loi de Lambert :

$$C_d = (k_d \odot C_s) \cos \theta \quad \text{soit} \quad (R_d, V_d, B_d) = (k_{dr} R_s \cos \theta, k_{dv} V_s \cos \theta, k_{db} B_s \cos \theta). \quad (2)$$

**Q 12.** Écrire une fonction d'entête

```
def couleur_diffusée(r:rayon, Cs:couleur, N:vecteur, kd:couleur) -> couleur:
```

qui renvoie la couleur de la lumière diffusée par le point  $P$  éclairé par un rayon lumineux  $r$  en provenance d'une source ponctuelle de couleur  $C_s$ . Les paramètres  $N$  et  $kd$  représentent respectivement le vecteur unitaire normal à l'objet en  $P$  et les coefficients de diffusion de l'objet (figure 2). La source  $S$  est supposée visible de  $P$ .

### II.C – Réflexion

Si la surface de l'objet est réfléchissante, au phénomène de diffusion s'ajoute le phénomène de réflexion. Lorsqu'un rayon lumineux ( $S, \vec{u}$ ) issu d'un point source  $S$  atteint un point  $P$  de la surface, il donne naissance au rayon réfléchi ( $P, \vec{w}$ ). Les lois de la réflexion de Descartes stipulent que, avec les notations de la figure 2,

- $\vec{u}$ ,  $\vec{w}$  et  $\vec{N}$  sont coplanaires ;
- $(\vec{u} + \vec{w}) \cdot \vec{N} = 0$ , ce qui correspond à  $\theta' = \theta$ .

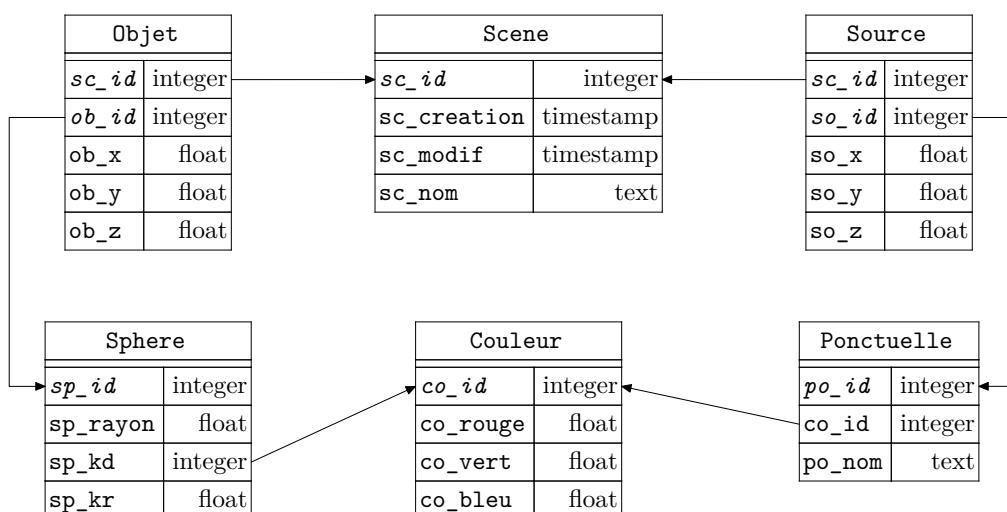
**Q 13.** Écrire une fonction d'entête

```
def rayon_réfléchi(s:sphère, P:point, src:point) -> rayon:
```

qui renvoie le rayon réfléchi par le point  $P$  de la sphère  $s$  en provenance de la source  $S$  placée en  $src$ . Le résultat est le couple  $(P, \vec{w})$  représentant le rayon émergent. La source  $S$  est supposée visible de  $P$ .

## III Enregistrement des scènes

Les différentes scènes à représenter sont enregistrées dans une base de données relationnelle. La figure 3 donne sa structure physique.

**Figure 3** Structure physique de la base de données des scènes

Cette base comporte les six tables listées ci-dessous avec la description de leurs colonnes :

- la table **Scene** répertorie les scènes
  - **sc\_id** identifiant (entier arbitraire) de la scène (clé primaire)
  - **sc\_creation** date de création de la scène
  - **sc\_modif** date de dernière modification de la scène
  - **sc\_nom** nom de la scène
- la table **Couleur** définit les couleurs utilisées
  - **co\_id** identifiant (entier arbitraire) de la couleur (clé primaire)
  - **co\_rouge** valeur de la composante rouge (dans l'intervalle [0, 1])
  - **co\_vert** valeur de la composante verte (dans l'intervalle [0, 1])
  - **co\_bleu** valeur de la composante bleue (dans l'intervalle [0, 1])
- la table **Sphère** liste les sphères utilisées pour construire les scènes
  - **sp\_id** identifiant (entier arbitraire) de la sphère (clé primaire)
  - **sp\_rayon** rayon de la sphère
  - **sp\_kd** coefficients de diffusion de la sphère (référence dans la table **Couleur**)
  - **sp\_kr** coefficient de réflexion de la sphère (cf. partie V)
- la table **Ponctuelle** répertorie les sources ponctuelles disponibles
  - **po\_id** identifiant (entier arbitraire) de la source ponctuelle (clé primaire)
  - **co\_id** couleur de la source (référence dans la table **Couleur**)
  - **po\_nom** nom de la source
- la table **Objet** fait le lien entre les scènes et les objets qu'elles contiennent, ses deux premières colonnes constituent sa clé primaire
  - **sc\_id** identifiant de la scène
  - **ob\_id** identifiant de l'objet
  - **ob\_x, ob\_y, ob\_z** coordonnées du centre de l'objet dans la scène considérée
- la table **Source** indique quelles sont les sources qui éclairent chaque scène, ses deux premières colonnes constituent sa clé primaire
  - **sc\_id** identifiant de la scène
  - **so\_id** identifiant de la source
  - **so\_x, so\_y, so\_z** coordonnées de la source dans la scène considérée

**Q 14.** Écrire une requête SQL qui donne le nom des scènes créées au cours de l'année 2021.

**Q 15.** Écrire une requête SQL qui donne, pour chaque scène, son identifiant et le nombre de sources qui l'éclairent.

**Q 16.** Écrire une requête SQL qui liste l'identifiant, les coordonnées du centre et le rayon de toutes les sphères contenues dans la scène dont le nom est **woodbox**. On suppose qu'une seule scène possède ce nom.

Il est possible en SQL de définir des fonctions utilisateur qui s'utilisent comme les fonctions SQL pré-définies. On dispose d'une fonction utilisateur booléenne de signature **OCCULTE(sc\_id, objr\_id, so\_id, objo\_id)** où **sc\_id**, **objr\_id**, **so\_id** et **objo\_id** sont les identifiants respectifs d'une scène, d'un objet de cette scène dit « récepteur », d'une source de cette scène et d'un autre objet de la scène dit « occultant ». La fonction renvoie TRUE si l'objet occultant projette son ombre sur l'objet récepteur lorsqu'il est éclairée par la source considérée.

**Q 17.** Écrire une requête SQL qui, pour la scène **woodbox**, renvoie tous les triplets **objr\_id, so\_id, objo\_id** pour lesquels l'objet d'identifiant **objo\_id** occulte la source d'identifiant **so\_id** pour l'objet d'identifiant **objr\_id**.

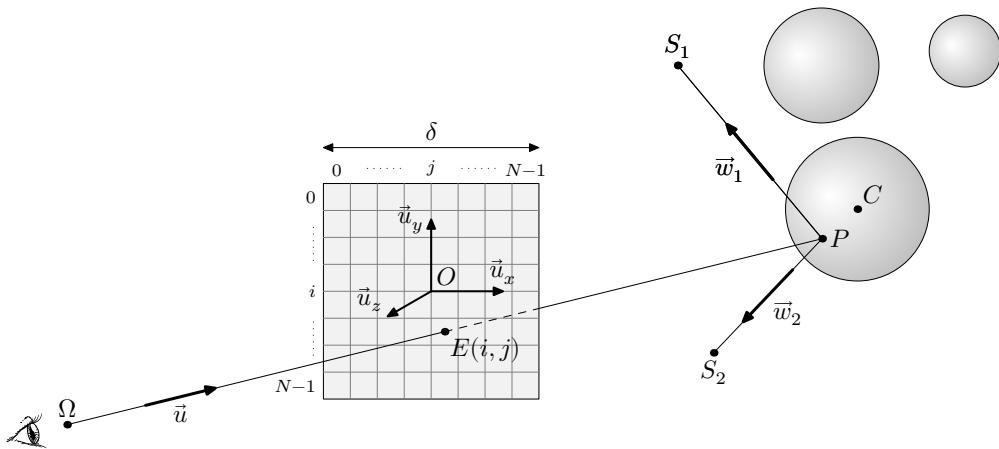
## IV Lancer de rayons

Cette partie implante l'algorithme qui génère l'image 2D de la scène 3D à visualiser. Pour représenter une scène en Python, on utilise les quatre variables globales suivantes :

- **Objet** est une liste des  $n_o$  objets (sphères de type **sphère**) contenues dans la scène à représenter ;
- **KdObj** est une liste de  $n_o$  tableaux de type **couleur** représentant les coefficients de diffusion des sphères, **KdObj[i]** est associé à la sphère **Objet[i]** ;
- **Source** est une liste de  $n_s$  points (de type **point**) donnant l'emplacement des  $n_s$  sources lumineuses (ponctuelles) qui éclairent la scène à représenter ;
- **ColSrc** est une liste de  $n_s$  couleurs, **ColSrc[i]** représente la couleur de la source placée en **Source[i]**.

Pour des raisons d'efficacité, il est d'usage de construire le parcours de la lumière de l'œil vers les sources : on « lance » des rayons. Cela est possible grâce au principe du retour inverse de la lumière : un rayon parcourt le

même chemin pour joindre 2 points  $A$  et  $B$ , qu'il aille de  $B$  vers  $A$  ou de  $A$  vers  $B$ . Dans la suite, les vecteurs unitaires caractérisant les rayons sont donc orientés dans le sens opposé au sens réel de propagation de la lumière.



**Figure 4** Une scène avec 3 sphères et 2 sources ponctuelles

Un écran translucide est placé dans le plan  $(O, \vec{u}_x, \vec{u}_y)$ . Le point  $O$  coïncide avec le centre de l'écran. Il sépare les objets de la scène, placés dans le demi espace  $z < 0$ , de l'œil  $\Omega$  placé de l'autre côté :  $z_\Omega > 0$ .

#### IV.A – Écran

L'écran est un carré de côté  $\delta$ , divisé en  $N \times N$  cases ( $N$  pair) qui représentent les pixels de l'image finale. Les variables globales `Delta` et `N` contiennent respectivement les valeurs de  $\delta$  et  $N$ .

**Q 18.** Écrire une fonction d'entête

```
def grille(i:int, j:int) -> point:
```

qui renvoie les coordonnées cartésiennes du point  $E$ , centre de la case repérée par les indices  $(i, j)$  de la grille (figure 4).

**Q 19.** Écrire une fonction d'entête

```
def rayon_ecran(omega:point, i:int, j:int) -> rayon:
```

qui renvoie le rayon issu du point  $\Omega$  et passant par  $E(i, j)$ .

#### IV.B – Couleur d'un pixel

Dans un premier temps, les objets de la scène à représenter sont supposés parfaitement mats, ils se contentent de diffuser la lumière des sources sans la réfléchir.

On considère un point  $P$  d'un objet de la scène atteint par un rayon issu de l'œil. On note  $E$  le point de l'écran coupé par ce rayon allant de l'œil au point  $P$ . La couleur  $C_d = (R_d, V_d, B_d)$  diffusée par  $P$  est la somme des couleurs diffusées par ce point, dues à l'éclairement des sources visibles du point  $P$ .

$$C_d = \sum_{S_i \text{ visibles}} (k_d \odot C_{s_i}) \cos \theta_i. \quad (3)$$

On suppose que les couleurs ne saturent pas : toutes les composantes de  $C_d$  restent inférieures à 1. La couleur  $C_d$  est affectée au point  $E$  de l'écran.

**Q 20.** Écrire une fonction d'entête

```
def interception(r:rayon) -> (point, int) or None:
```

qui prend en paramètre un rayon  $r$  et qui renvoie le premier point matériel de la scène atteint par ce rayon ainsi que l'indice de la sphère concernée dans la liste `Objet`. Si le rayon n'intercepte aucune sphère, la fonction renvoie `None`.

**Q 21.** Écrire une fonction d'entête

```
def couleur_diffusion(P:point, j:int) -> couleur:
```

qui renvoie la couleur diffusée par le point  $P$  appartenant à la sphère `Objet[j]`. Si aucune source n'éclaire  $P$ , la fonction renvoie `noir`.

#### IV.C – Constitution de l'image

L'image de la scène que l'on souhaite obtenir est construite dans un tableau à 3 dimensions, de taille  $N \times N \times 3$ , associé au type `image`. L'affectation de la couleur `c` de type `couleur` au pixel  $(i, j)$  de l'image `im` s'écrira simplement `im[i, j] = c`.

**Q 22.** Écrire une fonction d'entête

```
def lancer(omega:point, fond:couleur) -> image:
```

qui génère l'image associée à la scène. Si un rayon n'intercepte aucun objet, le pixel correspondant est de couleur `fond`.

#### IV.D – Complexités

Les complexités asymptotiques seront exprimées en fonction du nombre  $N$  de lignes de l'image, du nombre  $n_o$  d'objets et du nombre  $n_s$  de sources.

**Q 23.** Calculer la complexité temporelle de la fonction `lancer` dans le meilleur des cas en caractérisant la situation correspondante.

**Q 24.** Calculer la complexité temporelle de la fonction `lancer` dans le pire des cas en caractérisant la situation correspondante.

## V Améliorations

### V.A – Prise en compte de la réflexion

Désormais les sphères sont supposées au moins partiellement réfléchissantes. Un rayon issu de l'œil peut alors se réfléchir successivement sur plusieurs sphères.

**Q 25.** Écrire une fonction d'entête

```
def réflexions(r:rayon, rmax:int) -> [(point, int)]:
```

qui renvoie une liste de couples  $(P_k, i_k)$  correspondant aux points successivement rencontrés par le rayon `r` au fur et à mesure de ses réflexions. Dans chaque couple,  $P_k$  est le point où a lieu la  $(k + 1)^{\text{ème}}$  réflexion et  $i_k$  est l'indice de l'objet contenant  $P_k$ . Le résultat comporte au plus `rmax` éléments, les éventuelles réflexions ultérieures ne sont pas prises en compte.

Le pouvoir réfléchissant d'un objet est caractérisé par un coefficient de réflexion  $k_r$  propre à chaque objet,  $0 \leq k_r \leq 1$ . Les coefficients de réflexion des objets de la scène (de type `float`) sont stockés dans une liste globale `KrObj` à  $n_o$  éléments, telle que `KrObj[i]` est le coefficient de réflexion de l'objet `Objet[i]`.

En tenant compte des phénomènes de diffusion et de réflexion, la couleur  $C_k = (R_k, V_k, B_k)$  du point  $P_k$  vaut

$$C_k = C_{dk} + k_r C_{k+1} \quad (4)$$

où  $C_{dk}$  désigne la couleur diffusée par le point  $P_k$  (cf. IV.B) et  $C_k$  vaut `noir` si  $k$  est supérieur au nombre de réflexions considérées.

**Q 26.** Écrire une fonction d'entête

```
def couleur_perçue(r:rayon, rmax:int, fond:couleur) -> couleur:
```

qui renvoie la couleur du premier point de la scène rencontré par le rayon `r` en tenant compte d'au maximum `rmax` réflexions de ce rayon. Si le rayon ne rencontre aucun objet, la fonction renvoie la couleur `fond`.

**Q 27.** Écrire une fonction d'entête

```
def lancer_complet(omega:point, fond:couleur, rmax:int) -> image:
```

qui construit l'image de la scène en tenant compte des diffusions et des réflexions.

**Q 28.** Exprimer la nouvelle complexité dans le pire cas.

### V.B – Une optimisation

On construit, à partir de la base de données, les deux listes globales :

- `IdObj` telle que `IdObj[i]` soit l'identifiant dans la base de données (`ob_id`) de la sphère `Objet[i]` ;
- `IdSrc` telle que `IdSrc[i]` soit l'identifiant dans la base de données (`so_id`) de la source `Source[i]`.

**Q 29.** Écrire la fonction d'entête

```
def table_risque(risque:[int, int, int]) -> [[[int]]]:
```

qui prend en paramètre une liste de triplets correspondant au résultat de la requête SQL de la question 17 et construit une liste de listes de listes d'entiers telle que, si `res` est le résultat de la fonction, `res[i][j]` donne la liste (éventuellement vide) des indices des objets susceptibles de masquer la source `Source[j]` pour un point de l'objet `Objet[i]`.

**Q 30.** Le résultat de la fonction `table_risque` est conservé dans la variable globale `TableRisque`. Écrire la fonction `visible_opt` d'entête

```
def visible_opt(j:int, k:int, P:point) -> bool:
```

qui prend en paramètres l'indice `j` d'une source, l'indice `k` d'une sphère et un point `P` de cette sphère et qui, comme la fonction `visible` de la question 11, détermine si la source `Source[j]` est visible à partir du point `P`.

## Opérations et fonctions disponibles en Python et en SQL

### Constantes

- `math.inf`, `np.inf` correspondent à  $+\infty$ , n'importe quel nombre est strictement inférieur à cette valeur.

### Fonctions Python diverses

- `range(n)` itérateur sur les  $n$  premiers entiers ( $\llbracket 0, n - 1 \rrbracket$ ).  
`list(range(5)) → [0, 1, 2, 3, 4]`.
- `range(d, f, p)` où  $d$ ,  $f$  et  $p$  sont des entiers, itérateur sur les entiers  $(r_i = d + ip \mid r_i < f)_{i \in \mathbb{N}}$  si  $p > 0$  et  $(r_i = d + ip \mid r_i > f)_{i \in \mathbb{N}}$  si  $p < 0$ . Le paramètre  $p$  est optionnel avec une valeur par défaut de 1.  
`list(range(1, 5)) → [1, 2, 3, 4] ; list(range(20, 10, -2)) → [20, 18, 16, 14, 12]`.
- `math.sqrt(x)` calcule la racine carrée du nombre  $x$ .

### Opérations sur les listes

- `len(L)` donne le nombre d'éléments de la liste  $L$ .
- $L_1 + L_2$  construit une liste constituée de la concaténation des listes  $L_1$  et  $L_2$ .
- `e in L` et `e not in L` déterminent si l'objet  $e$  figure dans la liste  $L$ . Ces opérations ont une complexité temporelle en  $O(\text{len}(L))$ .
- `L.append(e)` ajoute l'élément  $e$  à la fin de la liste  $L$ .
- `L.index(e)` renvoie le plus petit entier  $i$  tel que  $L[i] == e$ . Lève l'exception `ValueError` si l'élément  $e$  n'apparaît pas dans la liste. Cette opération a une complexité temporelle en  $O(\text{len}(L))$ .
- `L.sort()` trie en place la liste  $L$  (qui est donc modifiée) en réordonnant ses éléments dans l'ordre croissant.

### Opérations sur les tableaux (`np.ndarray`)

- `np.array(s, dtype)` crée un nouveau tableau contenant les éléments de la séquence  $s$ . La taille de ce tableau est déduite du contenu de  $s$ . Le paramètre optionnel `dtype` précise le type des éléments du tableau créé.
- `np.empty(n, dtype)`, `np.empty((n, m), dtype)` crée respectivement un tableau à une dimension de  $n$  éléments et un tableau à  $n$  lignes et  $m$  colonnes dont les éléments, de valeurs indéterminées, sont de type `dtype`. Si le paramètre `dtype` n'est pas précisé, il prend la valeur `float`.
- `np.zeros(n, dtype)`, `np.zeros((n, m), dtype)` fonctionne comme `np.empty` en initialisant chaque élément à la valeur zéro pour les types numériques ou `False` pour les types booléens.
- `np.sum(a)` ou `a.sum()` renvoie la somme des éléments du tableau  $a$ .
- `np.inner(a, b)` calcule la somme des produits terme à terme dans le cas où  $a$  et  $b$  sont deux tableaux à une dimension de même taille.
- `np.all(a)` vaut `True` si tous les éléments du tableau  $a$  ont une valeur logique « vrai ».
- `np.any(a)` vaut `True` si au moins un des éléments du tableau  $a$  a une valeur logique « vrai ».

### SQL

- `SELECT ... FROM t1 JOIN t2 ON ...` effectue une requête sur le résultat du produit cartésien entre les tables  $t1$  et  $t2$  restreint par la condition indiquée. Par exemple `t1.a = t2.b` permet de limiter le résultat aux lignes pour lesquelles la colonne  $a$  de la table  $t1$  est égale à la colonne  $b$  de la table  $t2$ .
- `SELECT ... FROM t AS t1 JOIN t AS t2 ON ...` effectue une requête sur le résultat du produit cartésien de la table  $t$  avec elle-même restreint par la condition indiquée. Le premier exemplaire de la table  $t$  est désigné par  $t1$  et le second par  $t2$ .
- La fonction `EXTRACT(part FROM t)` extrait un élément de  $t$ , expression de type `date`, `time`, `timestamp` (jour et heure) ou `interval` (durée). `part` peut prendre les valeurs `year`, `month`, `day` (jour dans le mois), `doy` (jour dans l'année), `dow` (jour de la semaine), `hour`, etc.
- Les fonctions d'agrégation `SUM(e)`, `AVG(e)`, `MAX(e)`, `MIN(e)`, `COUNT(e)`, `COUNT(*)` calculent respectivement la somme, la moyenne arithmétique, le maximum, le minimum, le nombre de valeurs non nulles de l'expression  $e$  et le nombre de lignes pour chaque groupe de lignes défini par la cause `GROUP BY`. Si la requête ne comporte pas de clause `GROUP BY` le calcul est effectué pour l'ensemble des lignes sélectionnées par la requête.

• • • FIN • • •

SESSION 2021



PSI5IN

## ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

### ÉPREUVE SPÉCIFIQUE - FILIÈRE 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.*

#### RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

**Le sujet est composé de trois parties indépendantes.**

L'épreuve est à traiter en langage **Python**. La syntaxe de Python est rappelée en **Annexe** en fin de sujet.

Les différents algorithmes doivent être rendus dans leur forme définitive sur le document réponse dans l'espace réservé à cet effet en respectant les éléments de syntaxe du langage (les brouillons ne sont pas acceptés).

La réponse ne doit pas se cantonner à la rédaction de l'algorithme sans explication, les programmes doivent être expliqués et commentés.

Énoncé et Annexes : 12 pages

Document Réponse : 8 pages

**Seul le Document Réponse doit être rendu dans son intégralité.**

# Autour des montres multisports

## Présentation

On s'intéresse dans ce sujet aux montres multisport, en développement croissant depuis les dix dernières années.

Ces montres permettent aux sportifs amateurs ou professionnels d'enregistrer diverses données physiques et physiologiques. Elles permettent maintenant de connaître sa position GPS, d'enregistrer un trajet et tous les paramètres associés, de suivre un trajet prédéfini. Les plus sophistiquées intègrent également un altimètre, un baromètre, un thermomètre, une boussole, un lecteur MP3, etc.



On se limite dans ce sujet à :

- l'acquisition et le traitement des données GPS sur un parcours : **partie I**;
- l'acquisition du rythme cardiaque : **partie II**;
- le partage des activités : **partie III**.

Dans tout le sujet, on supposera que les bibliothèques `numpy` et `matplotlib.pyplot` sont importées par `from numpy import *` et `from matplotlib.pyplot import *`

## Partie I - Acquisition et traitement des données GPS

### I.1 - Introduction

Dans la montre, un module électronique capte les signaux GPS à l'aide d'une antenne et décode ces signaux pour générer une chaîne de caractères constituée de plusieurs lignes. On ne s'intéresse qu'au traitement de cette chaîne de caractères qui est transmise par le module GPS.

Les données envoyées par le récepteur GPS suivent la norme NMEA083 définie dans l'**annexe 1**. On suppose que la variable `donneesGPS` est une chaîne de caractères reçue par le récepteur. Cette chaîne contient plusieurs lignes dont le séparateur est le caractère spécial '`\n`'. Pour la lisibilité, cette chaîne est affichée avec les sauts de ligne.

```
"$GPRMC,092828.00,A,4754.68988,N,00154.69147,E,0.444,,070619,,,A*7C\n
$GPVTG,,T,M,0.444,N,0.822,K,A*2F\n
$GPGGA,092828.00,4754.68988,N,00154.69147,E,1,04,2.61,84.3,M,46.5,M,*64\n
$GPGSA,A,3,25,29,24,32,,,,,,3.99,2.61,3.02*0E\n
$GPGSV,3,1,11,02,32,075,,04,,22,06,15,037,,12,47,072,16*4A\n
$GPGSV,3,2,11,14,38,276,11,24,18,141,19,25,80,353,24,29,56,196,30*7F\n
$GPGSV,3,3,11,31,34,306,25,32,35,250,36,33,32,202,27*4F\n
$GPGLL,4754.68986,N,00154.69154,E,092828.00,A,A*6B\n"
```

## I.2 - Récupération des données brutes

L'objectif est de récupérer l'heure, la latitude, la longitude et l'altitude dans les données de la trame GPGGA (voir **annexe 1**). Cette procédure va se décomposer en 3 étapes :

- récupérer la trame GPGGA dans la série de trames reçues ;
- vérifier la validité de la trame ;
- extraire les données.

Pour répondre aux questions suivantes, il est conseillé d'utiliser les propriétés et fonctions sur les caractères et chaînes de caractères données en **annexe 2**.

Pour récupérer la trame GPGGA, on a écrit la fonction `recupererGPGGA(chaine)` (donnée sur le **Document Réponse**) qui prend en argument la chaîne de caractères `chaine` et qui renvoie la ligne correspondant à la trame GPGGA si elle existe sous forme de liste découpée suivant les virgules. Sinon elle renvoie une liste vide.

La fonction `indice_GPGGA(listeChaine)` renvoie l'indice de l'élément contenant '`$GPGGA`' de `listeChaine`.

- Q1.** Expliquer les lignes 2 à 8 de cette fonction.
- Q2.** Écrire une fonction `indice_GPGGA(listeChaine)` prenant en argument une liste de chaînes de caractères et qui renvoie l'indice de la position de la chaîne contenant '`$GPGGA`' quand elle existe ou la longueur de `listeChaine` sinon. Vous utiliserez une boucle `while`.
- Q3.** Donner l'instruction à écrire pour récupérer la trame GPGGA de la variable `donneesGPS` et la stocker dans la variable `listeGPGGABrute`.

## I.3 - Test de validité de la trame GPGGA

Pour que la trame soit valide, il faut vérifier que :

- les valeurs soient présentes ;  
par exemple, la trame "`$GPGGA,092811.00,,,,,0,00,99.99,,,,,*65\n`" n'est pas valide ;
- le coefficient de précision soit inférieur à 5 ;
- la somme de contrôle soit valide. La somme de contrôle est obtenue par une opération " ou exclusif " entre une représentation binaire de chaque caractère entre '\$' et '\*' (ces deux caractères étant exclus). Sa valeur dans la trame est donnée en base hexadécimale.

On suppose dans cette sous-partie que la trame, constituée de 14 éléments, a été récupérée et stockée dans `listeGPGGABrute` soit ici :

```
listeGPGGABrute= ['GPGGA', '092828.02', '4754.68988', 'N', '00154.69147', 'E', '1', '04', '2.61', '84.3', 'M', '46.5', 'M', '*64']
```

Pour les questions suivantes, `trame` correspond à une variable ayant la même forme que `listeGPGGABrute`.

- Q4.** Écrire une fonction `testPresence(trame)` qui prend en argument la liste de chaînes de caractères `trame` correspondant à une trame GPGGA et qui renvoie `True` si les données sont présentes et non vides, `False` sinon.
- Q5.** Écrire une fonction `testPrecision(trame)` qui prend en argument la liste de chaînes de caractères `trame` correspondant à une trame GPGGA et qui renvoie `True` si la précision est inférieure à 5, `False` sinon.

On donne dans le **Document Réponse** une fonction `testSC(trame)` qui prend la liste de chaînes de caractères `trame` correspondant à une trame GPGGA et qui renvoie `True` si la trame est valide, `False` sinon.

**Q6.** Donner la valeur renvoyée par la fonction `testSC` appliquée à la liste `['GPG', '*A0']`.

Donner notamment la valeur de la variable `csHex`. On donne les résultats suivants : `bin(ord('G'))` renvoie `'0b01000111'`, `bin(ord('P'))` renvoie `'0b01010000'`, `bin(ord('*'))` renvoie `'0b00101010'`, `bin(ord('A'))` renvoie `'0b01000001'`, `bin(ord('0'))` renvoie `'0b00110000'`.

#### I.4 - Récupération des données

La trame étant valide, on recherche maintenant à récupérer les données qui nous intéressent en les stockant dans le format adéquat :

- l'heure exprimé en secondes ;
- la latitude exprimée en degrés (0 correspond à l'équateur, elle varie de  $-90^\circ$  à  $90^\circ$ ). Au Nord l'angle sera compté positivement et au Sud négativement ;
- la longitude exprimée en degrés (0 correspond au méridien de Greenwich, elle varie de  $-180^\circ$  à  $180^\circ$ ). L'Ouest sera compté positivement et l'Est négativement ;
- l'altitude exprimée en mètres.

Pour la latitude, comme la longitude, les minutes d'angle sont toujours représentées avec 8 caractères (dont la virgule) alors que les degrés sont représentés avec 2 ou 3 caractères.

**Q7.** Écrire une fonction `convHoraire(chHoraire)` qui prend en argument la chaîne de caractères `chHoraire` représentant l'heure avec le format de la norme GPGGA (092828.00 correspondant à 09 h 28 min 28,00 secondes) et qui renvoie la valeur de l'heure exprimé en secondes.

**Q8.** Écrire une fonction `convAngle(chAngle, chCard)` qui prend en argument la chaîne de caractères `chAngle` correspondant à la latitude ou la longitude, ainsi que la chaîne de caractères `chCard` correspondant à la direction (N/S ou E/O) et qui renvoie le flottant signé correspondant à la valeur de l'angle en degré (il faudra convertir les minutes d'angle en valeur décimale).

**Q9.** Compléter l'ébauche de la fonction `recupDonnees(trame)` qui prend en argument `trame` correspondant à une trame GPGGA et qui renvoie une liste de 4 réels contenant (Heure, Latitude, Longitude, Altitude).

#### I.5 - Sauvegarde d'une activité

Lors d'une activité, on récupère toutes les secondes :

- les données du GPS : heure (en seconde), latitude (en degré), longitude (en degré) et altitude (en mètre) ;
- la fréquence cardiaque.

Toutes les secondes, ces 5 données sont enregistrées dans un fichier texte, en les séparant par une virgule. Chaque ligne du fichier correspond à un "point" de mesure.

Exemple d'une ligne du fichier texte : `"34108,47.911498,001.911526,0084.3,105\n"`.

L'heure, en secondes, est un entier représenté avec 5 caractères. On garde 6 chiffres après la virgule pour les angles en degrés. La partie entière de l'altitude est représentée avec 4 caractères et on prend une précision au décimètre. La fréquence cardiaque est un entier représenté avec 3 caractères. Tous les caractères sont stockés sur 1 octet (ascii).

**Q10.** On suppose que l'on réalise une activité d'une heure. Donner l'ordre de grandeur de la taille mémoire en octets du fichier texte généré. Dans le cahier des charges, on impose que la montre peut sauvegarder 200 h d'activités. Donner l'ordre de grandeur de la taille mémoire nécessaire au stockage des données pour répondre au cahier des charges.

La mémoire des montres est de l'ordre de 20 Mo.

**Q11.** Proposer une solution permettant de ne pas dépasser la mémoire disponible.

## Partie II - Acquisition du rythme cardiaque

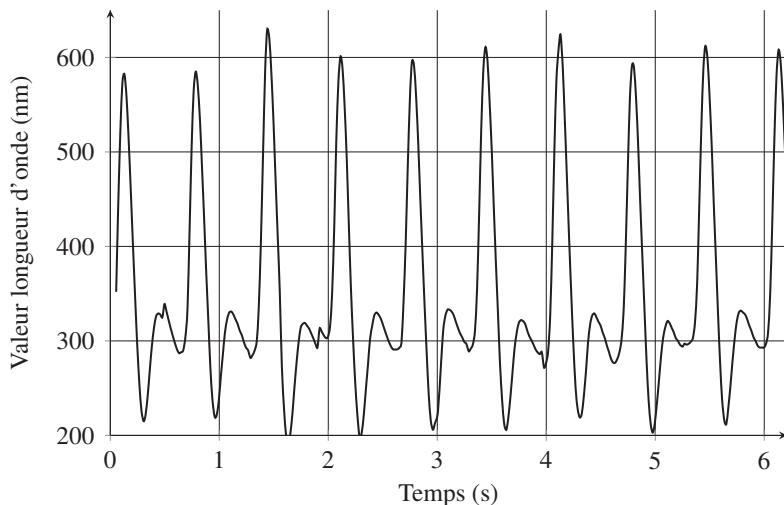
Lors d'une activité sportive, les montres multisport permettent l'enregistrement et la consultation du rythme cardiaque. Cela permet au sportif de contrôler son effort et d'optimiser ses performances. Cette donnée est intéressante durant les compétitions et les entraînements.

La première méthode utilisée est d'enregistrer l'ElectroCardioGramme (ECG). On mesure l'activité électrique du cœur à l'aide d'électrodes. Cette méthode nécessite d'installer une ceinture cardiothoracique qui mesure et envoie les données à la montre par Bluetooth ou ANT+ en fonction des marques.

La seconde méthode est la PhotoPlestysmoGraphie (PPG) ou Oxymétrie de pouls. L'hémoglobine, lorsqu'elle est chargée en oxygène, absorbe davantage les infrarouges que lorsqu'elle n'est pas chargée en oxygène. On peut donc avec un émetteur et un récepteur infrarouge mesurer en temps réel la saturation en oxygène du sang et en déduire le rythme cardiaque. En centre hospitalier, le capteur est positionné au bout d'un doigt. L'épaisseur de peau étant faible et la zone très vascularisée, on utilise des longueurs d'onde " optimales " de l'ordre de 800 nm (lumière rouge). Dans les montres, la prise de mesure s'effectue au poignet. La peau y étant plus épaisse, on utilise des longueurs d'onde de l'ordre de 570 nm (lumière verte) qui pénètrent mieux les tissus.



Le capteur PPG renvoie une valeur de longueur d'onde avec un temps d'échantillonnage  $T_e = 0,02 \text{ s}$ . Tous les  $T_e$ , on enregistre la valeur de ce signal (sans unité) dans une liste notée `signal` et le temps (en s) dans une liste notée `temps`. Chaque élément de la liste `signal` est un nombre entier, image de la quantité d'infrarouge absorbée.



**Figure 1** - Mesure PPG : tracé de la liste `signal` en fonction de la liste `temps`

Un essai d'une durée d'environ 6 secondes permet de tracer la courbe de la **figure 1**.

On observe que la valeur du signal infrarouge se décompose en deux valeurs :

- une valeur quasi constante, de l'ordre de 300 sur la **figure 1**, qui correspond à l'absorption due aux tissus, au flux veineux et à une partie constante du flux artériel;
- une partie variable due exclusivement à la variation du flux artériel.

### II.1 - Application d'un filtre passe-bas

Pour atténuer certaines perturbations, on applique un filtre passe-bas. Il est caractérisé par l'équation différentielle :

$$\frac{ds(t)}{dt} + 2\pi f_c s(t) = 2\pi f_c e(t) \quad (1)$$

où  $e(t)$  est le signal d'entrée du filtre,  $s(t)$  le signal en sortie du filtre et  $f_c$  la fréquence de coupure du filtre.

On peut intégrer l'équation (1) sur un intervalle  $[t, t + T_e]$  et poser :

$$s(t + dt) = s(t) + 2\pi f_c \int_t^{t+T_e} (e(u) - s(u)) du. \quad (2)$$

On obtient un signal discréteisé avec une fréquence d'échantillonnage  $f_e = 1/T_e$ .

On note  $e_k = e(t_k)$  et  $s_k = s(t_k)$  les valeurs respectives des signaux d'entrée  $e$  et de sortie  $s$  à l'instant  $t_k$ .

**Q12.** Montrer qu'on obtient, après discréétisation et en appliquant la méthode des trapèzes pour calculer une valeur approchée de l'intégrale dans l'équation (2), la relation de récurrence suivante :

$$s_{k+1} = \frac{(1 - G) s_k + G (e_{k+1} + e_k)}{1 + G} \quad (3)$$

où  $G$  est une constante dont vous donnerez l'expression en fonction des paramètres  $f_c$  et  $f_e$ .

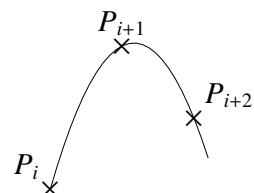
**Q13.** Écrire la fonction `filtrage(e, G)` d'argument `e` tableau du signal d'entrée, `G` la constante définie précédemment et qui renvoie le tableau de valeurs du signal filtré.

### II.2 - Méthode 1 - Récupération des pics

Le signal ayant été filtré, une première méthode consiste à récupérer les instants des pics, puis d'en déduire la fréquence cardiaque à partir du temps entre deux pics consécutifs.

Pour repérer un pic, nous allons prendre trois points de mesures consécutifs  $P_i$ ,  $P_{i+1}$  et  $P_{i+2}$ . On considère que l'on trouve un pic si :

- la valeur en  $P_{i+1}$  est supérieure à la valeur en  $P_i$  ;
- la valeur en  $P_{i+1}$  est supérieure à la valeur en  $P_{i+2}$  ;
- les valeurs des trois points sont supérieures à une valeur seuil minimale (par exemple 550 sur la figure 1) afin de ne trouver que des extrema " valides ".



Pour récupérer chaque pic, il est inutile de travailler sur l'ensemble du signal, mais il faut prendre suffisamment de points pour être certain d'avoir un pic. La fréquence cardiaque au repos usuelle étant de 60 pulsations par minute, soit un pic par seconde, on fait le choix de travailler, par sécurité, sur un intervalle de 3 s, ce qui correspond à des fenêtres de 150 points avec notre temps d'échantillonnage de 0,02 s. Cette partie du signal sera notée `extSignal`.

**Q14.** Compléter le test du `while` de la fonction `detectionPics(extSignal, seuil)` qui prend en argument `extSignal`, liste des données du signal partiel avec 150 points de mesure, ainsi que la valeur minimale `seuil` et qui renvoie l'indice du premier " pic " dans la partie `extSignal` du signal traité.

- Q15.** Commenter précisément la fonction `pulsationCardiaque(signal, Te, seuil)` qui prend en argument `signal`, liste des données du signal complet, `Te` le temps d'échantillonnage en seconde et `seuil` la valeur du seuil de détection. Expliciter ce que représente la valeur renvoyée.

### II.3 - Méthode 2 - Transformée de Fourier discrète

Une seconde méthode consiste à effectuer une étude fréquentielle du signal. Pour cela, on applique aux données une Transformation de Fourier Discrète (TFD).

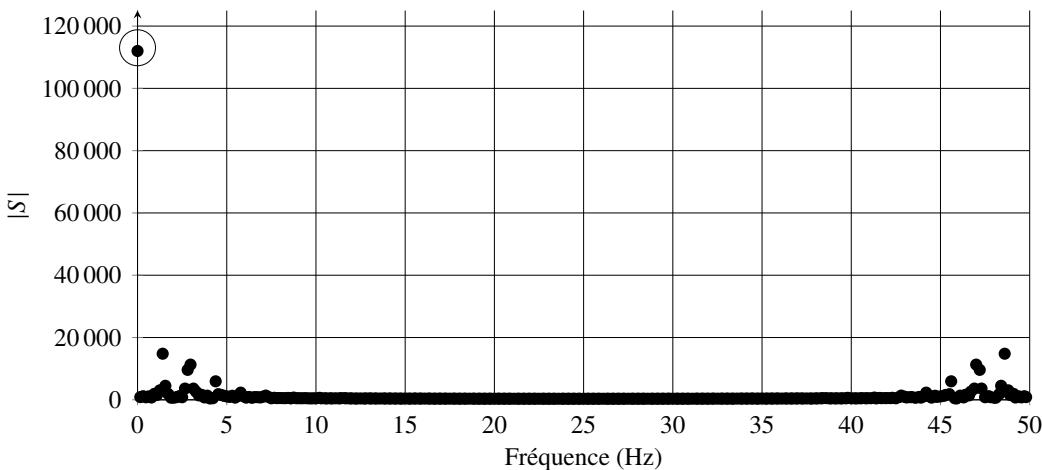
Pour un signal  $s$  de  $N$  échantillons obtenus à une fréquence  $f_e$ , la TFD du signal, notée  $S(k)$ , est donnée par :

$$S(k) = \sum_{n=0}^{N-1} s_n e^{-2\pi i k \frac{n}{N}} \quad \text{pour } 0 \leq k < N. \quad (4)$$

Les termes  $S(k)$  représentent une approximation de la transformée de Fourier du signal  $s$ , aux  $\frac{k}{N}$  fréquences  $f_k = \frac{k}{N} f_e$ .  $S(k)$  est un nombre complexe dont seul le module nous intéresse.

- Q16.** Écrire une fonction `TFD(signal, Te)` qui prend en argument `signal`, liste des données du signal et `Te` le temps d'échantillonnage et qui renvoie la liste des fréquences et la liste des modules de  $|S(k)|$ . La fonction `abs(z)` renvoie la valeur du module du complexe `z` (fonctionne sur un tableau). En Python, le complexe  $1+2i$  s'écrit `1+2j` et le complexe  $a+ib$  s'écrit `a+b*j`.

Pour le signal de la **figure 1**, nous obtenons le diagramme des fréquences de la **figure 2** réalisé avec un temps d'échantillonnage de 0,02 s.



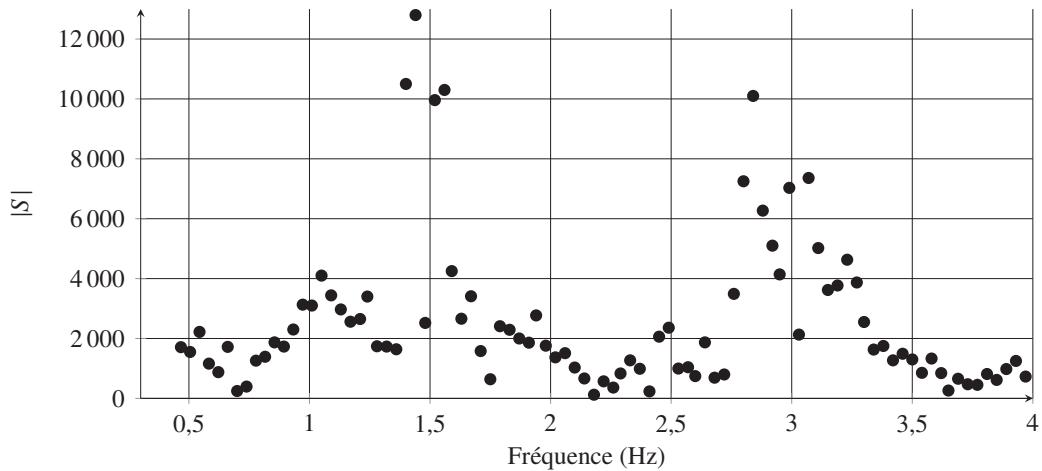
**Figure 2 - Spectre après application de la TFD**

- Q17.** Justifier la présence d'une valeur très élevée pour une fréquence nulle.

Le rythme cardiaque a une pulsation " normale " comprise entre 50 et 200 pulsations par minute. Nous allons prendre une marge en considérant l'amplitude de fréquences de pulsations dans l'intervalle [30, 220] pulsations par minute.

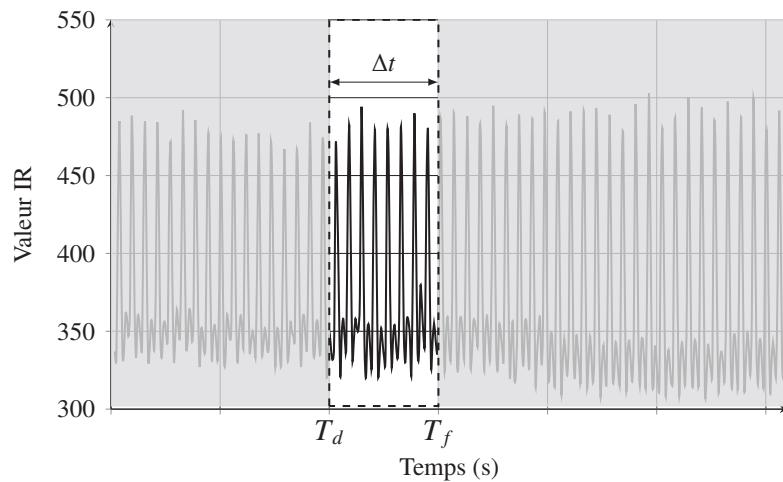
- Q18.** Modifier la fonction TFD précédente pour qu'elle ne calcule que les valeurs de  $S(k)$  comprises entre ces deux fréquences.

On obtient alors le spectre de la **figure 3**.



**Figure 3** - Spectre issu de la TFD pour  $f \in [0,2 \text{ Hz}, 4 \text{ Hz}]$

Pour déterminer la pulsation cardiaque toutes les secondes d'un signal, on calcule la fréquence cardiaque sur un intervalle de temps  $\Delta t = T_f - T_d$ , puis, on fait " glisser " cette fenêtre d'une seconde et on recalcule la fréquence cardiaque (**figure 4**).



**Figure 4** - Fenêtrage du signal

Pour appliquer ce principe, on propose le programme non commenté suivant où **Signal** contient la liste des valeurs de l'infrarouge durant le temps de l'activité.

```

52 | # Acquisition des données
53 | Temps, Signal=acquisition(Donnees)
54 |
55 | # fréquence d'échantillonnage
56 | G = 0.628
57 | fe = 50
58 | # Filtrage du signal
59 | SignalFiltre=filtrage(Signal,G)
60 |
61 | indDepart=0
62 | indFin=600
63 |
64 | HR=[]
65 | nbPoints=len(SignalFiltre)
66 | while indFin<nbPoints:
67 |     partSignal=SignalFiltre[indDepart:indFin]
68 |     freq,Sk=TFD(partSignal,fe)
69 |     HR.append(60*freq[Sk.index(max(Sk))])
70 |     indDepart=.....
71 |     indFin=.....

```

**Q19.** Donner l'intervalle  $\Delta t$  choisi par le programmeur. Expliquer la **ligne 69** et compléter les **lignes 70 et 71**.

Dans la méthode précédente, on calcule la fréquence cardiaque par application de la TFD à toutes les valeurs contenues dans une fenêtre temporelle de largeur  $\Delta t$  sans traitement sur ces valeurs. On parle de fenêtre rectangulaire. Cependant, l'application de la TFD sur ces valeurs peut induire, par des effets de lobes secondaires, des erreurs sur la fréquence maximale et donc sur la pulsation cardiaque relevée.

Pour corriger ce problème, on peut appliquer un coefficient multiplicateur en réduisant l'influence des valeurs lorsque l'on s'approche des bords de la fenêtre. Un fenêtrage classique est la fenêtre dite **de Hann** dont on donne la fonction suivante :

```

def Hann(extSignal, fe):
    t = arange(len(extSignal)) / fe
    hann = 1/2 - 1/2 * cos(2 * pi * t / t[-1])
    return list(extSignal * hann)

```

**Q20.** Parmi les quatre propositions du **Document Réponse**, entourer la fonction correspondant à la définition de la fenêtre dite de Hann.

**Q21.** Expliquer comment modifier le programme donné à la **Q19** pour prendre en compte la fenêtre dite de Hann précédente.

## Partie III - Partage des activités

Les fabricants de montres offrent la possibilité d'enregistrer les activités dans une base de données afin qu'elles soient accessibles sur PC ou tablettes et qu'elles puissent être partagées avec des amis. La base de données est constituée des tables suivantes :

**Table activite**

- Ida : entier permettant d'identifier l'activité
- Idm : entier correspondant à l'identifiant du membre " propriétaire " de l'activité
- Date : correspondant à la date (type date) de l'activité
- Type : chaîne de caractères correspondant au type d'activité " course ", " marche "...
- Distance : entier correspondant à la distance parcourue en mètre de l'activité
- Temps : entier correspondant à la durée en secondes de l'activité
- Fichier : contient le lien vers le fichier de l'activité

**Table amis** qui établit une relation entre 2 membres

- Idl : entier, identifiant du lien
- Membre1 : Idm d'un membre
- Membre2 : Idm d'un second membre

Il ne peut pas y avoir de doublons dans la table **amis** : si A est le membre 1 et B le membre 2 d'une relation d'amitiés, la ligne membre 1=B et membre 2=A n'existe pas dans la table.

On considère pour les questions suivantes un membre dont l'identifiant est 1 (*Idm = 1*).

**Q22.** Écrire une requête SQL permettant de récupérer la liste des identifiants des activités du membre dont l'identifiant est 1.

**Q23.** Écrire une requête SQL permettant de donner la date, la distance parcourue et la vitesse moyenne en km/h des activités de type " course " du membre dont l'identifiant est 1.

On donne la requête suivante :

```

SELECT activite.Ida FROM activite
JOIN (SELECT membre1 AS idam1 FROM amis WHERE membre2=1
      UNION
      SELECT membre2 AS idam1 FROM amis WHERE membre1=1 ) AS amis1
ON activite.idm=amis1.idam1
WHERE Type='marche'
```

**Q24.** Décrire chaque instruction de la requête et expliquer ce qu'elle renvoie.

**FIN**

## ANNEXE

### Annexe 1 - Norme NMEA 0183

La National Marine Electronics Association (NMEA) est une association américaine de fabricant de matériels électroniques maritimes. Elle a défini un protocole de communication entre équipements marins dont les GPS. Ainsi, un récepteur GPS envoie une série de phrases (trames) sur une communication série. Chaque trame est constituée de chaînes de caractères commençant par '\$' terminées par un retour chariot. Les caractères constituant la trame ont un code ASCII compris entre 20 et 127, mais sont codés sur un octet.

Un récepteur GPS de type NEO-6M transmet les trames suivantes toutes les secondes :

```
$GPGLL,4754.68986,N,00154.69154,E,092827.00,A,A*6B
$GPRMC,092828.00,A,4754.68988,N,00154.69147,E,0.444,,070619,,,A*7C
$GPVTG,,T,M,0.444,N,0.822,K,A*2F
$GPGGA,092828.00,4754.68988,N,00154.69147,E,1,04,2.61,84.3,M,46.5,M,*64
$GPGSA,A,3,25,29,24,32,,,,,,3.99,2.61,3.02*0E
$GPGSV,3,1,11,02,32,075,,04,,22,06,15,037,,12,47,072,16*4A
$GPGSV,3,2,11,14,38,276,11,24,18,141,19,25,80,353,24,29,56,196,30*7F
$GPGSV,3,3,11,31,34,306,25,32,35,250,36,33,32,202,27*4F
```

Pour une montre multisport, nous n'utiliserons que la trame GGA qui contient les informations suivantes : \$GPGGA,092828.00,4754.68988,N,00154.69147,E,1,04,2.61,84.3,M,46.5,M,\*64

- GP : Type de système (GP : GPS, GA : Galileo, GL : Glonass, ...)
- GGA : Type de trame
- 092828.00 : Horaire UTC - 09 h 28 min 28,00 s
- 4754.68988 : Latitude : 47°54,68988'
- N : latitude Nord
- 00154.69147 : Longitude 001°54,69147'
- E : Longitude Est
- 1 : Type de positionnement
- 04 : Nombre de satellites
- 2.61 : Coefficient de précision
- 84.3,M : Altitude par rapport au niveau moyen des océans
- 46.5,M : Correction de la hauteur de la géoïde par rapport à l'ellipsoïde WGS84
- ,,: Champ toujours vide
- \*64 : Somme de contrôle

### Annexe 2 - Caractères et chaînes de caractères en Python

#### Fonctions

<code>ord(char1)</code>	renvoie la valeur entière correspondant à <code>char1</code> dans la table ASCII	<code>ord('A')</code> → 65
<code>chr(int1)</code>	renvoie le caractère correspondant à <code>int1</code> dans la table ASCII	<code>chr(65)</code> → 'A'

#### Caractères d'échappement

- '\n' : saut de ligne
- '\t' : tabulation

Exemple :

```
>>> chaine = 'Ceci est un test.\nOn peut mettre une tabulation entre a et
      b : a\tb.'
>>> print(chaine)
Ceci est un test.
On peut mettre une tabulation entre a et b : a      b.
```

## Fonctions sur les chaînes de caractères

Une chaîne de caractères est gérée comme une liste.

Exemple : chaine = "Concours Commun INP \n".

chaine[i]	Renvoie le ième terme de la chaine	chaine[5] → u
chaine[i:j]	Renvoie la chaîne comprise entre le ième et le jème-1 terme	chaine[9:15] → Commun
chaine.split()	Découpe la chaîne au caractère désigné, par défaut ‘espace’.	chaine.split('C') → [", 'oncours ', 'ommun INP \n']
bin(int1)	Convertit un entier en la chaîne de caractères de son expression binaire, précédée de ‘0b’ pour indiquer la base binaire	bin(12) → '0b1100'
hex(int1)	Convertit un entier en la chaîne de caractères de son expression hexadécimale, précédée de ‘0x’ pour indiquer la base hexadécimale	hex(12) → '0xc' hex(0b01010000) → '0x50'

## Annexe 3 - Ou Exclusif

La fonction Ou Exclusif renvoie le ou exclusif bit à bit de deux entiers.

Exemple : 10 ^ 15 renvoie 5

ou en représentation binaire : bin(0b1010 ^ 0b1111) renvoie '0b0101'.

## Annexe 4 - Fonctions sur les tableaux et les listes

**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.

	Python
tableau à une dimension	L=[1,2,3] (liste) v=array([1,2,3]) (vecteur)
créer un vecteur rempli de 0 de taille n	zeros(n)
accéder à un élément	v[0] renvoie 1 (L[0] également)
ajouter un élément	L.append(5) uniquement sur les listes
tableau à deux dimensions (matrice)	M=array([[1,2,3],[3,4,5]])
accéder à un élément	M[1,2] donne 5
extraire une portion de tableau (2 premières colonnes)	M[:,0:2]
extraire la colonne i	M[:,i]
extraire la ligne i	M[i,:]
tableau de 0 (2 lignes, 3 colonnes)	zeros((2,3))
Transformer une ligne en colonne	a = array([1,2,3]) b = a.reshape(3,1) print(b) >>> array([[1],[2],[3]])
produit matrice-vecteur	a = array([[1,2,3],[4,5,6],[7,8,9]]) b = array([1,2,3]) print(a.dot(b)) >>> array([14,32,50])

 <b>CONCOURS COMMUN</b> <b>INP</b>	<b>Numéro d'inscription</b> <input style="width: 50px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <b>Numéro de table</b> <input style="width: 50px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <b>Né(e) le</b> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/>	<b>Nom :</b> _____  <b>Prénom :</b> _____
	<b>Filière :</b> <b>PSI</b>	<b>Session :</b> <b>2021</b>
	<b>Épreuve de : INFORMATIQUE</b>	
<b>Emplacement QR Code</b>		
<b>Consignes</b>	<ul style="list-style-type: none"> <li>• Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer</li> <li>• Rédiger avec un stylo non effaçable bleu ou noir</li> <li>• Ne rien écrire dans les marges (gauche et droite)</li> <li>• Numérotter chaque page (cadre en bas à droite)</li> <li>• Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre</li> </ul>	

PSI5IN

## **DOCUMENT RÉPONSE**

**Ce Document Réponse doit être rendu dans son intégralité.**

## Q1 - Expliquer les lignes 2 à 8

```
1 | def recupererGPGGA(chaine):
2 |     chaineDecoupe=chaine . split( '\n' )
3 |     indGGA = indice_GPGGA(chaineDecoupe)
4 |     ligne = []
5 |     if indGGA != len(chaineDecoupe):
6 |         ligne = chaineDecoupe[indGGA][1:]
7 |         ligne = ligne . split( ',' )
8 |         ligne . pop(len(ligne)-2)
9 |     return ligne
```

## Q2 - Écriture de la fonction indice\_GPGGA(listeChaine)

## **NE RIEN ÉCRIRE DANS CE CADRE**

**Q3** - Instruction permettant de créer listeGPGGABrute

\_\_\_\_\_

#### **Q4 - Écriture de la fonction testPresence(trame)**

### **Q5 - Écriture de la fonction testPrecision(trame)**

**Q6** - Valeur de csHex et valeur renvoyée par `testSC(['GPG', '*A0'])`

```

def testCS ( trame ) :
    somme=trame [ -1 ][ 1 : ]
    cs=0
    for chaine in trame [ : -1 ]:
        for ch in chaine :
            cs=cs^ord(ch)
    csHex=hex(cs)[2:]
    return csHex==somme

```

#### **Q7 - Écriture de la fonction convHoraire(chHoraire)**

**Q8** - Écriture de la fonction `convAngle(chAngle, chCard)`

### **Q9 - Complétez la fonction recupererDonnees(trame)**

```
def recupererDonnees(trame):  
    Horaire = .....  
    Latitude = .....  
    Longitude = .....  
    Altitude = .....  
  
return .....
```

**Q10** - Espace mémoire pour stocker une activité d'une heure. Espace mémoire pour stocker 200 h d'activités

**Q11** - Solution pour ne pas dépasser la quantité de mémoire disponible

## **Q12 - Démonstration de la relation de récurrence**

### **Q13 - Écriture de la fonction filtre(e, G)**

 <b>Numéro d'inscription</b> <input style="width: 100px; height: 40px; border: 1px solid black; margin-bottom: 10px;" type="text"/> <b>Numéro de table</b> <input style="width: 100px; height: 40px; border: 1px solid black; margin-bottom: 10px;" type="text"/> <b>Né(e) le</b> <input style="width: 25px; height: 25px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 25px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 25px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 25px; border: 1px solid black; margin-right: 10px;" type="text"/>	<b>Nom :</b> _____  <b>Prénom :</b> _____
<b>Filière :</b> <b>PSI</b>	<b>Session :</b> <b>2021</b>
<b>Emplacement QR Code</b>  <b>Épreuve de : INFORMATIQUE</b>	
<b>Consignes</b>	<ul style="list-style-type: none"> <li>• Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer</li> <li>• Rédiger avec un stylo non effaçable bleu ou noir</li> <li>• Ne rien écrire dans les marges (gauche et droite)</li> <li>• Numérotter chaque page (cadre en bas à droite)</li> <li>• Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre</li> </ul>

PSI5IN

**Q14** - Complétez le test de la fonction `detectionPics(extSignal, seuil)`

```
def detectionPics(extSignal , seuil):
    P0,P1,P2=extSignal [:3]
    i=1
    while i<len(extSignal)-2 and .....:
        P0,P1=P1,P2
        P2=extSignal [i+2]
        i=i+1
    return i
```

**Q15** - Commentaire de la fonction pulsationCardiaque(signal,Te,seuil)

```
1 | def pulsationCardiaque( signal ,Te , seuil ):  
2 |     pics = []  
3 |     deb = 0  
4 |     nbPoints = int(3/Te)  
5 |     while deb+nbPoints < len( signal ):  
6 |         deb = deb+detectionPics( signal [ deb :deb+nbPoints ] , seuil )  
7 |         pics .append(deb)  
8 |     periode_moy = (pics [-1]-pics [0])*Te / (len( pics )-1)  
9 | return 60/periode_moy
```

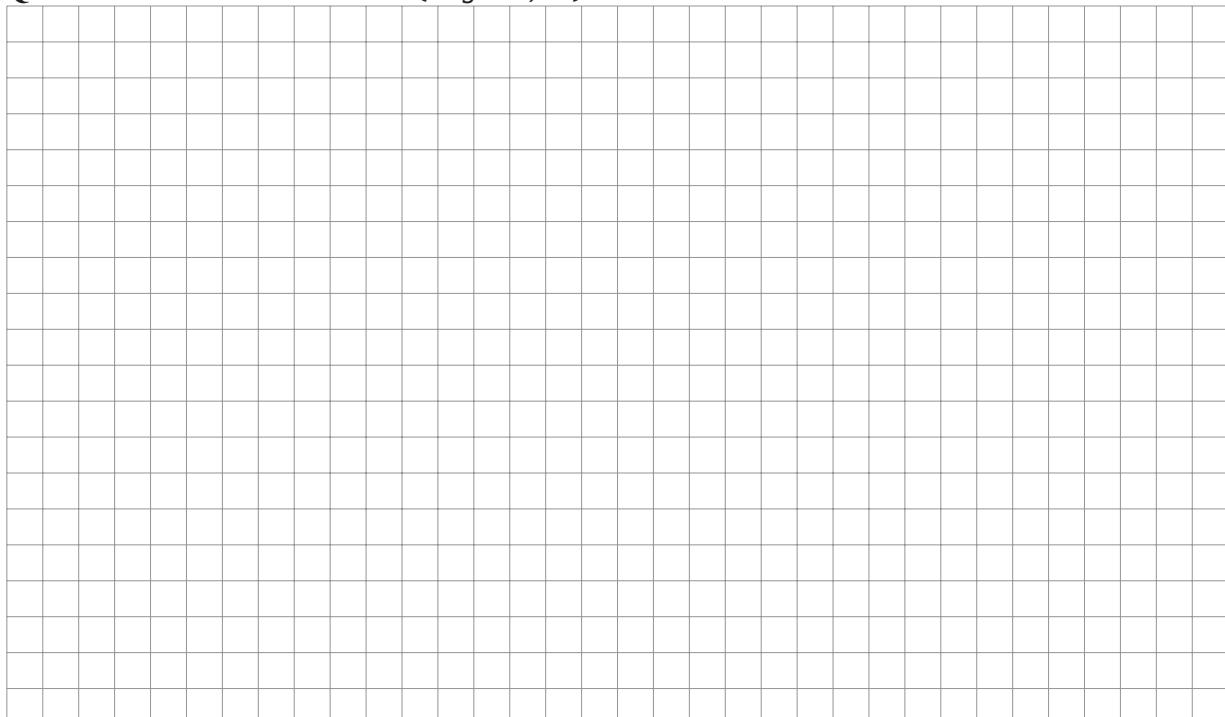


(espace libre page suivante)

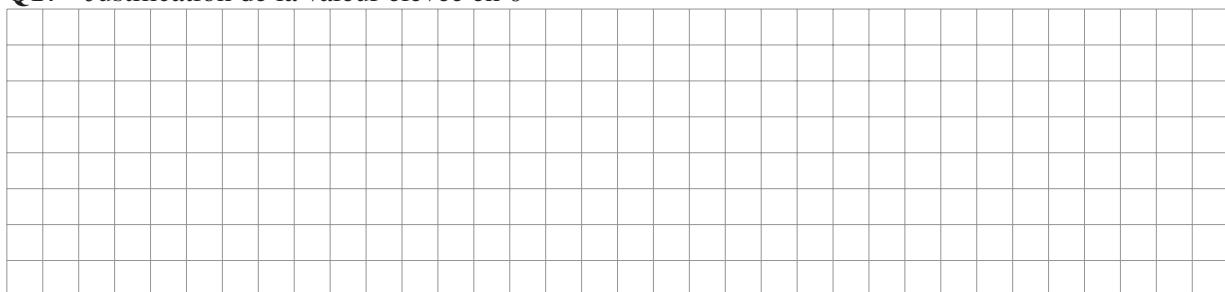
NE RIEN ÉCRIRE DANS CE CADRE



**Q16** - Écriture de la fonction TFD(signal, te)



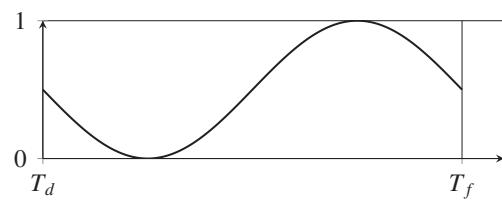
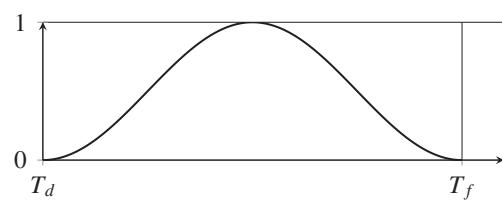
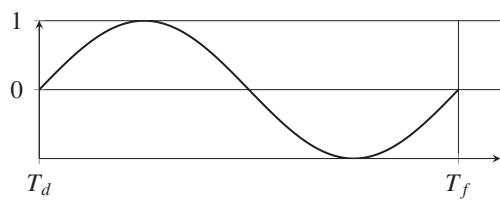
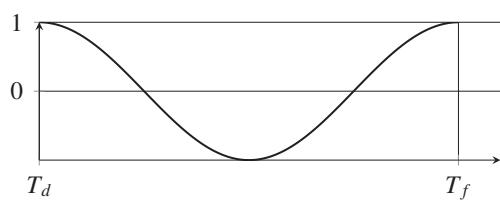
**Q17** - Justification de la valeur élevée en 0



**Q18** - Modification de la fonction TFD(signal, te)

**Q19** - Valeur de  $\Delta t$ . Commentaire de la ligne 69 et complémentation des lignes 70 et 71

**Q20** - Choix de la courbe correspondant à ce que renvoie la fonction hann(extSignal, fe)



**Q21** - Explication comment modifier le programme de la **Q19**

**Q22** - Requête SQL permettant de récupérer la liste des identifiants des activités du membre dont l'identifiant est 1

**Q23** - Requête SQL permettant de récupérer la date, le kilométrage et la vitesse moyenne des activités de type "course" du membre dont l'identifiant est 1

## **Q24** - Explication de la requête SQL. Résultat retourné

SESSION 2021

TSI5IN



## ÉPREUVE SPÉCIFIQUE - FILIÈRE 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.

#### RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

**Le sujet est composé de deux parties indépendantes.**

Le sujet comporte :

- le texte du sujet : page 1 à page 6 ;
- le Document Réponse : page 1 à page 11.

**Le Document Réponse doit être rendu dans son intégralité.**

## Optimisation de rendement d'une entreprise de livraison

Une entreprise de livraison dispose de plusieurs locaux en France et chacun possède plusieurs camions de livraisons. Celle-ci souhaite optimiser le chargement de ses camions pour diminuer ses frais de fonctionnement.

### Partie I - Optimisation du chargement

Chaque camion de l'entreprise peut charger une cargaison jusqu'à un poids maximal noté  $P_{max}$ . L'entreprise dispose de différentes informations provenant de ses clients :

- le poids de chaque produit  $p_i$  (chaque client propose un seul produit) ;
- la valeur  $v_i$  associée au transport de chaque produit : c'est-à-dire l'argent gagné par l'entreprise si elle réalise le transport de ce produit.

En considérant que l'entreprise dispose de  $n$  clients, l'entreprise cherche donc à trouver une liste d'indices notée  $I$  contenue dans  $\{1, \dots, n\}$  telle que :

$$\sum_{i \in I} p_i \leq P_{max} \quad (\text{respect du poids maximal}) \quad (1)$$

et

$$\sum_{i \in I} v_i \text{ soit maximal} \quad (\text{optimisation du profit pour l'entreprise}). \quad (2)$$

Dans toute la suite, les poids seront donnés en centaines de kilogrammes et les valeurs en centaines d'euros.

#### I.1 - Un exemple

Dans cette sous-partie, on suppose que  $n = 4$  et que  $P_{max} = 8$  centaines de kilogrammes. On stocke alors les différentes informations dans trois listes :

- $Pr$  est la liste des produits proposés par les clients numérotés de 1 à 4 :  $Pr = [1, 2, 3, 4]$  ;
- $P$  est la liste des poids associés :  $P = [3, 2, 1, 4]$  ;
- $V$  est la liste des valeurs associées :  $V = [4, 3, 1, 9]$ .

Par exemple, le produit 2 a un poids de deux centaines de kilogrammes et une valeur de trois centaines d'euros.

Les questions **Q1**, **Q2** et **Q3** se traitent à l'aide de calculs simples, à faire à la main.

- Q1.** Expliquer pourquoi une cargaison constituée d'un, de deux ou de quatre produits ne répond pas au problème posé, c'est-à-dire ne maximise pas le profit fait par l'entreprise en respectant la condition donnée sur le poids maximal.
- Q2.** Donner toutes les cargaisons de trois produits respectant le poids maximal. On donnera à chaque fois le profit fait par l'entreprise.
- Q3.** Quelle est la cargaison maximisant le profit de l'entreprise ? Que vaut le profit dans ce cas ?

## I.2 - Une méthode intuitive pour la résolution du problème

On garde les notations de la sous-partie précédente dans le cas général :

- $\text{Pr}$  est la liste des produits (numérotés de 1 à  $n$  inclus) ;
- $P = [p_1, \dots, p_n]$  est la liste des poids associés aux produits ;
- $V = [v_1, \dots, v_n]$  est la liste des valeurs associées aux produits.

**Q4.** Définir une fonction `ListeProduits` ayant pour argument un entier naturel non nul  $n$  et renvoyant la liste  $\text{Pr}$ .

Une méthode intuitive pour tenter d'optimiser le profit de l'entreprise est la suivante : on calcule les ratios  $\frac{v_i}{p_i}$ , puis on trie les objets par ordre décroissant suivant ces valeurs. Les produits sont alors classés par rentabilité : le premier produit devient le plus rentable " au poids " et ainsi de suite. On ajoute progressivement chaque produit dans la cargaison, dans cet ordre, sans dépasser la limite du poids maximal.

**Q5.** Définir une fonction `Ratio` ayant pour arguments deux listes  $P, V$  où  $P$  correspond à la liste des poids et  $V$  correspond à la liste des valeurs, renvoyant la liste des ratios  $\frac{v_i}{p_i}$ .

La fonction suivante est associée à une méthode de tri :

```

1 def Tri(L):
2     ''' L est une liste de nombres réels '''
3     for i in range(1,len(L)):
4         x=L[i]
5         j=i
6         while j>0 and x<L[j-1]:
7             L[j]=L[j-1]
8             j=j-1
9         L[j]=x
10    return L

```

**Q6.** On exécute `Tri(L)` avec  $L=[3, 5, 2, 1]$ . Combien y a t-il d'itérations de la boucle `for`? Donner la valeur de  $L$  à la fin de chaque itération de la boucle `for`.

**Q7.** Ce tri fonctionnerait-il pour une chaîne de caractères dont chaque caractère est un entier ? Justifier.

**Q8.** Quelle est la méthode de tri utilisée dans la fonction `Tri`? Donner sa complexité (en nombre de comparaisons) dans le pire des cas et dans le meilleur des cas. *On justifiera soigneusement les complexités données.*

**Q9.** Définir une fonction `Inverse` ayant pour argument une liste de nombres réels  $L$  et renvoyant l'inverse de celle-ci. Par exemple, l'inverse de  $[1, 5, 3, 4]$  est  $[4, 3, 5, 1]$ .

L'utilisation de  $L[::-1]$  n'est pas autorisée.

**Q10.** On souhaite trier une liste de poids  $P$  et une liste de valeurs  $V$  associées à une liste de produits en suivant l'ordre décroissant de la liste des ratios  $\frac{v_i}{p_i}$ . Justifier que les fonctions `Ratio`, `Tri` et `Inverse` ne permettent pas de répondre simplement au problème posé.

- Q11.** Écrire, à l'aide des fonctions `Ratio` et `Inverse`, une fonction `Tri2` ayant pour arguments une liste de poids  $P$  et une liste de valeurs  $V$  associées à une liste de produits. Cette fonction renverra les listes de poids et de valeurs triées par ordre décroissant de la liste des ratios.
- Q12.** Compléter la définition de la fonction `Vmax` ayant pour arguments les listes de poids  $P$  et de valeurs  $V$  et le poids maximal  $P_{max}$  du chargement et renvoyant la valeur maximale du profit de l'entreprise en suivant la méthode proposée.
- Q13.** On souhaite appliquer cette méthode en utilisant les listes de poids et de valeurs de la sous-partie **I.1**. Donner la liste des ratios, les listes de poids et de valeurs obtenues à l'aide de la fonction `Tri2` ainsi que le profit obtenu. Commenter le résultat.

### I.3 - Une méthode récursive

Nous gardons les notations du cas général de la sous-partie **I.2** :  $\text{Pr}$ ,  $P$  et  $V$ . On considère pour simplifier que les poids des produits sont des entiers, ainsi que  $P_{max}$ .

Nous introduisons une méthode récursive pour résoudre le problème d'optimisation :

- pour chacun des produits, deux choix sont possibles : il fait partie de la cargaison ou non ;
- la récursivité s'effectuera sur la liste des indices de  $\text{Pr}$  : le premier appel de la fonction se fera en utilisant l'indice  $n$ , puis l'indice  $n - 1$  et ainsi de suite jusqu'à l'indice 0 (correspondant au cas où il n'y a plus de produits) ;
- pour  $i \in \{0, 1, \dots, n\}$  et  $\omega \in \{0, 1, \dots, P_{max}\}$ , on note  $S(i, \omega)$  la valeur maximale cumulée des produits que l'on peut placer dans un camion d'une capacité maximale (en poids) de  $\omega$  avec la liste constituée des  $i$  premiers produits de  $\text{Pr}$ .

On pose alors la relation de récursivité suivante :

$$S(i, \omega) = \begin{cases} 0 & \text{si } i = 0 \\ S(i - 1, \omega) & \text{si } i > 0 \text{ et } p_i > \omega \\ \max(S(i - 1, \omega), v_i + S(i - 1, \omega - p_i)) & \text{si } i > 0 \text{ et } p_i \leq \omega. \end{cases} \quad (3)$$

- Q14.** Justifier les relations précédentes dans les trois cas.
- Q15.** Justifier la terminaison de l'algorithme associé à la relation de récursivité précédente, sachant que la première valeur donnée pour  $i$  sera  $n$  et la première valeur pour  $\omega$  sera  $P_{max}$ .
- Q16.** Définir une fonction `Max` ayant pour arguments deux réels et renvoyant le maximum parmi ces deux valeurs. Il est interdit d'utiliser la fonction `max` prédéfinie dans Python.
- Q17.** En vous basant sur la relation (3), compléter la définition de la fonction récursive `recur` ayant pour arguments les listes de poids et de valeurs  $P$  et  $V$ , un indice  $i$ , un poids  $\omega$ , et renvoyant  $S(i, \omega)$ .
- Q18.** Donner une série d'instructions utilisant la fonction `recur` et permettant de déterminer le profit de la sous-partie **I.1**.

### I.4 - Amélioration de la méthode récursive

On souhaite améliorer la méthode récursive de la sous-partie **I.3**. Nous allons procéder en mémorisant des calculs déjà effectués. Voici le principe : nous allons stocker les valeurs  $S(i, \omega)$  dans un tableau `Memoire`, de taille  $(n + 1) \times (P_{max} + 1)$ , initialisé au départ avec des éléments tous égaux à  $-1$ .

Si la valeur de  $S(i, \omega)$  a déjà été calculée, l’élément d’indice  $(i, \omega)$  de ce tableau **Memoire** ne sera plus égal à  $-1$  : on renverra donc directement la valeur. Sinon, on la calculera en suivant le principe de la sous-partie **I.3** et on la stockera dans le tableau avant de la renvoyer.

Nous faisons le choix de représenter les tableaux comme des listes de listes.

**Q19.** Donner l’instruction permettant de créer le tableau **Memoire** initialisé avec des coefficients égaux à  $-1$ , en supposant **Pmax** et **n** connus.

**Q20.** Compléter la fonction **recur2** en suivant le principe expliqué et permettant d’améliorer la fonction **recur**. La variable **Memoire** sera utilisée comme une variable globale.

Avec les données suivantes :

- **P** = [5, 3, 3, 3];
- **V** = [4, 3, 1, 1];
- **P<sub>max</sub>** = 8;
- **Memoire** un tableau de taille  $5 \times 9$ ;

et en exécutant **recur2(P, V, len(P), Pmax, Memoire)**, on obtient alors la valeur 7.

## Partie II - Données liées aux livraisons conservées par l’entreprise

À chaque livraison, l’entreprise stocke des données relatives à celle-ci. L’entreprise dispose de 20 locaux, numérotés de 1 à 20, disposant chacun d’un certain nombre de camions. Pour faciliter ses livraisons, l’entreprise découpe la France en 30 zones et associe à chaque local, trois zones possibles de livraisons.

Ces données sont enregistrées dans une base de données contenant trois tables :

La table **livraison** constituée des champs suivants :

- *date* : date de la livraison au format "jj-mm-aaaa" (chaine de caractères);
- *heure* : heure de la livraison au format : "hh-mm-ss" (chaine de caractères);
- *id\_client* : identifiant du client recevant la livraison (entier);
- *id\_local* : identifiant du local de l’entreprise (entier compris entre 1 et 20).

La table **client** constituée des champs suivants :

- *id* : identifiant du client (entier);
- *zone* : entier compris entre 1 et 30.

La table **local** constituée des champs suivants :

- *id* : identifiant du local (entier compris entre 1 et 20);
- *zone1* : entier;
- *zone2* : entier;
- *zone3* : entier.

**Q21.** Donner une clé primaire pour la table **livraison**.

**Q22.** Écrire une requête SQL permettant d'obtenir les identifiants des clients livrés le 10 janvier 2021.

**Q23.** Écrire une requête SQL permettant de récupérer les dates et les heures de toutes les livraisons ayant eu lieu dans la zone 5 le 2 mars 2021.

**Q24.** Écrire une requête SQL permettant de compter le nombre de livraisons effectuées le 3 février 2021 par des camions dont les locaux ne livrent que dans des zones possibles inférieures ou égales à dix.

Chaque identifiant de client est stocké par l'entreprise en codage binaire (avec 8 bits). Par exemple, 00010111 est associé au client dont l'identifiant est le numéro 23.

**Q25.** Donner le codage associé au client 39.

Afin de retrouver l'identifiant de chaque client à l'aide de son code binaire, la fonction suivante est proposée :

```
1 def Identifiant(Bin):
2     '''Bin est une chaîne de caractères
3         constituée de 0 et 1'''
4     S=0
5     for i in range(len(Bin)):
6         S=S+Bin[i]*2**len(Bin)-i
7     return S
```

**Q26.** Trouver les deux erreurs dans le code de la fonction précédente.

On suppose maintenant la fonction précédente corrigée. La ligne 6 pose un problème de complexité : à chaque itération de boucle, la puissance de 2 est recalculée entièrement.

**Q27.** Écrire une fonction **Identifiant2**, qui donne le même résultat que la fonction **Identifiant** avec une meilleure complexité. Le nombre de multiplications devra être linéaire suivant la longueur de Bin.

**Q28.** Si l'entreprise souhaite aussi stocker la zone de chaque client en codage binaire, donner le nombre de bits minimal nécessaire.

**FIN**







**Numéro  
d'inscription**

**Numéro de table**

**Né(e) le**

**Nom :** \_\_\_\_\_

**Prénom :** \_\_\_\_\_

Filière : TSI

Session : 2021

## **Épreuve de : INFORMATIQUE**

## **Consignes**

- Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer
  - Rédiger avec un stylo non effaçable bleu ou noir
  - Ne rien écrire dans les marges (gauche et droite)
  - Numérotter chaque page (cadre en bas à droite)
  - Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre

TSI5IN

## DOCUMENT RÉPONSE

**Ce Document Réponse doit être rendu dans son intégralité.**

# Optimisation de rendement d'une entreprise de livraison

Q1

NE RIEN ÉCRIRE DANS CE CADRE

**Q2**

.....  
.....  
.....  
.....  
.....  
.....

**Q3**

Cargaison maximisant le profit : .....

Profit maximal : .....

**Q4**

```
def ListeProduits(n) :
```

**Q5**

```
def Ratio(P,V) :
```

**Q6**

Nombre d'itérations de la boucle for : .....

Valeur de L après chaque itération :

.....  
.....  
.....  
.....

**Q7**

.....  
.....

**Q8**

Méthode de tri utilisée : .....

Complexité dans le pire des cas : .....

Complexité dans le meilleur des cas : .....

Justifications : .....

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

**Q9**

```
def Inverse(L) :
```

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

**Q10**

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

 CONCOURS COMMUN INP	Numéro d'inscription	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>	Nom :	
	Numéro de table	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>	Prénom :	
	Né(e) le	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>		
Emplacement QR Code	Filière : TSI		Session : 2021	
	<b>Épreuve de : INFORMATIQUE</b>			
<b>Consignes</b>	<ul style="list-style-type: none"><li>• Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer</li><li>• Rédiger avec un stylo non effaçable bleu ou noir</li><li>• Ne rien écrire dans les marges (gauche et droite)</li><li>• Numérotter chaque page (cadre en bas à droite)</li><li>• Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre</li></ul>			

TSI5IN

**Q11**

def Tri2(P,V) :

NE RIEN ÉCRIRE DANS CE CADRE

**Q12**

```
def Vmax(P,V,Pmax) :  
    P2,V2=Tri2(P,V)  
    SP=0  
    SV=0  
    i=0  
    while .....  
        SP=SP+P2[i]  
        SV=SV+V2[i]  
        i=i+1  
    return .....
```

**Q13**

Profit obtenu avec cette méthode :

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

Commentaire :

.....  
.....

**Q14**

Justification pour  $i = 0$  :

.....  
.....

Justification pour  $i > 0$  et  $p_i > \omega$  :

.....  
.....  
.....

Justification pour  $i > 0$  et  $p_i \leq \omega$  :

.....  
.....  
.....  
.....  
.....

**Q15**

Justification de la terminaison :

.....  
.....  
.....  
.....

**Q16**

```
def Max(a,b) :
```

**Q17**

```
def recur(P,V,i,w) :
    if i==0:
        return .....
    if P[i-1]>w:
        return recur(P,V,i-1,w)
    else :
        .....
```

**Q18**

```
.....  
.....  
.....  
.....
```

**Q19**

```
.....  
.....  
.....
```

 <b>CONCOURS COMMUN INP</b>	<b>Numéro d'inscription</b> <input style="width: 100px; height: 20px; border: 1px solid black; margin-bottom: 5px;" type="text"/>	<b>Nom :</b> _____
	<b>Numéro de table</b> <input style="width: 100px; height: 20px; border: 1px solid black; margin-bottom: 5px;" type="text"/>	<b>Prénom :</b> _____
	<b>Né(e) le</b> <input style="width: 100px; height: 20px; border: 1px solid black; margin-bottom: 5px;" type="text"/>	
<b>Emplacement QR Code</b>	<b>Filière : TSI</b>	<b>Session : 2021</b>
<b>Épreuve de : INFORMATIQUE</b>		
<p><b>Consignes</b></p> <ul style="list-style-type: none"> <li>• Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer</li> <li>• Rédiger avec un stylo non effaçable bleu ou noir</li> <li>• Ne rien écrire dans les marges (gauche et droite)</li> <li>• Numérotter chaque page (cadre en bas à droite)</li> <li>• Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre</li> </ul>		

TSI5IN

**Q20**

```

def recur2(P,V,i,w,Memoire) :
    if i==0:
        return 0
    if Memoire[i][w]>-1:
        return .....
    if P[i-1]>w:
        Memoire[i][w]=recur2(P,V,i-1,w,Memoire)
        return Memoire[i][w]
    else:
        if Memoire[i-1][w]==-1:
            Memoire[i-1][w]= .....
        if .....
            Memoire[i-1][w-P[i-1]]=recur2(P,V,i-1,w-P[i-1],Memoire)
    a=max(Memoire[i-1][w],V[i-1]+Memoire[i-1][w-P[i-1]])
    Memoire[i][w]=a
    return .....

```

**Q21**

.....  
.....

NE RIEN ÉCRIRE DANS CE CADRE

Q22

.....  
.....  
.....

Q23

.....  
.....  
.....

Q24

.....  
.....  
.....  
.....

Q25

.....  
.....  
.....  
.....  
.....

Q26

.....  
.....  
.....  
.....

Q27

```
def Identifiant2(Bin) :
```

Q28

.....  
.....  
.....  
.....  
.....





## ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

### ÉPREUVE SPÉCIFIQUE - FILIÈRE PC

#### MODÉLISATION DE SYSTÈMES 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.

#### RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont autorisées.

Le sujet est composé de trois parties indépendantes.

Sujet : page 1 à page 12.

Annexe : page 13 à page 16.

# Modélisation de la fuite de matière d'un réservoir rempli de dioxyde de carbone gazeux

## Présentation générale

Ce sujet repose sur l'étude théorique et numérique d'une fuite de matière au sein d'une cuve contenant du dioxyde de carbone ( $\text{CO}_2$ ) gazeux. Il est constitué de **trois parties indépendantes**.

- La première partie est dédiée à l'établissement du modèle thermodynamique du phénomène de fuite de matière contenue dans une cuve à travers un orifice.
- La seconde partie est consacrée à l'étude numérique du problème ; la relation entre la capacité thermique molaire  $C_{P,m}$  du  $\text{CO}_2$  gaz parfait et la température  $T$  est admise.
- La troisième partie est consacrée à l'étude de deux modèles pour décrire la relation entre  $C_{P,m}$  et  $T$  dont les coefficients sont déduits de mesures expérimentales.

## Traitements numériques et calcul scientifique réalisés à partir d'un programme écrit en langage Python

- Les programmes demandés au candidat seront réalisés dans le langage Python.
- On veillera à apporter les commentaires facilitant la compréhension du programme et à utiliser des noms de variables explicites.
- **Il est demandé de répondre précisément aux questions posées (par exemple, on écrira une fonction uniquement lorsque cela est explicitement demandé).**
- *Une annexe décrivant quelques éléments de langage Python utiles pour ce sujet est fournie en page 13.*

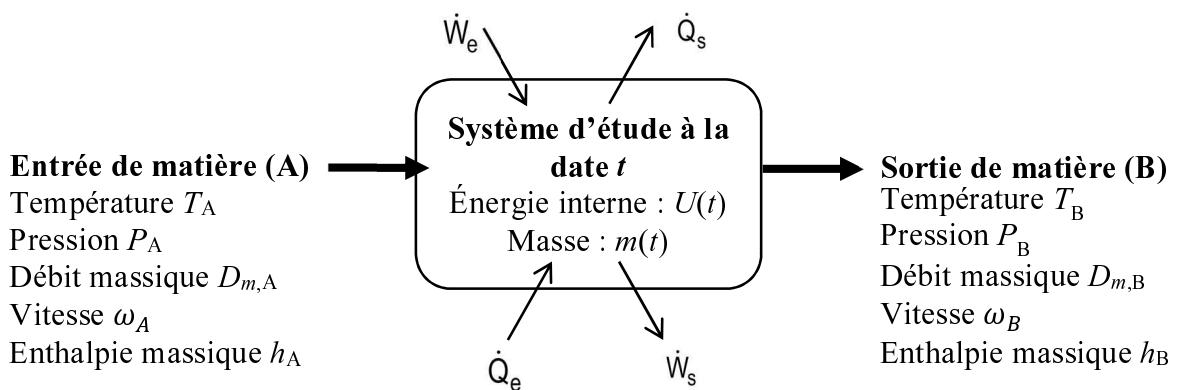
## Précisions concernant les notations utilisées

Symbol	Nom	Unité
$C_{P,m}$	Capacité thermique <b>molaire</b> à pression constante	$\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
$c_P$	Capacité thermique <b>massique</b> à pression constante	$\text{J}\cdot\text{K}^{-1}\cdot\text{kg}^{-1}$
$D_m$	Débit <b>massique</b>	$\text{kg}\cdot\text{s}^{-1}$
$h$	Enthalpie <b>massique</b>	$\text{J}\cdot\text{kg}^{-1}$
$M$	Masse <b>molaire</b>	$\text{kg}\cdot\text{mol}^{-1}$
$m$	Masse	kg
$P$	Pression	Pa
$\dot{Q} = \delta Q/dt$	Puissance thermique	W
$R$	Constante des gaz parfaits	$\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
$T$	Température	K
$T_c$	Température critique	K
$U$	Énergie interne	J
$u$	Énergie interne <b>massique</b>	$\text{J}\cdot\text{kg}^{-1}$
$V$	Volume	$\text{m}^3$
$\dot{W} = \delta W/dt$	Puissance utile	W
$\omega$	Vitesse des courants de matière	$\text{m}\cdot\text{s}^{-1}$
$\Omega$	Section de fuite	$\text{m}^2$

## Partie I - Modélisation de la fuite d'un réservoir : mise en équation

### Généralités sur les bilans de matière et d'énergie en système ouvert

On s'intéresse au système ouvert décrit par la **figure 1**. Ce système possède une entrée de matière (notée A), une sortie (notée B). Il reçoit du milieu extérieur une puissance thermique  $\dot{Q}_e$  et une puissance de force  $\dot{W}_e$ . Il fournit au milieu extérieur une puissance thermique  $\dot{Q}_s$  et une puissance de force  $\dot{W}_s$ . Les grandeurs  $\dot{Q}_e$ ,  $\dot{Q}_s$ ,  $\dot{W}_e$ ,  $\dot{W}_s$  sont définies comme des quantités algébriques. En revanche, les débits massiques  $D_{m,A}$  et  $D_{m,B}$  sont définis comme des quantités positives.



**Figure 1** - Représentation schématique d'un système ouvert

- Q1.** Donner l'unité des grandeurs  $\dot{W}$ ,  $\dot{Q}$  et  $D_m$  mentionnées sur la **figure 1**.
- Q2.** Le système considéré est supposé évoluer en régime stationnaire. Quelle est la relation entre les débits des courants de matière entrant  $D_{m,A}$  et sortant  $D_{m,B}$ ? Justifier.
- Q3.** Appliquer le premier principe de la thermodynamique au système ouvert stationnaire de la **figure 1**. Montrer qu'il peut se mettre sous la forme :

$$\begin{pmatrix} \text{Débit d'énergie} \\ \text{entrant} \end{pmatrix} = \begin{pmatrix} \text{Débit d'énergie} \\ \text{sortant} \end{pmatrix}. \quad (1)$$

On explicitera les termes *débits d'énergie* (homogènes à une puissance) en fonction des grandeurs introduites par la **figure 1**.

- Q4.** Le système étudié de la **figure 1** est supposé évoluer en régime transitoire. On note  $m(t)$ , la fonction représentant l'évolution de sa masse  $m$  en fonction du temps  $t$ . À partir d'un bilan de matière, déduire la relation existant entre  $\frac{dm(t)}{dt}$ ,  $D_{m,A}$  et  $D_{m,B}$ . Proposer une interprétation qualitative du bilan.

On admet dans la suite l'écriture du premier principe en système ouvert, étendue aux systèmes immobiles en régime transitoire :

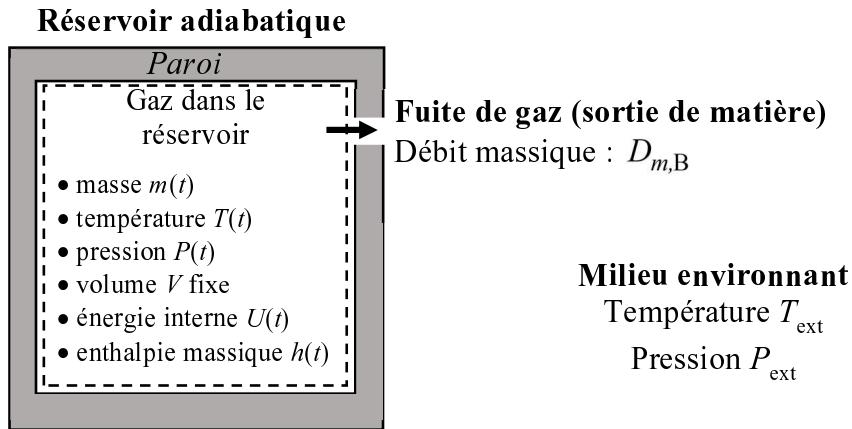
$$\frac{dU}{dt} = \begin{pmatrix} \text{Débit d'énergie} \\ \text{entrant} \end{pmatrix} - \begin{pmatrix} \text{Débit d'énergie} \\ \text{sortant} \end{pmatrix} \quad (2)$$

où  $dU/dt$  désigne la dérivée de l'énergie interne du système étudié par rapport au temps  $t$ .

- Q5.** Proposer une interprétation qualitative du bilan d'énergie traduit par l'équation (2).

### Écriture d'un modèle décrivant la fuite d'un réservoir adiabatique contenant du CO<sub>2</sub>

On s'intéresse à présent à un réservoir contenant un gaz supposé parfait. Ce réservoir indéformable (donc de volume  $V$  fixe) est le siège d'une fuite vers le milieu environnant à la température  $T_{\text{ext}} = 293$  K et à la pression  $P_{\text{ext}} = 1,01$  bar supposées fixes dans tout le sujet. Il n'est pas agité mécaniquement. Toutefois, les propriétés du gaz dans le réservoir sont supposées uniformes à chaque instant. Ce réservoir est décrit par la **figure 2**.



**Figure 2** - Réservoir adiabatique renfermant du gaz et soumis à une fuite de gaz vers le milieu environnant

#### Q6. Le modèle gaz parfait

- Dans quelle situation limite un gaz réel s'identifie-t-il exactement à un gaz parfait ?
- Donner la valeur du rapport  $C_{P,m}/R$  pour un gaz parfait monoatomique. Faire de même pour un gaz parfait diatomique.
- Pour le CO<sub>2</sub> gaz parfait, le rapport  $C_{P,m}/R$  dépend de la température selon une loi notée par la suite  $f(T)$ . L'étude de la loi  $f(T)$  sera l'objet de la partie suivante.  
Donner un argument physique expliquant pourquoi, pour le CO<sub>2</sub>,  $C_{P,m}/R$  est une fonction de la température contrairement au cas des gaz parfaits mono- et di-atomiques.
- Donner la relation unissant les variations d'énergie interne molaire  $dU_m$  et de température  $dT$  à la fonction  $f(T)$ . En déduire l'expression de  $du/dT$  où  $u$  désigne l'énergie interne massique d'un gaz parfait pur.

- On suppose que les propriétés intensives du gaz dans le réservoir (entre autres, sa température  $T$ , sa pression  $P$  et son enthalpie massique  $h$ ) sont uniformes. On considérera que les propriétés intensives du gaz sortant sont les mêmes que celles du gaz dans le réservoir. On s'intéresse au système {volume contenant le gaz à  $T$  et  $P$  dans le réservoir, paroi non comprise} représenté par des pointillés sur la **figure 2**. Montrer que l'application du bilan de matière et du premier principe au système en pointillés décrit par la **figure 2** amène :

$$\begin{cases} \frac{dm(t)}{dt} = -D_{m,B} \\ \frac{dU(t)}{dt} = -D_{m,B} \cdot h(t) \end{cases}. \quad (3)$$

On supposera négligeable l'énergie cinétique massique de la matière quittant le système.

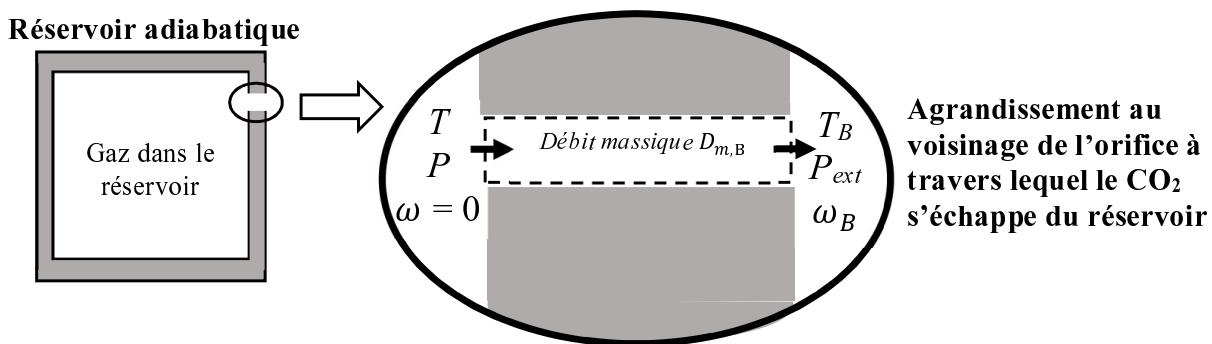
- Q8.** En introduisant la relation entre l'énergie interne massique  $u$  (en  $\text{J}\cdot\text{kg}^{-1}$ ) et l'énergie interne  $U$  (en J) du système étudié, montrer que l'équation (3) amène :

$$\frac{du}{dt} = \frac{R \cdot T}{M \cdot m} \cdot \frac{dm}{dt} \quad (4)$$

où  $R$  et  $M$  désignent respectivement la constante des gaz parfaits et la masse molaire du gaz.  
En déduire que :

$$[f(T) - 1] \frac{dT}{dt} = \frac{T}{m} \cdot \frac{dm}{dt}. \quad (5)$$

On cherche à présent à estimer le débit  $D_{m,B}$  de gaz quittant le réservoir. Pour ce faire, on s'intéresse à la zone de l'espace dans laquelle se produit la fuite.



**Figure 3** - Fuite de  $\text{CO}_2$  gazeux à travers l'orifice dans la paroi du réservoir

On considère le système {orifice dans la paroi du réservoir} ; on supposera que :

- l'écoulement de gaz à travers l'orifice est **adiabatique et stationnaire** ;
- la vitesse du gaz en entrée du système est nulle ; elle est notée  $\omega_B$  en sortie ;
- la température et la pression du gaz en entrée sont notées  $T$  et  $P$  (ce sont celles du gaz à l'intérieur du réservoir) ; en sortie, elles sont notées  $T_B$  et  $P_B = P_{\text{ext}} = 1,01$  bar.

- Q9.** Montrer que le débit massique  $D_{m,B}$  de fluide à travers la section de l'orifice est donné par :

$$D_{m,B} = \frac{\Omega \cdot \omega_B \cdot M \cdot P_{\text{ext}}}{R \cdot T_B} \quad (6)$$

où la surface  $\Omega$  désigne la section de passage du fluide à travers l'orifice.

- Q10.** *Expression de  $\omega_B$  en fonction de  $T_B$*

En appliquant l'expression du premier principe en système ouvert stationnaire, donnée par l'équation (1), au système {orifice dans la paroi du réservoir} délimité par des pointillés sur la **figure 3**, montrer que la vitesse de sortie du  $\text{CO}_2$  a pour expression :

$$\omega_B = \sqrt{2 \frac{R}{M} \int_{T_B}^T f(T) dT}. \quad (7)$$

**Q11.** Méthode de calcul de  $T_B$ 

a) En négligeant les frottements au sein du système considéré, on peut supposer l'écoulement **réversible**. Montrer, par application du second principe, que cette hypothèse amène à supposer l'écoulement *isentropique massique* (ou molaire) (i.e., à entropie massique - ou molaire - constante).

b) Montrer que la variation d'entropie massique d'un gaz parfait pur s'écrit :

$$ds = \frac{R}{M} \left[ \frac{C_{P,m}}{R} \frac{dT}{T} - \frac{dP}{P} \right]. \quad (8)$$

En déduire que  $T_B$  est solution de l'équation :

$$\int_T^{T_B} \frac{f(T)}{T} dT + \ln\left(\frac{P}{P_{\text{ext}}}\right) = 0 \quad \text{avec } P = \frac{m \cdot R \cdot T}{M \cdot V}. \quad (9)$$

Finalement, en réunissant les équations (3), (5), (6), (7) et (9), le modèle ainsi constitué est représenté par le système suivant :

$$\begin{cases} \frac{dT}{dt} = \frac{T}{m[f(T) - 1]} \cdot \frac{dm}{dt} \\ \frac{dm}{dt} = -\frac{\Omega P_{\text{ext}} \sqrt{2 \int_{T_B}^T f(T) dT}}{\sqrt{\frac{R}{M}} T_B} \end{cases} \quad (10)$$

avec  $T_B$  solution de :

$$\int_T^{T_B} \frac{f(T)}{T} dT + \ln\left(\frac{m \cdot R \cdot T}{M \cdot V \cdot P_{\text{ext}}}\right) = 0. \quad (11)$$

## Partie II - Modélisation de la fuite d'un réservoir : traitement numérique

Dans cette partie, on fera référence au modèle défini par les équations (10) et (11).

**Données numériques utiles pour cette partie :**

$$R = 8,3144 \text{ J} \cdot \text{mol}^{-1} \cdot \text{K}^{-1}$$

$$M = 44 \cdot 10^{-3} \text{ kg/mol}$$

$$T_{\text{ext}} = 293 \text{ K}$$

$$P_{\text{ext}} = 1,01 \text{ bar}$$

$$V = 1 \text{ m}^3$$

$$\Omega = 1 \text{ cm}^2$$

$$\text{Modèle pour } C_{P,m}/R : f(T) = A_1 + \frac{A_2}{T + A_3} \text{ avec } \begin{cases} T \text{ en K} \\ A_1 = 8,303 \\ A_2 = -2810 \\ A_3 = 485,6 \end{cases}$$

$$\text{Conditions initiales : } \begin{cases} T(0) = 473,15 \text{ K} \\ m(0) = 11,0 \text{ kg} \end{cases}$$

**Q12.** Intégration numérique

a) Donner le code de la fonction `integ1(T1,T2)` réalisant le **calcul analytique** de l'intégrale  $\int_{T_1}^{T_2} f(T)dT$  ( $T_1$  et  $T_2$  sont des arguments d'entrée de la fonction `integ1` qui désignent les bornes de l'intégrale).

b) Donner le code de la fonction `integ2(T1,T2)` réalisant le **calcul numérique** de l'intégrale  $\int_{T_1}^{T_2} \frac{f(T)}{T}dT$  ( $T_1$  et  $T_2$  sont des arguments d'entrée de la fonction `integ2` qui désignent les bornes de l'intégrale).

La méthode des rectangles sera adoptée pour estimer l'intégrale ; l'intervalle des températures sera divisé en 100 sous-intervalles).

Ces deux fonctions pourront être appelées autant de fois que nécessaire par la suite.

**Q13.** La fonction `chercheTB(T,m)` fournie ci-dessous renvoie, pour des valeurs connues des variables  $T$  et  $m$ , la valeur correspondante de  $T_B$  obtenue par résolution de l'équation (11) selon la méthode de Newton :

```
import numpy as np
A1 = 8.303; A2 = -2810.; A3 = 485.6
T0 = 473.15; m0 = 11.0
Pext = 1.01e5; Mw = 44e-3; Volume = 1.0; Text = 293.
R = 8.3144
def chercheTB(T,m):
    TB = 300.
    [instruction1]
    while residu > 1e-10:
        eq = [instruction2]
        deq = [instruction3]
        TBold = TB
        TB = TB - eq/deq
        residu = [instruction4]
    return TB
```

Pour compléter la portion de code ci-dessus, indiquer par quelles instructions il convient de remplacer les séquences `[instruction1]` à `[instruction4]`. Justifier.

La fonction `chercheTB(T,m)` pourra être appelée autant de fois que nécessaire par la suite.

Pour résoudre une équation différentielle mise sous la forme  $y'(t) = f(t,y(t))$ , on peut utiliser le schéma d'Euler dont on rappelle l'expression :

$$y(t + \Delta t) = y(t) + f(t,y(t)) \times \Delta t. \quad (12)$$

Dans le cas d'un système d'équations différentielles d'ordre 1, de la forme  $\mathbf{y}'(t) = \mathbf{f}(t,\mathbf{y}(t))$ , on peut généraliser le schéma d'Euler de la manière suivante :

$$\mathbf{y}(t + \Delta t) = \mathbf{y}(t) + \mathbf{f}(t,\mathbf{y}(t)) \times \Delta t \quad (13)$$

où  $\mathbf{y}(t)$  désigne le vecteur des fonctions inconnues évaluées à la date courante  $t$ .

**Q14.** Donner l'algorithme général du calcul numérique des fonctions temporelles  $T$ ,  $m$  et  $P$  entre  $t = 0$  et  $t = 10$  s, par résolution du système différentiel (10) reposant sur l'utilisation du schéma d'Euler généralisé (on choisira comme pas de temps :  $\Delta t = 0,10$  s).

Cet algorithme sera fourni sous la forme d'un logigramme. On veillera à mentionner les procédures d'initialisation des processus itératifs ainsi que leurs critères d'arrêt. Quand cela sera nécessaire, on fera appel à la fonction `chercheTB(T, m)` sans détailler sa structure.

**Q15.** Écrire le code mettant en œuvre l'algorithme proposé à la question précédente. On pourra faire appel à toutes les fonctions programmées précédemment.

**Q16.** Intuitivement, quand la fuite s'arrêtera-t-elle en pratique ?

### Partie III - Développement de corrélations pour la capacité thermique à pression constante molaire du dioxyde de carbone

On dispose d'un faible nombre de mesures expérimentales du rapport  $C_{P,m}/R$  du CO<sub>2</sub> gazeux sur un large domaine de températures  $T$  sous pression atmosphérique (où  $C_{P,m}$  désigne la capacité thermique molaire à pression constante et  $R$ , la constante des gaz parfaits).

Ces mesures (au nombre de 6) sont consignées dans un fichier `cP.txt` (voir **tableau 1**). La première colonne donne la température en K, la seconde fournit les mesures expérimentales de la quantité  $C_{P,m}/R$ .

100.0	3.513
500.0	5.367
1000.0	6.532
2000.0	7.257
3000.0	7.475
4000.0	7.586

**Tableau 1** - Contenu du fichier `cP.txt`

Dans cette partie, on cherche à développer une corrélation de la propriété  $C_{P,m}/R$  en fonction de la température. Le modèle est noté  $f(T)$  par la suite :

$$(C_{P,m}/R)_{\text{modèle}} = f(T). \quad (14)$$

Les coefficients intervenant dans la corrélation seront déterminés de manière à reproduire efficacement les données expérimentales ( $T$ ,  $(C_{P,m}/R)_{\text{exp}}$ ) dont on dispose. Pour y parvenir, deux stratégies sont envisagées et testées :

- un premier modèle de la forme :

$$f_{\text{modèle 1}}(T) = A_1 + \frac{A_2}{T + A_3} \quad (15)$$

où  $A_1$ ,  $A_2$  et  $A_3$  sont des constantes ;

- un second modèle de forme polynomiale :

$$f_{\text{modèle 2}}(T) = \sum_{i=0}^n B_i \cdot T_r^i \quad \text{avec } T_r = T/T_c \quad (16)$$

où  $T_c = 304,21$  K désigne la température critique du CO<sub>2</sub>,  $n$ , le degré du polynôme. Les  $B_i$  sont les coefficients réels du polynôme.

**Q17.** Les valeurs des capacités thermiques reportées dans le **tableau 1** ont été mesurées par calorimétrie. Il existe de nombreuses techniques de mesures calorimétriques. Décrire succinctement le principe d'une technique expérimentale de mesure de la capacité thermique d'un liquide ou d'un gaz par calorimétrie que vous connaissez ; en particulier, fournir un schéma pour illustrer le principe de la mesure et préciser comment la valeur de la capacité thermique est déduite de la mesure.

**Q18.** Montrer par une analyse dimensionnelle que le rapport  $C_{P,m}/R$  est sans dimension.

**Q19.** *Programme de lecture des données*

Indiquer la syntaxe à utiliser pour charger le fichier `cP.txt` qui contient les données expérimentales et stocker celles-ci dans deux tableaux de réels `temp` (pour les températures) et `CpR_exp` (pour les valeurs expérimentales du rapport  $C_{P,m}/R$ ).

De plus, déterminer automatiquement le nombre de points expérimentaux  $N_{\text{exp}}$  présents dans le fichier chargé.

### III.1 - Développement du premier modèle

On cherche à déterminer les coefficients  $A_1$ ,  $A_2$  et  $A_3$  du modèle n°1 donné par l'équation (15). Ces coefficients sont inconnus dans cette partie. Pour ce faire, on va utiliser une technique de minimisation. L'idée générale est la suivante :

- on forme une fonction dite *objectif*, notée  $f_{\text{obj}}$  rendant compte des écarts entre les prédictions du modèle et les valeurs expérimentales ;
- on va chercher à minimiser cette fonction en jouant sur les valeurs des coefficients  $A_1$ ,  $A_2$  et  $A_3$ .

La fonction objectif choisie a pour expression :

$$f_{\text{obj}}(A_1, A_2, A_3) = \sum_{i=1}^{N_{\text{exp}}} \delta_i^2 \quad \text{avec : } \delta_i = f_{\text{modèle 1}}(T_{\text{point exp. n}^{\circ}i}) - (C_{P,m}/R)_{\text{point exp. n}^{\circ}i}. \quad (17)$$

**Q20.** Dans l'expression de la fonction objectif, pour quelle raison les écarts ont-ils été élevés au carré ?

On range les coefficients recherchés dans le vecteur  $\mathbf{a} = (A_1, A_2, A_3)$ .

**Q21.** Écrire la syntaxe de la fonction `delta(vecA, temp, CpR_exp)` qui prend comme argument d'entrée un jeu quelconque de coefficients  $\mathbf{a}$  (`vecA` désigne le tableau contenant les éléments du vecteur  $\mathbf{a}$ ), les tableaux `temp` et `CpR_exp` et renvoie en sortie le vecteur  $\boldsymbol{\delta}$  contenant les éléments  $\delta_i$  définis par l'équation (17).

**Q22.** Écrire la syntaxe de la fonction `fobj(vecA, temp, CpR_exp)` permettant l'estimation de la fonction-objectif pour un jeu quelconque de coefficients  $\mathbf{a}$  (`vecA` désigne le tableau contenant les éléments du vecteur  $\mathbf{a}$ ). On pourra utiliser la fonction `delta` définie à la question précédente.

Un extremum local de la fonction objectif vérifie :

$$\frac{\partial f_{\text{obj}}}{\partial A_j} = 0 \quad \text{pour } 1 \leq j \leq 3. \quad (18)$$

C'est donc, en particulier, le cas d'un minimum. À présent, on a besoin d'exprimer et d'estimer les dérivées  $\frac{\partial f_{\text{obj}}}{\partial A_j}$  (pour  $1 \leq j \leq 3$ ).

**Q23.** Donner les expressions analytiques des dérivées  $\frac{\partial f_{\text{modèle } 1}}{\partial A_1}$ ,  $\frac{\partial f_{\text{modèle } 1}}{\partial A_2}$  et  $\frac{\partial f_{\text{modèle } 1}}{\partial A_3}$ .

**Q24.** Montrer que les expressions analytiques des dérivées  $\frac{\partial f_{\text{obj}}}{\partial A_j}$  peuvent se mettre sous la forme :

$$\frac{\partial f_{\text{obj}}}{\partial A_j} = 2 \sum_{i=1}^{N_{\text{exp}}} \delta_i \times \frac{\partial f_{\text{modèle } 1}}{\partial A_j} \quad \text{pour } 1 \leq j \leq 3. \quad (19)$$

et fournir explicitement leurs expressions.

On note  $\mathbf{f}'$ , le gradient de  $f_{\text{obj}}$ , i.e., le vecteur des 3 dérivées partielles  $\frac{\partial f_{\text{obj}}}{\partial A_j}$  (pour  $1 \leq j \leq 3$ ).

**Q25.** Écrire la syntaxe d'une fonction `deriv_fobj(vecA, temp, CpR_exp)` permettant l'estimation de  $\mathbf{f}'$ . Cette fonction pourra faire appel aux fonctions programmées précédemment.

Pour résoudre l'équation (18), une méthode de type Newton est envisagée. Les valeurs de  $\mathbf{a}$  minimisant la fonction-objectif sont obtenues à partir du schéma itératif suivant :

$$\mathbf{a}^{\text{itération } (k+1)} = \mathbf{a}^{\text{itération } (k)} - \mathbf{H}^{-1} \mathbf{f}' \quad (20)$$

où  $\mathbf{H}$  et  $\mathbf{f}'$  sont respectivement la matrice hessienne et le gradient de  $f_{\text{obj}}$  estimés à l'itération  $(k)$ . La matrice hessienne  $\mathbf{H}$  de la fonction  $f_{\text{obj}}$  désigne la matrice (symétrique) de ses dérivées partielles secondes dont l'expression est définie ci-après :

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f_{\text{obj}}}{\partial A_1^2} & \frac{\partial^2 f_{\text{obj}}}{\partial A_1 \partial A_2} & \frac{\partial^2 f_{\text{obj}}}{\partial A_1 \partial A_3} \\ \frac{\partial^2 f_{\text{obj}}}{\partial A_2 \partial A_1} & \frac{\partial^2 f_{\text{obj}}}{\partial A_2^2} & \frac{\partial^2 f_{\text{obj}}}{\partial A_2 \partial A_3} \\ \frac{\partial^2 f_{\text{obj}}}{\partial A_3 \partial A_1} & \frac{\partial^2 f_{\text{obj}}}{\partial A_3 \partial A_2} & \frac{\partial^2 f_{\text{obj}}}{\partial A_3^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial f'_1}{\partial A_1} & \frac{\partial f'_2}{\partial A_1} & \frac{\partial f'_3}{\partial A_1} \\ \frac{\partial f'_1}{\partial A_2} & \frac{\partial f'_2}{\partial A_2} & \frac{\partial f'_3}{\partial A_2} \\ \frac{\partial f'_1}{\partial A_3} & \frac{\partial f'_2}{\partial A_3} & \frac{\partial f'_3}{\partial A_3} \end{pmatrix}. \quad (21)$$

On choisit d'estimer numériquement les éléments de la matrice hessienne à partir de l'approximation suivante :

$$H_{ij} = \frac{\partial f'_j}{\partial A_i} \approx \frac{f'_j(\mathbf{a}_{\text{après}}) - f'_j(\mathbf{a})}{\epsilon} \quad \text{pour } 1 \leq i, j \leq 3 \quad (22)$$

où  $\mathbf{a}_{\text{après}}$  est égal à  $\mathbf{a}$  excepté pour la  $i^{\text{ème}}$  composante qui est augmentée de la quantité  $\epsilon$ .

Par exemple, si  $i = 2$ , on a :  $\mathbf{a}_{\text{après}} = (A_1 ; A_2 + \epsilon ; A_3)$ .

**Q26.** Écrire la syntaxe d'une fonction `hessienne_fobj(vecA, temp, CpR_exp)` permettant l'estimation de la matrice hessienne de la fonction-objectif en un point  $\mathbf{a}$  quelconque. Cette fonction pourra faire appel aux fonctions programmées précédemment. On prendra  $\epsilon = 10^{-5}$ .

**Q27.** On choisit comme point de départ de la procédure itérative :  $\mathbf{a}^{(0)} = (5 ; -1\ 000 ; 500)$ .

Écrire un code permettant de mettre en œuvre le calcul du  $\mathbf{a}$  optimal (noté  $\mathbf{a}^*$ ) à partir d'une méthode de type Newton. On proposera en particulier un critère d'arrêt pertinent des itérations (ce choix sera justifié).

- Q28.** Pour s'assurer que l'extremum obtenu est un minimum, il convient de vérifier que la matrice  $\mathbf{H}$  évaluée en  $\mathbf{a}^*$  est semi-définie positive (i.e., que ses valeurs propres sont positives ou nulles). Pour ce faire, écrire une fonction prenant comme argument d'entrée  $\mathbf{H}(\mathbf{a}^*)$  et renvoyant en sortie 0 si la plus petite des valeurs propres de  $\mathbf{H}(\mathbf{a}^*)$  est positive ou nulle et 1 si ce n'est pas le cas. Le calcul des valeurs propres sera effectué à l'aide d'une fonction-intrinsèque adaptée (voir annexe).

### III.2 - Développement du second modèle : le modèle polynomial

Pour décrire les mesures expérimentales, on choisit d'utiliser un **polynôme d'interpolation** dont l'expression générale est donnée par l'équation (16). Cela signifie que le degré  $n$  du polynôme et ses coefficients seront déterminés de manière à ce que la courbe représentative de la fonction  $f_{\text{modèle 2}}$  passe exactement par les 6 points expérimentaux.

**Q29.** Déduire le degré  $n$  du polynôme du nombre de données expérimentales. Justifier brièvement.

**Q30.** Écrire le système d'équations que doivent vérifier les coefficients  $B_i$  du polynôme de manière à reproduire exactement les données expérimentales.

**Q31.** Montrer que ce système est linéaire en le mettant sous la forme :

$$\mathbf{y}_{\text{exp}} = \mathbf{M} \mathbf{b} \quad (23)$$

où  $\mathbf{y}_{\text{exp}}$  est le vecteur contenant les mesures expérimentales de  $(C_{P,m}/R)_{\text{exp}}$ ,  $\mathbf{b}$  le vecteur contenant les coefficients  $B_i$  du polynôme et  $\mathbf{M}$  une matrice carrée dont les coefficients ne dépendent que des valeurs expérimentales de  $T_r = T/T_c$ .

Donner explicitement l'expression de la matrice  $\mathbf{M}$ .

**Q32.** Mathématiquement, quelle opération algébrique faut-il effectuer pour accéder aux valeurs des coefficients  $B_i$  ?

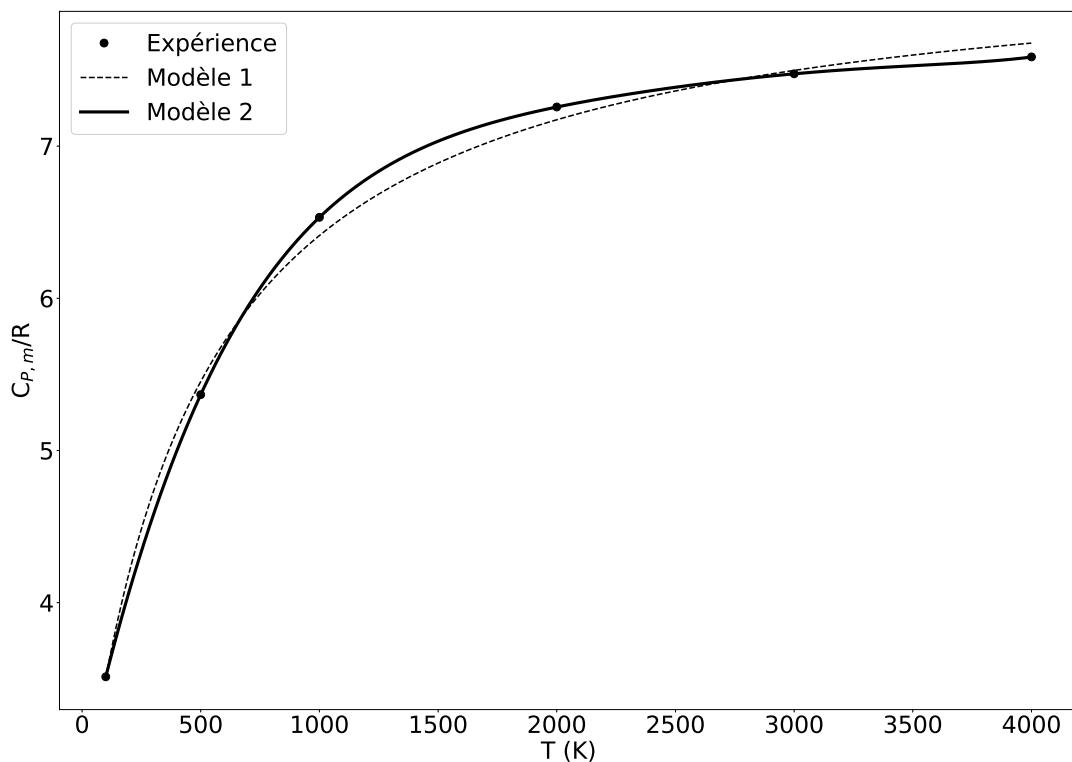
Dans la **figure 4**, on a superposé dans le plan  $(C_{P,m}, T)$  les points expérimentaux ainsi que les courbes associées aux deux modèles proposés. Les coefficients de ces modèles ont été déterminés suivant les procédures présentées précédemment.

**Q33.** Donner la syntaxe du code permettant de générer cette figure (*la courbe du modèle 1 sera représentée par un trait pointillé tandis que celle du modèle 2 sera représentée par un trait continu*). On rappelle que les coordonnées des points expérimentaux sont contenues dans les tableaux `temp` et `CpR_exp`; les 3 coefficients du modèle 1 sont stockés dans `vecA`; les 6 coefficients du modèle 2 sont supposés être stockés dans un vecteur `vecB`.

**Q34.** Commentez le graphe de la **figure 4**. Finalement, parmi les deux modèles proposés, lequel retiendriez-vous et pourquoi ?

Vous pourrez prendre en considération l'influence des incertitudes expérimentales sur les valeurs des paramètres des deux modèles considérés précédemment.

Évolution de la capacité calorifique molaire ( $C_{P,m}$ ) du CO<sub>2</sub> gazeux avec la température (T)



**Figure 4** - Confrontation des points expérimentaux et des valeurs prédites par les deux modèles proposés dans le plan  $C_{P,m}/R$  versus  $T$

## ANNEXE

### Quelques commandes utiles en langage Python

A - Bibliothèque NUMPY de Python (gestion des tableaux, matrices, vecteurs et fichiers)	13
B - Résolution d'une équation non linéaire à une inconnue	15
C - Bibliothèque MATPLOTLIB.PYPLOT de Python (gestion des graphes)	16

#### **A - Bibliothèque NUMPY de Python (gestion des tableaux, matrices, vecteurs et fichiers)**

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)**

Description : fonction permettant de créer une matrice (de type tableau) à partir d'une liste.

Argument d'entrée : une liste définissant un tableau à 1 dimension (vecteur) ou 2 dimensions (matrice).

Argument de sortie : un tableau (matrice).

<i>Exemples :</i>	Commande	Résultat
	<code>np.array([4, 3, 2])</code>	[4 3 2]
	<code>np.array([[5], [7], [1]])</code>	[[5] [7] [1]]
	<code>np.array([[3, 4, 10], [1, 8, 7]])</code>	[[3 4 10] [1 8 7]]

##### **A[i,j]**

Description : fonction qui retourne l'élément  $(i + 1, j + 1)$  de la matrice  $A$ . Pour accéder à l'intégralité de la ligne  $i + 1$  de la matrice  $A$ , on écrit  $A[i, :]$ . De même, pour obtenir toute la colonne  $j + 1$  de la matrice  $A$ , on utilise la syntaxe  $A[:, j]$ .

Argument d'entrée : une liste contenant les coordonnées de l'élément dans le tableau  $A$ .

Arguments de sortie : l'élément  $(i + 1, j + 1)$  de la matrice  $A$ .

**RAPPEL** : en langage Python, les lignes d'un tableau  $A$  de taille  $n \times m$  sont numérotées de 0 à  $n - 1$  et les colonnes sont numérotées de 0 à  $m - 1$ .

<i>Exemples :</i>	Commande	Résultat
	<code>A=np.array([[3, 4, 10], [1, 8, 7]])</code>	
	<code>A[0, 2]</code>	10
	<code>A[1, :]</code>	[1 8 7]
	<code>A[:, 2]</code>	[10 7]

##### **np.zeros((n,m))**

Description : fonction créant une matrice (tableau) de taille  $n \times m$  dont tous les éléments sont nuls.

Argument d'entrée : un tuple de deux entiers correspondant aux dimensions de la matrice à créer.

Argument de sortie : un tableau (matrice) d'éléments nuls.

<i>Exemple :</i>	Commande	Résultat
	<code>np.zeros((3, 4))</code>	[[0 0 0 0] [0 0 0 0] [0 0 0 0]]

**np.dot(mat1,mat2)**

Description : fonction calculant le produit de deux matrices `mat1` et `mat2`.

Argument d'entrée : matrices `mat1` et `mat2`.

Argument de sortie : la matrice produit de `mat1` et `mat2`.

**np.linalg.inv(mat)**

Description : fonction calculant la matrice inverse de la matrice `mat`.

Argument d'entrée : matrice `mat`.

Argument de sortie : la matrice inverse de `mat`.

**np.linalg.eig(mat)**

Description : fonction calculant les valeurs propres de la matrice `mat`.

Argument d'entrée : matrice `mat`.

Argument de sortie : c'est un tuple dont le premier élément correspond aux valeurs propres tandis que le second élément contient les vecteurs propres.

Pour accéder aux valeurs propres de `mat`, on écrira :

```
np.linalg.eig(mat)[0]
```

**np.linspace(Min,Max,nbElements)**

Description : fonction créant un vecteur (tableau) de `nbElements` nombres espacés régulièrement entre `Min` et `Max`. Le premier élément est égal à `Min`, le dernier est égal à `Max` et les éléments sont espacés de  $(\text{Max}-\text{Min})/(\text{nbElements}-1)$ .

Argument d'entrée : un tuple de 3 entiers.

Argument de sortie : un tableau (vecteur).

<i>Exemple :</i>	Commande	Résultat
	<code>np.linspace(3,25,5)</code>	<code>[3 8.5 14 19.5 25]</code>

**np.loadtxt('nom\_fichier',delimiter='string',usecols=[n])**

Description : fonction permettant de lire les données sous forme de matrice dans un fichier texte et de les stocker sous forme de vecteurs.

Argument d'entrée : le nom du fichier qui contient les données à charger, le type de caractère utilisé dans ce fichier pour séparer les données (par exemple, une espace ou une virgule) et le numéro de la colonne à charger (RAPPEL : la première colonne est affectée du numéro 0).

Argument de sortie : un tableau.

Exemple :

```
data=np.loadtxt('fichier.txt',delimiter=' ',usecols=[0])
```

Dans cet exemple, `data` est un vecteur qui correspond à la première colonne de la matrice contenue dans le fichier `fichier.txt`.

## B - Résolution d'une équation non linéaire à une inconnue

La bibliothèque adaptée est chargée par l'instruction : `from scipy import optimize`.

L'équation à résoudre est mise sous la forme :  $f(x) = 0$ , où  $x$  désigne l'inconnue. La fonction  $f$  doit d'abord être définie à l'aide de l'instruction `def`.

*Exemple* : si l'équation à résoudre s'écrit  $\log x = 3$ , la fonction  $f$  a pour expression :  $f(x) = \log x - 3$ .

Codage :

```
def f(x):  
    return log10(x) - 3.
```

La fonction `root` peut alors être utilisée pour résoudre l'équation. La variable  $x$  doit être initialisée à une valeur aussi proche que possible de la solution.

```
# Valeur initiale de x :  
Xinit = 1.  
  
# jac = None signifie que la dérivée de f (dont se sert  
#       la fonction root pour effectuer la résolution)  
#       n'est pas fournie par l'utilisateur mais estimée  
#       numériquement par Python.  
Xsol = root(f,Xinit,jac=None)
```

La solution de l'équation est rangée dans le premier élément du vecteur `Xsol.x`, c'est-à-dire :

```
Xsol.x[0]
```

## C - Bibliothèque MATPLOTLIB.PYPILOT de Python (gestion des graphes)

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,'SC')**

Description : fonction permettant de tracer 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.

Argument d'entrée : un vecteur d'abscisses  $x$  (tableau de  $n$  éléments) et un vecteur d'ordonnées  $y$  (tableau de  $n$  éléments).

La chaîne de caractères 'SC' précise le style et la couleur de la courbe tracée. Des valeurs possibles pour ces deux critères sont :

### Valeurs possibles pour S (style) :

Description	Ligne continue	Ligne traitillée	Marqueur rond	Marqueur plus
Symbol S	-	--	o	+

### Valeurs possibles pour C (couleur) :

Description	bleu	rouge	vert	noir
Symbol C	b	r	g	k

Argument de sortie : un graphique.

*Exemple :*

```
x= np.linspace(3,25,5)
y=sin(x)
plt.plot(x,y,'-b') # tracé d'une ligne bleue continue
plt.title('titre_graphique') # titre du graphe
plt.xlabel('x') # titre de l'axe des abscisses
plt.ylabel('y') # titre de l'axe des ordonnées
plt.show()
```

**FIN**

**ÉPREUVE SPÉCIFIQUE - FILIÈRE PSI****MODÉLISATION ET INGÉNIERIE NUMÉRIQUE****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.*

**RAPPEL DES CONSIGNES**

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

**Les calculatrices sont interdites.****Le sujet est composé de cinq parties.**

Sujet : 15 pages

Annexe : 1 page

Document Réponse : DR 1 à DR 4

**Le Document Réponse doit être rendu dans son intégralité avec la copie.**

# Régulation d'un système de climatisation à débit d'air variable

## Partie I - Introduction

Le réchauffement climatique est devenu l'une des principales problématiques à laquelle l'humanité doit faire face. Pour minimiser ce changement déjà en cours, celle-ci doit absolument rechercher en premier lieu une efficacité énergétique optimale dans ses activités.

Le conditionnement de l'air intérieur des habitations, notamment son maintien à une température agréable, est l'une de ces activités particulièrement énergivore ! En France, le chauffage des habitations représente, selon l'Agence de l'Environnement et de la Maîtrise de l'Energie (ADEME), environ 20 % de l'énergie totale consommée.

Suite à la montée des températures et à la répétition des périodes caniculaires, il devient de plus en plus nécessaire d'avoir recours à des systèmes de climatisation en période estivale. Ces derniers doivent également présenter le meilleur bilan énergétique possible.

Dans ce sujet, on s'intéresse plus particulièrement aux spécificités liées à la climatisation de l'ensemble d'un bâtiment. Dans ce cadre, on montre que la climatisation dite à Débit d'Air Variable (DAV) constitue le meilleur compromis entre le confort des personnes et la consommation énergétique globale du bâtiment.

Le cas d'étude proposé est la régulation à 24 °C de la température d'une pièce d'un bâtiment en contrôlant le débit d'air conditionné injecté. Les principaux échanges thermiques mis en jeu sont évalués dans la **partie II**, ce qui conduit à l'obtention de certains éléments du schéma électrique équivalent de la pièce. Ce schéma équivalent est par la suite complété dans la **partie III** où une résolution numérique permet d'aboutir à un modèle simple de l'évolution de la température de la pièce en fonction du débit d'air conditionné. La **partie IV** s'intéresse au contrôle du débit d'air conditionné par le registre à volets. Enfin, dans la **partie V**, le diagramme fonctionnel complet correspondant à la régulation de la température de la pièce est établi par rapport au cahier des charges à respecter et la robustesse de la régulation est évaluée.

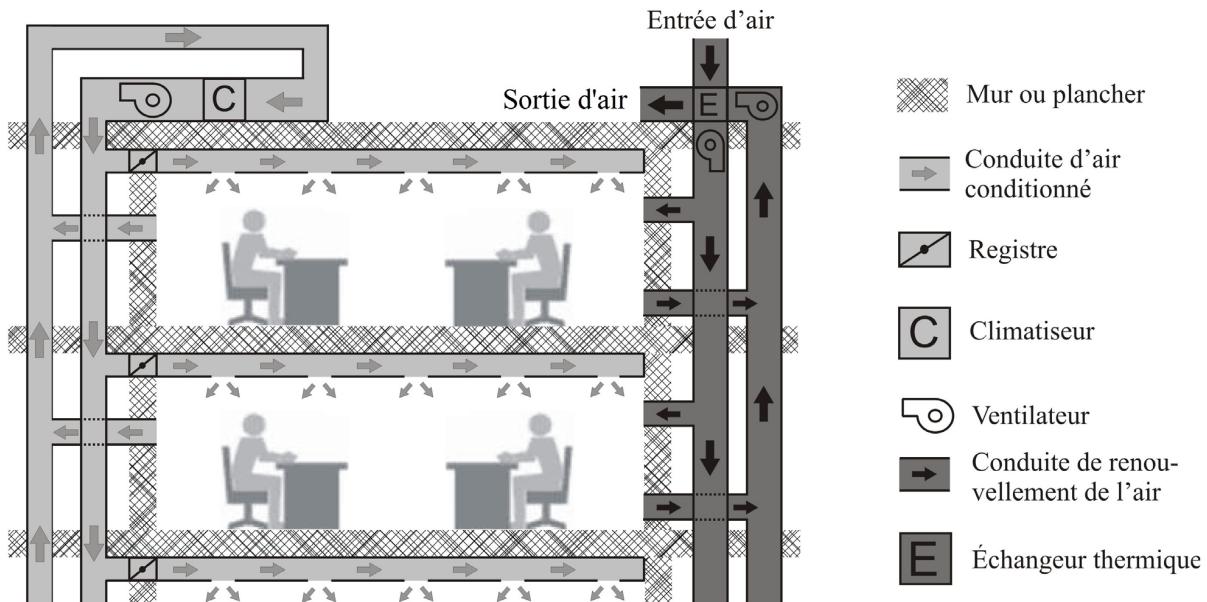
Exigences	Critères	Valeurs
Rapidité	tr 5%	2 500 s pour une variation de 4 °C
Amortissement	$D_1 \%$	35 % maximum
Précision	écart statique	1 % maximum
Stabilité	Marge de phase	45° minimum
Stabilité	Marge de gain	12 dB minimum

**Tableau 1** - Cahier des charges de l'asservissement

## Partie II - Contexte et système étudié

- Q1.** Donner deux solutions simples permettant d'améliorer la consommation énergétique d'un bâtiment.

La **figure 1** représente schématiquement une partie d'un bâtiment et son système de climatisation (conduites en gris clair) et de renouvellement de l'air (conduites en gris foncé).



**Figure 1** - Schéma d'un bâtiment et des flux d'air associés

Par souci d'économie et pour des raisons pratiques liées à l'entretien, la climatisation de l'ensemble d'un bâtiment est faite à l'aide d'un seul climatiseur. L'air conditionné est injecté dans chaque pièce par l'intermédiaire de conduites représentées en gris clair; un ventilateur permet d'en assurer la circulation.

Le système de renouvellement de l'air intérieur du bâtiment est également représenté par des conduites en gris foncé. Celui considéré ici comporte un échangeur thermique.

Les flux d'air correspondant aux systèmes de climatisation et de renouvellement de l'air sont indépendants.

- Q2.** Expliquer, brièvement, pourquoi il est nécessaire de renouveler constamment l'air intérieur d'un bâtiment.

- Q3.** Quel est le rôle de l'échangeur thermique ?

Le système de climatisation DAV consiste à maintenir l'air conditionné injecté dans les différentes pièces du bâtiment à une température constante, par exemple  $T_c = 20^\circ\text{C}$ . Le débit d'air injecté dans chaque pièce est contrôlé par des registres à volets. Il y a ainsi un registre par pièce, comme on peut le voir sur la **figure 1**. L'inclinaison des volets d'un registre modifie le débit d'air conditionné injecté dans la pièce correspondante.

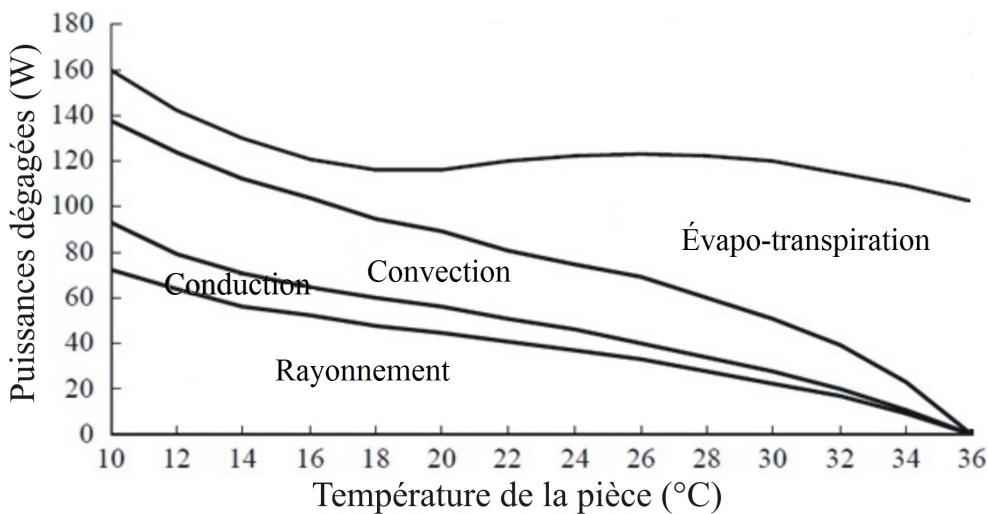
Enfin, via la vitesse de rotation du ventilateur, la pression au sein des conduites est maintenue constante.

- Q4.** Citer deux avantages au fait de contrôler le débit d'air conditionné dans chaque pièce.

## Partie III - Étude des principaux échanges thermiques

### III.1 - Puissance thermique dégagée par les personnes présentes dans la pièce

Une personne se trouvant dans la pièce dégage une certaine puissance thermique. Le transfert thermique entre la personne et la pièce se fait par conduction, par convection, par rayonnement et par évapo-transpiration (évacuation d'énergie par évaporation de la sueur). L'évolution des proportions relatives de ces quatre modes de transfert thermique est donnée, pour une personne, en fonction de la température de la pièce (**figure 2**). Par exemple, pour une température de la pièce de 10 °C, la puissance thermique fournie est au total de 160 W. Cette puissance totale comprend 73 W de rayonnement, 20 W de conduction, 45 W de convection et 22 W d'évapo-transpiration.



**Figure 2** - Puissances thermiques dégagées par une personne

- Q5.** Déterminer, pour une personne, la part en pourcentage de la puissance dégagée par chacun des quatre modes de transfert thermique à 24 °C.
- Q6.** Calculer la puissance totale  $P_{pers}$  dégagée par quatre personnes présentes dans la pièce. Dans la suite du sujet,  $P_{pers}$  sera supposée constante dans la plage de température considérée.

### III.2 - Puissance thermique extraite par le climatiseur

Le climatiseur est une machine ditherme qui reçoit un travail sous forme électrique et dont la source chaude est l'air à l'extérieur du bâtiment de température  $T_e = 28^\circ\text{C}$  et la source froide, l'air à l'intérieur de la conduite d'air conditionné de température  $T_c = 20^\circ\text{C}$ . On rappelle que l'efficacité maximale d'une telle machine, obtenue pour un fonctionnement réversible, est donnée par :

$$e_{max} = \frac{T_c}{T_e - T_c}. \quad (1)$$

Lorsque toutes les pièces sont maintenues à la température  $T$ , la puissance thermique extraite du bâtiment par le climatiseur est :

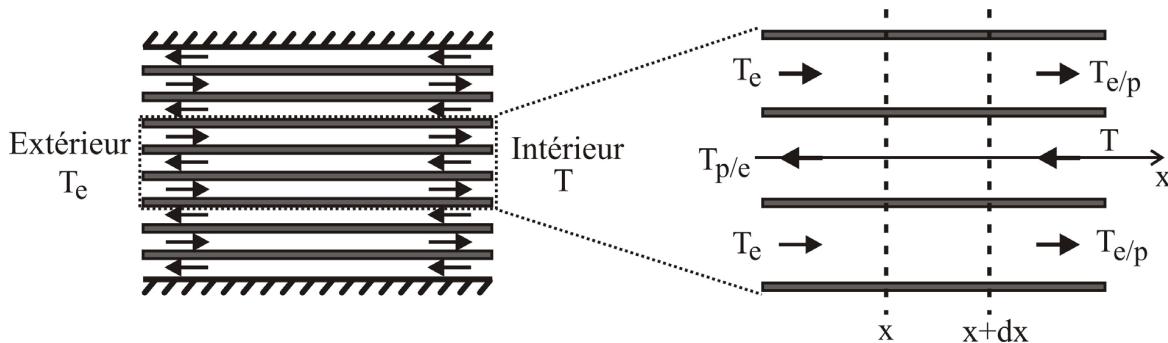
$$P_{clim,tot} = D_{m,tot} c_P (T - T_c) \quad (2)$$

avec  $D_{m,tot}$  le débit massique total d'air conditionné et  $c_P = 1,0 \cdot 10^3 \text{ J}\cdot\text{K}^{-1}\text{kg}^{-1}$  la capacité thermique massique à pression constante de l'air, supposée indépendante de la température.

- Q7.** On estime que le débit massique d'air conditionné nécessaire à la climatisation de l'ensemble du bâtiment est au maximum de  $25 \cdot 10^3 \text{ kg} \cdot \text{h}^{-1}$ . Calculer la puissance thermique maximale extraite du bâtiment par le climatiseur lorsque toutes les pièces sont maintenues à la température  $T = 24^\circ\text{C}$ .
- Q8.** Calculer la puissance électrique qui serait consommée au maximum par le climatiseur si son efficacité était égale à son efficacité maximale (la puissance mécanique fournie à l'appareil est supposée égale à la puissance électrique fournie pour le faire fonctionner).
- Q9.** En pratique la puissance électrique consommée peut dépasser la valeur calculée à la **Q8**. Expliquer pourquoi.

### III.3 - Puissance thermique apportée par le renouvellement de l'air

L'échangeur étudié ici est un échangeur à plaques schématisé sur la partie gauche de la **figure 3** : il est constitué d'une succession de fines plaques d'un bon conducteur thermique entre lesquelles circulent alternativement et à contre-courant les flux d'air entrant et sortant.



**Figure 3** - Schéma de l'échangeur à plaques étudié

La partie droite de la **figure 3** est un zoom sur un flux d'air sortant du bâtiment et sur les deux flux d'air entrants auxquels il cède une puissance thermique.

On note dans la suite :

- $T$  la température de l'air à l'intérieur de la pièce ;
- $T_e$  la température de l'air à l'extérieur du bâtiment ;
- $T_{e/p}$  la température des flux d'air entrants dans la pièce juste en sortie de l'échangeur ;
- $T_{p/e}$  la température des flux d'air sortants de la pièce juste en sortie de l'échangeur ;
- $T_{FS}(x)$  la température à l'abscisse  $x$  d'un flux d'air sortant de la pièce ;
- $T_{FE}(x)$  la température à l'abscisse  $x$  d'un flux d'air entrant dans la pièce.

Les questions **Q10** à **Q16** ont pour objectif d'établir le lien entre la puissance thermique apportée  $P_{ren}$  et le débit massique  $d_m$  de renouvellement de l'air. L'étude est menée en régime stationnaire. On ne tiendra pas compte du renouvellement de l'air des autres pièces du bâtiment.

**Q10.** Justifier que la puissance thermique apportée à la pièce par le renouvellement de l'air s'écrit :

$$P_{ren} = d_m c_P (T_{e/p} - T). \quad (3)$$

Les températures  $T_{FS}(x)$  et  $T_{FE}(x)$  vérifient le système d'équations différentielles suivant :

$$\frac{dT_{FS}}{dx}(x) = -\frac{T_{FE}(x) - T_{FS}(x)}{\lambda} \quad (4)$$

$$\frac{dT_{FE}}{dx}(x) = -\frac{T_{FE}(x) - T_{FS}(x)}{\lambda} \quad (5)$$

où  $\lambda$  est un paramètre dépendant des caractéristiques de l'échangeur.

On note  $K_{th}$  la conductance thermique linéique d'une plaque. Ainsi,  $K_{th} dx$  correspond, pour les échanges thermiques entre les flux d'air entrant et sortant, à la conductance thermique de la portion de plaque comprise entre les abscisses  $x$  et  $x + dx$ .

- Q11.** En appliquant le premier principe à l'écoulement entre les abscisses  $x$  et  $x + dx$ , établir l'équation différentielle (4) et expliciter le paramètre  $\lambda$  en fonction de  $c_P$ ,  $d_m$ ,  $K_{th}$  et  $N$  le nombre de paires flux entrant/flux sortant de l'échangeur. Comme  $N$  est grand, les effets de bords sont négligés, chaque flux d'air est supposé échangé avec deux flux d'air voisins de sens opposés.
- Q12.** Expliquer, succinctement et sans faire de calcul, comment établir l'équation différentielle (5).
- Q13.** Déduire des équations différentielles (4) et (5) que  $T_{FE}(x) - T_{FS}(x) = T_{e/p} - T$ .
- Q14.** Établir une équation différentielle vérifiée uniquement par  $T_{FE}(x)$ . La résoudre en tenant compte des conditions aux limites.
- Q15.** Montrer finalement que la puissance thermique apportée par le renouvellement de l'air s'écrit :

$$P_{ren} = \frac{d_m c_P}{1 + \frac{2 N K_{th} L}{d_m c_P}} (T_e - T) \quad (6)$$

avec  $L$  la longueur de l'échangeur. Justifier que l'échangeur est équivalent à une résistance thermique. Donner son expression.

- Q16.** Pourquoi est-il intéressant de disposer d'un échangeur pour lequel  $N$  est grand ? Pour un volume occupé par l'échangeur donné, quelle caractéristique physique de l'air, non mentionnée dans l'étude précédente, constitue un frein à l'augmentation excessive de  $N$  ? Expliquer de manière succincte.

## Partie IV - Influence du débit d'air conditionné sur la température

L'objectif de cette partie est d'établir un modèle simple de l'évolution de la température  $T$  d'une pièce en fonction du débit d'air conditionné  $D_m$ . On suppose que la pièce dispose d'une fenêtre donnant sur l'extérieur et qu'elle est, comme à la **Q6**, occupée par quatre personnes.

Les lois de la thermique, analogues aux lois de l'électrocinétique (loi des mailles, des nœuds, ponts diviseurs de tension et de courant, associations série et parallèle), pourront être utilisées sans démonstration. L'analogie est rappelée dans le **tableau 2**.

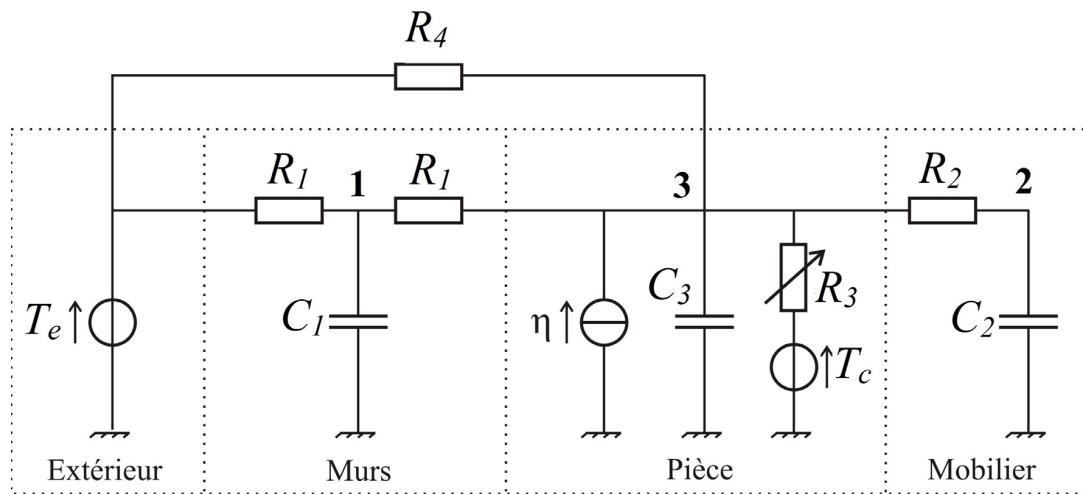
Thermique	Électrocinétique
Température	Potentiel électrique
Déférence de température	Tension
Flux ou puissance thermique	Courant électrique
Résistance thermique	Résistance électrique
Conductance thermique	Conductance électrique
Capacité thermique	Capacité
Source de chaleur ou thermostat	Source de tension
Source de puissance thermique	Source de courant

**Tableau 2** - Analogie diffusion thermique-électrocinétique**IV.1 - Schéma électrique équivalent de la pièce**

Le schéma électrique équivalent de la pièce étudiée est donné sur la **figure 4**. Il comporte :

- une source de tension  $T_e$  correspondant à l'extérieur du bâtiment ;
- une association  $R_1-C_1-R_1$  modélisant le comportement thermique des murs séparant la pièce considérée de l'extérieur du bâtiment ;
- une association  $R_2-C_2$  modélisant le comportement thermique du mobilier de la pièce ;
- une source de tension  $T_c$  associée à une résistance variable  $R_3$ , l'ensemble modélisant le système de climatisation ;
- une capacité  $C_3$  modélisant l'air à l'intérieur de la pièce ;
- une résistance  $R_4$  modélisant la fenêtre et l'échangeur ;
- une source de courant  $\eta$ .

Par souci de simplification, les échanges thermiques avec les autres pièces du bâtiment ne sont pas pris en compte.

**Figure 4** - Schéma électrique équivalent de la pièce

Les points **1** et **2** correspondent à des températures fictives notées respectivement  $T_1$  et  $T_2$ . Le point **3** correspond pour sa part à  $T$ , la température de la pièce.

- Q17.** Exprimer  $R_4$  en fonction des résistances thermiques de l'échangeur et de la fenêtre. Justifier.
- Q18.** À quoi correspond la source de courant  $\eta$ ? Justifier. On rappelle que la pièce est supposée être occupée par quatre personnes.
- Q19.** En écrivant la relation (2) pour une seule pièce du bâtiment, donner l'expression de la résistance  $R_3$ . On définira avec soin la (ou les) grandeur(s) introduite(s). Pourquoi s'agit-il d'une résistance variable pour le système de climatisation DAV étudié ici ?
- Q20.** En appliquant la loi des nœuds aux points **1**, **2** et **3**, établir trois équations faisant intervenir  $T_1$ ,  $T_2$ ,  $T$  et leurs dérivées.
- Q21.** Montrer que la modélisation adoptée conduit à l'équation différentielle matricielle :

$$\dot{X} = A X + B U \quad (7)$$

$$\text{avec } X = (T_1 \ T_2 \ T)^T \text{ et } U = (T_e \ T_c \ \eta)^T. \quad (8)$$

On explicitera avec soin les matrices  $A$  et  $B$ .  $M^T$  désigne la transposée de la matrice  $M$ .

#### IV.2 - Résolution numérique

L'objectif de cette sous-partie est de résoudre numériquement l'équation matricielle (7) à l'aide de la méthode d'Euler, dans le but de déterminer l'évolution temporelle de la température de la pièce  $T$ . On définit pour cela les instants  $t_k = k \Delta t$  avec  $\forall k \in \llbracket 0, K \rrbracket$  où  $\Delta t$  est le pas de calcul, fixé à 6 secondes par la suite, et  $K$  un entier naturel.

Le programme Python correspondant est donné dans le **tableau 3**. Les **lignes 5** et **6** définissent les matrices  $A$  et  $B$  pour la pièce considérée et dans le cas où le débit d'air conditionné dans la pièce est fixé à sa valeur maximale de  $3,5 \cdot 10^3 \text{ kg} \cdot \text{h}^{-1}$ . Les **lignes 7, 10, 21, 22 et 23** sont manquantes.

La température à l'extérieur du bâtiment vaut  $T_e = 28^\circ\text{C}$ , la température de l'air conditionné est toujours  $T_c = 20^\circ\text{C}$  et la valeur de la source de courant est  $\eta = 0,50 \text{ kW}$ .

- Q22.** Compléter la **ligne 7** définissant le vecteur  $U$ .
- Q23.** On souhaite initialiser les températures  $T_1$ ,  $T_2$  et  $T$  à  $28^\circ\text{C}$ . Compléter la **ligne 10** initialisant le vecteur  $X$ .
- Q24.** Approcher  $X(t_{k+1})$  à l'aide de  $X(t_k)$ ,  $\dot{X}(t_k)$  et  $\Delta t$  en utilisant la méthode d'Euler. À quelle condition cette approximation est-elle justifiée ?
- Q25.** Compléter les **lignes 21 à 23**. Les éléments des listes "temps" ( $t$ ) et "température" ( $T$ ) devront respectivement correspondre à des minutes et des degrés Celsius. On pourra utiliser  $np.dot(X, Y)$  pour calculer le produit de la matrice  $X$  par la matrice  $Y$ .

N°	Programme Python
1	import numpy as np
2	import matplotlib.pyplot as plt
3	
4	# Définition des matrices A, B et du vecteur U
5	A=np.array([[-8.4e-6,0,4.2e-6],[0,-2.95e-5,2.95e-5],[2.6e-3,4.7e-4,-1.08e-2]])
6	B=np.array([[4.2e-6,0,0],[0,0,0],[3.2e-5,7.74e-3,7.9e-6]])
7	...
8	
9	# Initialisation du vecteur X
10	...
11	
12	# Pas de calcul fixé à 6 secondes
13	dt=6
14	
15	# Définition des listes "temps" et "température"
16	t=[]
17	T=[]
18	
19	# Itération de la méthode d'Euler
20	for i in range(100) :
21	...
22	...
23	...
24	
25	# Affichage
26	plt.plot(t,T)
27	plt.ylim(min(T)-1,max(T)+1)
28	plt.ylabel("Température en °C")
29	plt.xlabel("Temps en min")
30	plt.grid(True)
31	plt.show()

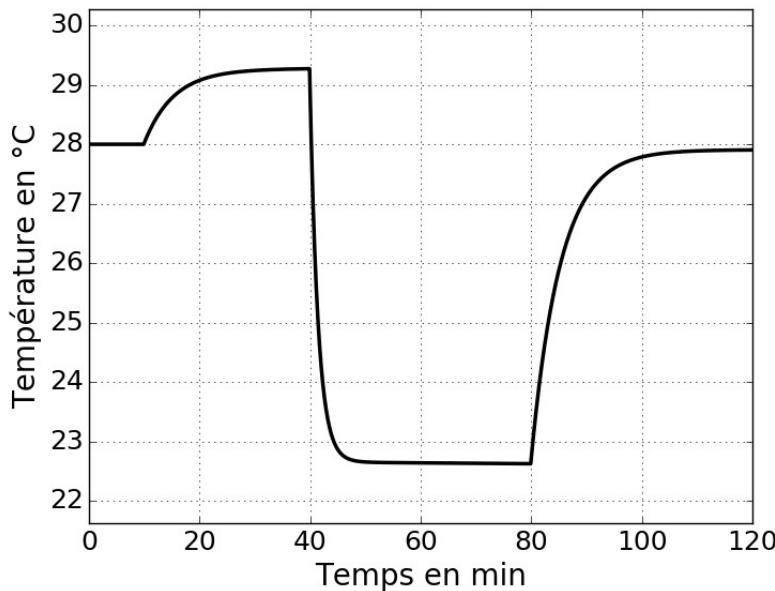
**Tableau 3** - Résolution numérique via la méthode d'Euler**IV.3 - Résultat de simulation et identification à un premier ordre**

Le programme élaboré précédemment est modifié de manière à pouvoir tenir compte des variations dans le temps de la source de courant  $\eta$  et du débit massique d'air conditionné injecté dans la pièce  $D_m$ . On considère une situation où :

- à  $t = 0$ , la pièce est depuis longtemps inoccupée et non-climatisée ;
- à  $t = 10$  min, quatre personnes entrent dans la pièce ;
- à  $t = 40$  min, le registre à volets qui contrôle l'arrivée d'air conditionné est ouvert en grand, le débit d'air conditionné est alors à son maximum, soit  $3,5 \cdot 10^3 \text{ kg} \cdot \text{h}^{-1}$  ;
- à  $t = 80$  min, la climatisation est coupée et les quatre personnes sortent de la pièce.

La température à l'extérieur du bâtiment et la température de l'air conditionné valent toujours respectivement  $T_e = 28^\circ\text{C}$  et  $T_c = 20^\circ\text{C}$ .

L'évolution simulée de la température de la pièce en fonction du temps, pour la situation considérée, est donnée **figure 5**.



**Figure 5** - Simulation de l'évolution temporelle de la température de la pièce

**Q26.** Représenter graphiquement les évolutions temporelles de la source de courant  $\eta$  et du débit massique d'air conditionné  $D_m$  sur les **DR1** et **DR2** du **Document Réponse**.

L'ouverture en grand du registre à volets à  $t = 40$  min permet de simuler la réponse indicielle du système  $S$  dont l'entrée est le débit massique d'air conditionné et la sortie, la variation de température correspondante. On se propose de déterminer un modèle de comportement de  $S$ . Celui-ci, supposé linéaire et invariant, sera par conséquent caractérisé par sa fonction de transfert :

$$H(p) = \frac{\Delta T(p)}{D_m(p)} \quad (9)$$

avec  $D_m(p)$  et  $\Delta T(p)$  les transformées de Laplace respectivement du débit massique d'air conditionné et de la variation de température.

Le **DR3** présente l'évolution temporelle de la température de la pièce déjà donnée par la **figure 5**, mais sur l'intervalle de temps plus réduit allant de  $t = 39$  min à  $t = 50$  min.

**Q27.** Justifier qu'un modèle de comportement de type passe-bas du premier ordre semble approprié pour  $S$ .

On posera donc dans la suite :

$$H(p) = \frac{H_0}{1 + \tau p}. \quad (10)$$

**Q28.** À partir du **DR3**, identifier, en explicitant clairement la démarche suivie et en adoptant les unités du Système International, le gain statique  $H_0$  et la constante de temps  $\tau$  du modèle de comportement de  $S$ .

## Partie V - Contrôle du débit d'air conditionné

L'objectif de cette partie est de modéliser la relation entre l'action du moteur sur les volets du registre et le débit qui va en résulter. Le registre est présenté **figure 6** et sur l'**annexe**.



**Figure 6** - Registre à volets

Le registre sert au réglage du débit d'air, par création d'une perte de charge variable, qui n'est pas directement proportionnelle à l'angle de pivotement des volets. Dans le modèle proposé, un seul moteur va piloter l'ensemble des volets grâce à un ensemble de bielles qui les relie.

Les étapes de la modélisation sont listées ci-dessous :

- modéliser la relation débit-angle à partir de la caractéristique du ventilateur et des conduites ;
- modéliser la relation entre l'angle de rotation du moteur et celui des volets ;
- résoudre numériquement et linéariser le modèle obtenu ;
- modéliser l'action mécanique de l'air sur un volet à partir des résultats d'une expérimentation ;
- modéliser la relation entre le couple moteur et l'angle d'inclinaison des volets.

### V.1 - Étude du débit en fonction de l'inclinaison des volets du registre

La **figure 7** montre la caractéristique aérolégique pour des angles d'inclinaison  $\theta$  de ( $0^\circ, 15^\circ, 30^\circ, 45^\circ, 60^\circ$ ). L'angle  $\theta$  correspond à l'inclinaison du volet central. Lorsque celui-ci est à l'horizontale, l'angle est de  $0^\circ$ . Pour le registre fermé, le volet est à la verticale ( $\theta = 90^\circ$ ). Le schéma cinématique du registre à volets est donné dans l'**annexe**.

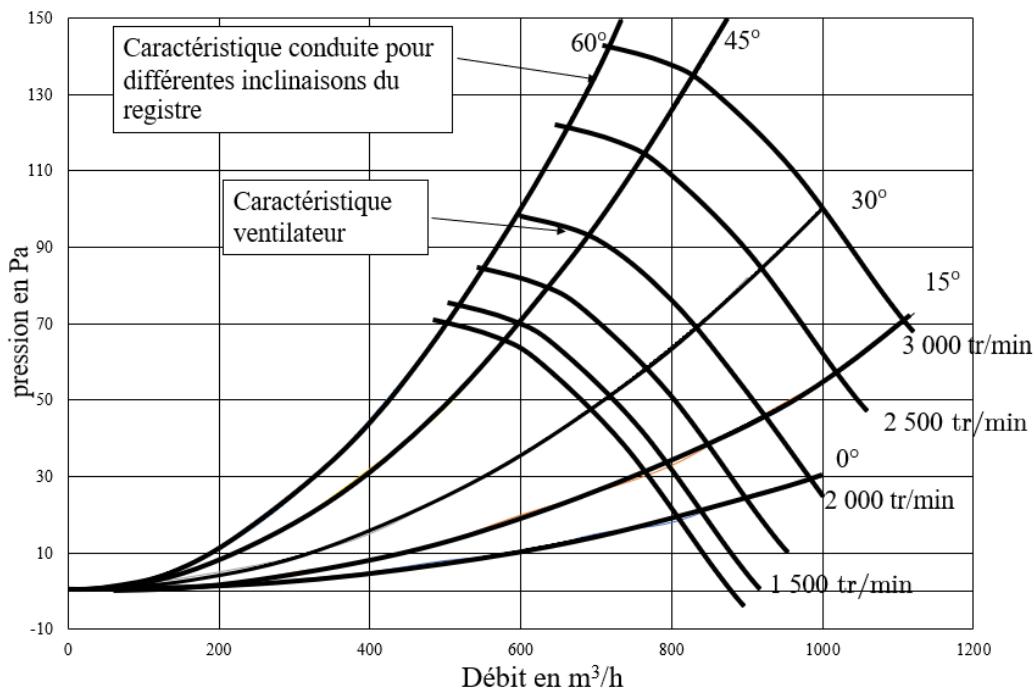
Le point de fonctionnement est l'intersection entre la caractéristique du ventilateur et celle de la conduite. Par exemple, pour une vitesse de rotation de 3 000 tr/min et une inclinaison de  $30^\circ$  des volets, le débit sera de  $1\ 000\ m^3/h$ .

Afin de minimiser les pertes d'énergie et équilibrer les différentes sorties, il faut travailler à pression constante. La pression de fonctionnement choisie est de 70 Pa. Cette pression est relative par rapport à la pression atmosphérique. Il est aussi possible de l'appeler surpression.

**Q29.** En utilisant les différents points de fonctionnement de la **figure 7**, tracer sur le **DR4** la caractéristique de débit en fonction de l'angle d'inclinaison des volets. Sachant qu'il y a deux ventilateurs qui doublent le débit, proposer une modélisation affine sous la forme :

$$D_m = K_R \theta + D \quad (11)$$

et donner les valeurs numériques de  $D$  et  $K_R$ .



**Figure 7 - Caractéristique aérolrique**

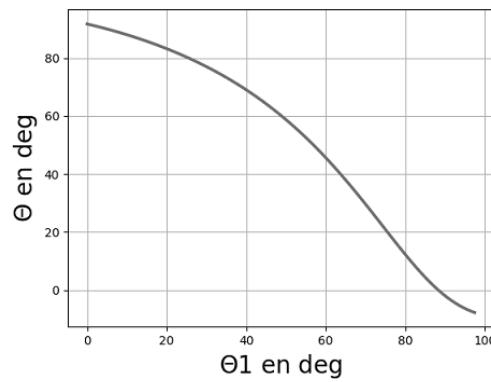
## V.2 - Modélisation de la relation entre l'angle des volets et l'angle moteur

Le schéma cinématique du registre à volets est donné sur l'**annexe**. Le moteur entraîne le volet 1.

**Q30.** Écrire la fermeture géométrique de la boucle 0-1-4-3-0 en projection sur  $\vec{x}_0$  et  $\vec{y}_0$ . Éliminer le paramètre  $\theta_4$ . Mettre le résultat sous la forme  $f(\theta_1, \theta_3) = 0$ .

**Q31.** La résolution de cette équation non-linéaire est effectuée informatiquement. Ainsi, pour la plage de valeur de  $\theta_1 \in [0, 90^\circ]$ , il faut résoudre l'équation  $f(\theta_3) = 0$ . La résolution de cette équation est obtenue en utilisant la technique de la dichotomie. Recopier et compléter les lignes 15 à 18 de la fonction dichotomie du **tableau 4**.

Le résultat de la simulation numérique est donné dans la **figure 8**.



**Figure 8 - Résultat simulation**

**Q32.** Proposer un modèle linéaire reliant  $\theta$  à l'angle  $\theta_1$  sous la forme  $\theta = 90 + K_c \theta_1$ .

N°	Programme Python
1	import numpy as np
2	import matplotlib.pyplot as plt
3	from math import *
4	
5	# Définition de la fonction f(theta3)
6	def f(theta3) :
7	s=(2.5-1.3*sin(theta1)-1.8*sin(theta3))**2+(1.8*cos(theta3)+1.3*cos(theta1))**2-2.8**2
8	return s
9	#liste de valeurs pour theta1
10	abscisse = np.linspace(0, 1.7, 100)
11	abscissedeg=(180/pi)*abscisse
12	#résolution par dichotomie avec a<b
13	def Dichotomie(f,a,b,eps) :
14	if f(b)*f(a)>=0 :return None
15	while...
16	...
17	...
18	...
19	return a
20	ordonnee_scipy=[]
21	for i in range(len(abscisse)) :
22	theta1=abscisse[i]
23	theta3=dichotomie(f,-5,5,0.1)
24	thetadeg=(180/pi)*theta3+45
25	ordonnee_scipy.append(thetadeg)
26	#Tracé du résultat
27	plt.legend()
28	plt.grid(True)
29	plt.xlabel('θ1 en deg')
30	plt.ylabel('θ en deg')
31	plt.title('loi entrée sortie')
32	plt.show()

**Tableau 4** - Programmation de la technique de la dichotomie

### V.3 - Modélisation du couple moteur en fonction de l'inclinaison

L'air exerce une action mécanique sur les volets modélisable par des torseurs couples au centre des liaisons pivots en A, E et D (**annexe**). Un essai a été effectué avec un ventilateur et un dynamomètre pour obtenir la modélisation des actions mécaniques.

Les torseurs d'actions mécaniques sont :

$$\{T_{air \rightarrow volet1}\}_A = \left\{ \begin{matrix} \vec{0} \\ -M_{A(\theta_m)} \vec{z}_0 \end{matrix} \right\}_A ; \{T_{air \rightarrow volet2}\}_E = \left\{ \begin{matrix} \vec{0} \\ -M_{E(\theta_m)} \vec{z}_0 \end{matrix} \right\}_E ; \{T_{air \rightarrow volet3}\}_D = \left\{ \begin{matrix} \vec{0} \\ -M_{D(\theta)} \vec{z}_0 \end{matrix} \right\}_D \quad (12)$$

avec  $M_{A(\theta_m)} = M_{E(\theta_m)} = 0,4 \theta_m$  et  $M_{D(\theta)} = 0,4 \theta$ .

Les torseurs cinématiques des volets sont :

$$\{V_{volet1/0}\}_A = \begin{Bmatrix} \dot{\theta}_m \vec{z}_0 \\ 0 \end{Bmatrix}_A ; \{V_{volet2/0}\}_E = \begin{Bmatrix} \dot{\theta}_m \vec{z}_0 \\ 0 \end{Bmatrix}_E ; \{V_{volet3/0}\}_D = \begin{Bmatrix} \dot{\theta} \vec{z}_0 \\ 0 \end{Bmatrix}_D . \quad (13)$$

**Q33.** Isoler l'ensemble des pièces mobiles et effectuer un bilan des puissances extérieures et intérieures. Seules seront considérées les puissances des liaisons parfaites, l'action de l'air ainsi que la puissance motrice  $P_{moteur \rightarrow 1/0} = C_m \dot{\theta}_m$ .

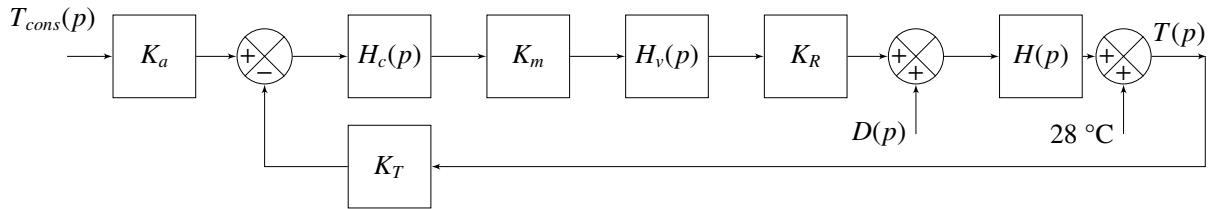
**Q34.** La faible vitesse permet de négliger l'énergie cinétique. Écrire le théorème de l'énergie cinétique et en déduire la relation entre  $C_m$ ,  $M_{E(\theta_m)}$ ,  $M_{A(\theta_m)}$ ,  $M_{D(\theta)}$ .

En prenant  $\theta_m \approx \theta$ , en déduire la relation  $C_m = f(\theta)$  et  $H_{v(p)}$ . Conclure sur la partie.

## Partie VI - Régulation de la température

L'objectif de cette partie est de régler la commande des registres afin de réguler la température de la pièce de 28 °C à 24 °C. Le cahier des charges de cet asservissement est donné **tableau 1**.

### VI.1 - Modélisation de la régulation sous la forme d'un schéma-bloc



**Figure 9** - Schéma-bloc de la commande du système

Le système est composé :

- d'un adaptateur  $\frac{U_c(p)}{T_{cons}(p)} = K_a$ ;
- d'un capteur de température  $\frac{U_T(p)}{T(p)} = K_T$  avec  $K_T = 0,05 \text{ V/}^\circ\text{C}$ ;
- d'un comparateur  $\epsilon(p) = U_c(p) - U_{T(p)}$ ;
- d'un correcteur de fonction de transfert  $H_c(p) = \frac{U(p)}{\epsilon(p)} = K_i \frac{1 + T_i p}{T_i p}$ ;
- d'un moteur  $\frac{C_m(p)}{U(p)} = K_m$  avec  $K_m = 0,01 \text{ N}\cdot\text{m/V}$ ;
- d'un registre  $H_v(p) = \frac{\theta(p)}{C_m(p)} = K_v$  avec  $K_v = 0,8 \text{ }^\circ\text{N}\cdot\text{m}$ ;
- d'un ensemble de conduite et registre  $D_m(p) = K_R \theta(p) + D(p)$ ;
- d'un bâtiment  $H(p) = \frac{\Delta T(p)}{D_m(p)} = \frac{H_0}{1 + \tau p}$ .

**Q35.** Donner la valeur de  $K_a$  afin d'avoir un asservissement correct. Justifier l'intérêt d'un correcteur proportionnel intégral.

## VI.2 - Asservissement de la température

**Q36.** Régler  $T_i$  afin de compenser le pôle dominant de la fonction de transfert en boucle ouverte. Donner, dans cette configuration, l'expression de la fonction de transfert en boucle ouverte sans prendre en compte les perturbations. Le gain en boucle ouverte sera noté  $K_{BO}$ . L'expression sera mise sous forme canonique.

**Q37.** Justifier la stabilité du système avec la fonction obtenue.

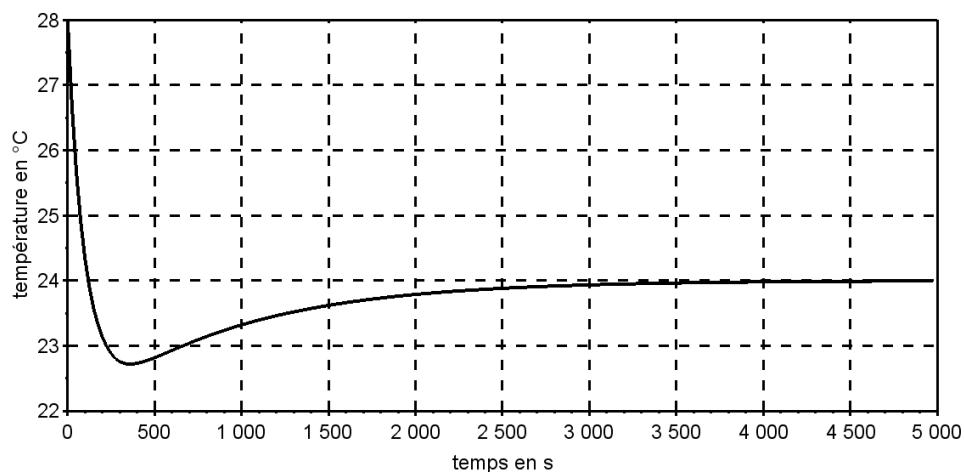
**Q38.** Donner l'expression de la fonction de transfert en boucle fermée (sans les perturbations) du système corrigé par compensation du pôle dominant. L'expression sera mise sous forme canonique.

**Q39.** Sachant que  $K_{BO} = 2 \cdot 10^{-7} K_i$ , calculer  $K_i$  afin de respecter le critère du cahier des charges en rapidité.

Ce modèle suppose que l'angle  $\theta$  peut prendre n'importe quelle valeur alors que  $\theta \in [0, 90^\circ]$ .

**Q40.** Que faut-il modifier dans le schéma afin de prendre en compte cette limitation ? Quelle conséquence aura cette modification sur les critères du cahier des charges ?

La simulation, avec cette modification, est donnée **figure 10**.

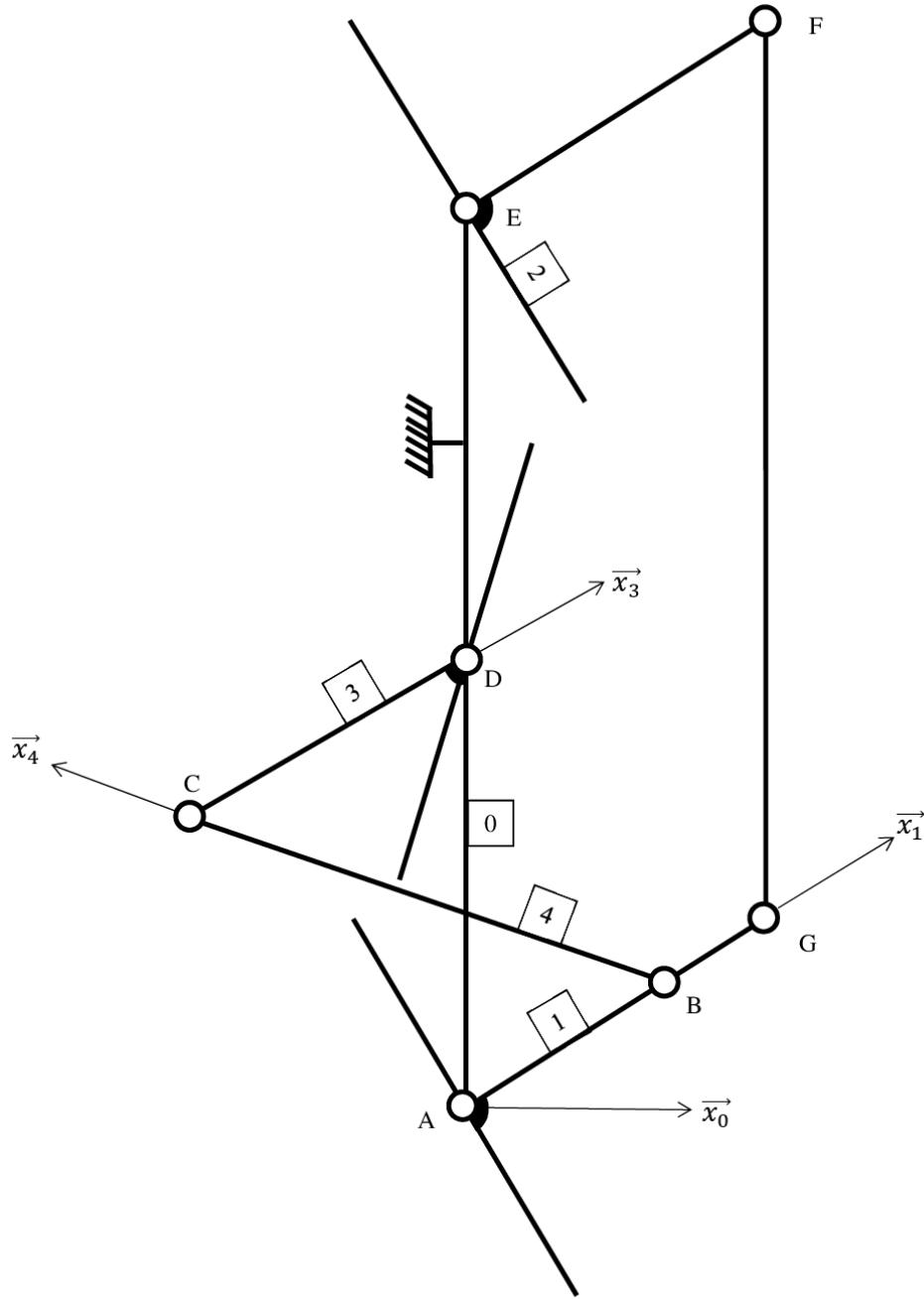


**Figure 10** - Résultat simulation finale

**Q41.** Conclure vis-à-vis des critères du cahier des charges.

## ANNEXE

### Schéma cinématique



### Paramétrage

$$\vec{AB} = 1,3l\vec{x}_1; \vec{BC} = 2,8l\vec{x}_4; \vec{CD} = 1,8l\vec{x}_3; \vec{AD} = 2,5l\vec{y}_0;$$

$$(\vec{x}_0, \vec{x}_1) = \theta_1; (\vec{x}_0, \vec{x}_4) = \theta_4; (\vec{x}_0, \vec{x}_3) = \theta_3;$$

l'angle des volets 1 et 2 est noté  $\theta_m$  avec  $\theta_m = 90 - \theta_1$ ;

l'angle du volet 3 est noté  $\theta$ , avec  $\theta = 45 + \theta_3$ .

**FIN**

**16/16**



Numéro d'inscription

Nom : \_\_\_\_\_

Numéro de table

Prénom : \_\_\_\_\_

Né(e) le

Emplacement  
QR Code

Filière : **PSI**

Session : **2021**

**Épreuve de : MODÉLISATION ET INGÉNIERIE NUMÉRIQUE**

- Consignes**
- Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer
  - Rédiger avec un stylo non effaçable bleu ou noir
  - Ne rien écrire dans les marges (gauche et droite)
  - Numérotter chaque page (cadre en bas à droite)
  - Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre

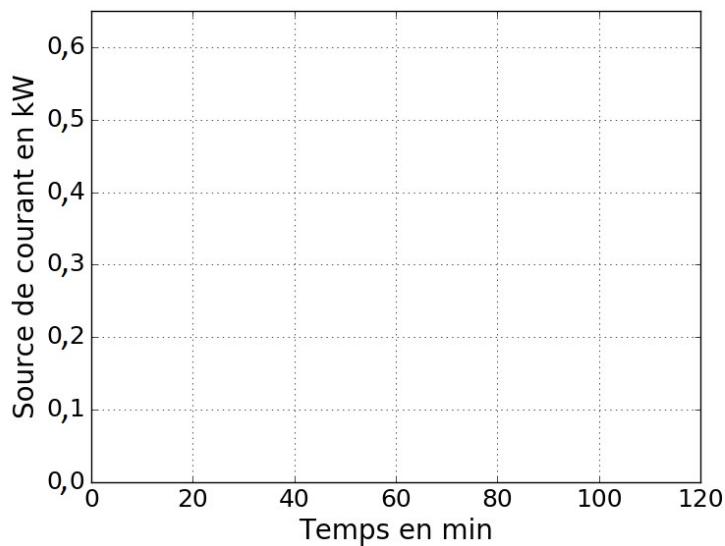
PSI3MO

## DOCUMENT RÉPONSE

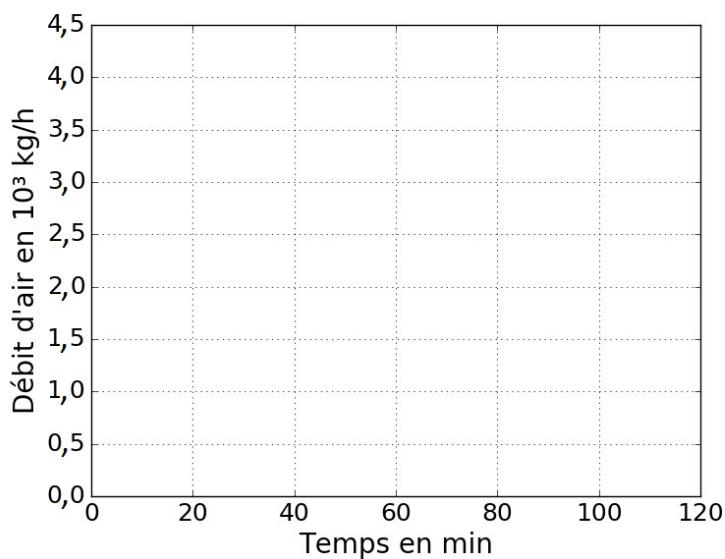
Ce Document Réponse doit être rendu dans son intégralité avec la copie.

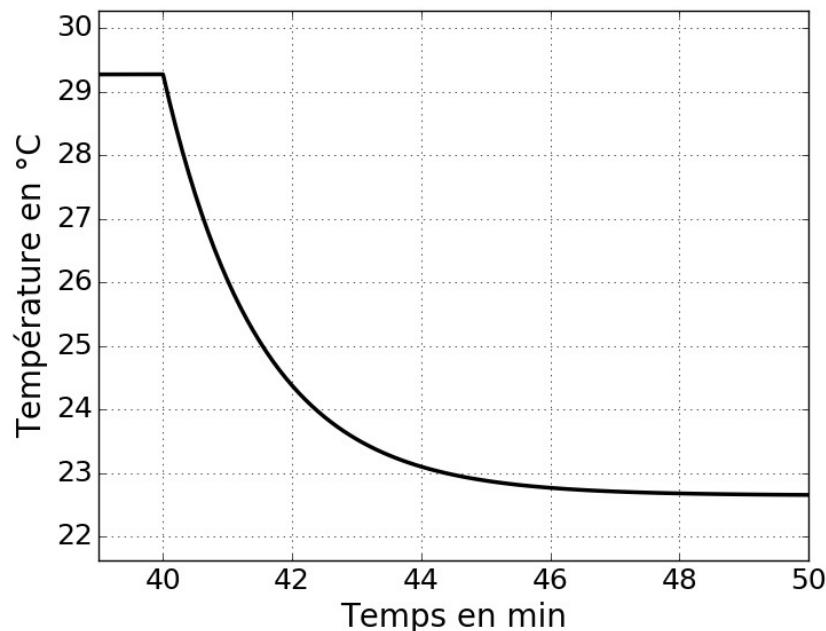
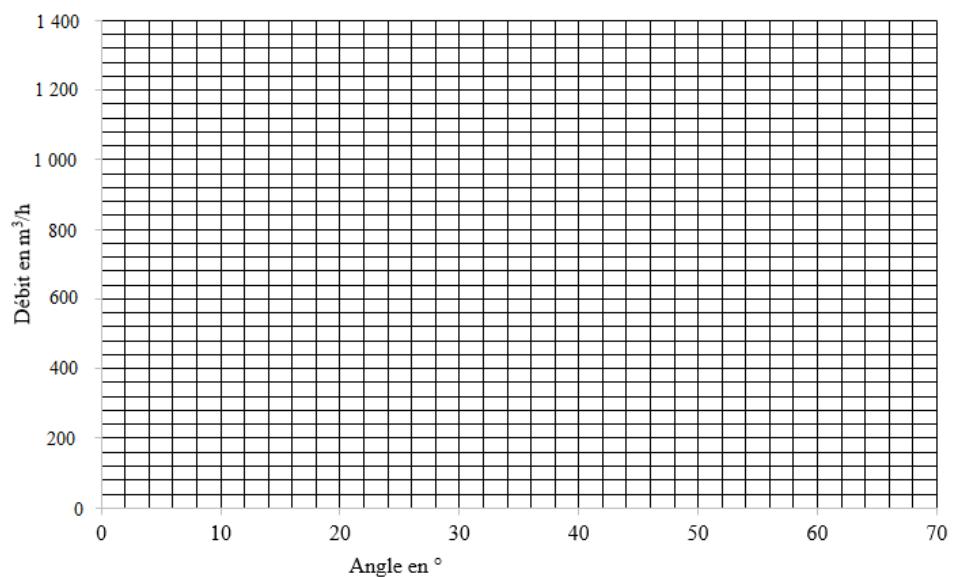
NE RIEN ÉCRIRE DANS CE CADRE

**DR1 - Q26 - Évolution temporelle de la source de courant  $\eta$**



**DR2 - Q26 - Évolution temporelle du débit massique d'air conditionné  $D_m$**



**DR3 - Q28 - Évolution de la température de la pièce entre 39 min et 50 min****DR4 - Q29 - Débit - Inclinaison**





## ÉPREUVE SPÉCIFIQUE - FILIÈRE TSI

### MODÉLISATION

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.*

#### RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

**Les calculatrices sont interdites.**

**Le sujet est composé de deux parties qui peuvent être traitées indépendamment.**

Si besoin, le candidat pourra admettre le résultat d'une question et l'utiliser dans les questions suivantes.

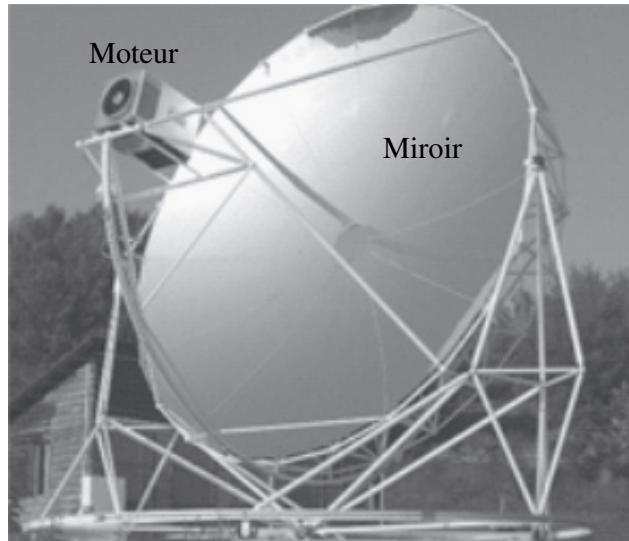
## Étude d'un système dish-stirling

### Présentation générale

Le support étudié (**figure 1**) est une unité de conversion d'énergie solaire en énergie électrique constituée principalement :

- d'un miroir parabolique (" dish " en anglais) qui concentre le rayonnement solaire sur un récepteur plan situé en son foyer;
- d'un moteur à air chaud de type Stirling dont la source chaude est fournie par le récepteur précédent;
- d'un générateur électrique entraîné par le moteur.

Un dispositif de poursuite (ou tracking) permet également d'orienter le miroir afin de suivre le mouvement du soleil.



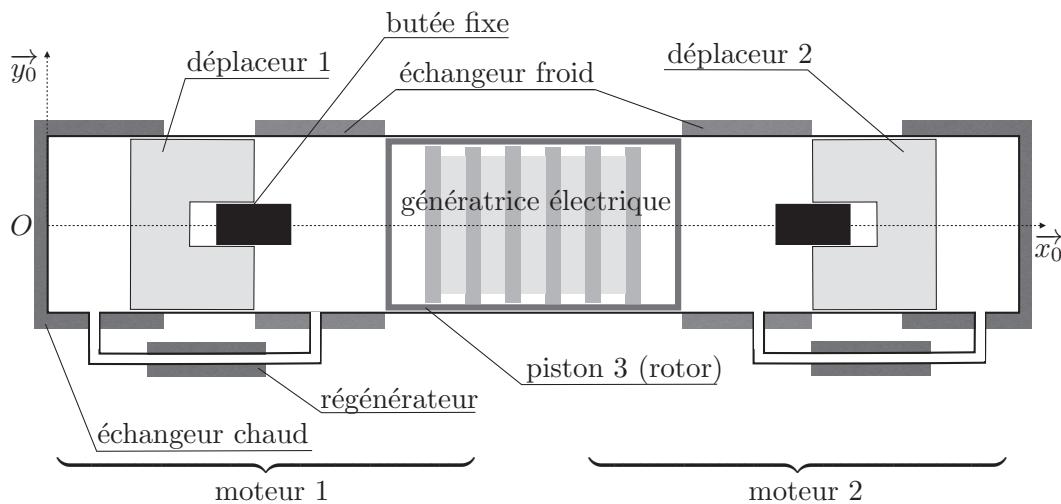
**Figure 1** - Unité dish-stirling

### Objectifs de l'étude

Dans la partie I du sujet, on présente un concept innovant de moteur Stirling dont on effectue ensuite une modélisation thermomécanique.

Dans la partie II, on propose de déterminer les caractéristiques géométriques et les performances optiques d'un miroir parabolique.

## Partie I - Moteur Stirling " double effet " de type $\beta$



**Figure 2** - Structure du moteur Stirling " double effet "

Le fonctionnement du moteur Stirling " double effet " (figure 2) est caractérisé par le travail en opposition de phase thermodynamique de ses deux moteurs Stirling  $\beta$ .

Chaque moteur est ainsi muni d'un déplaceur dont le mouvement résulte des échanges thermiques et des transferts de gaz entre les chambres froides et chaudes. Les variations de pression de chaque moteur permettent alors de mouvoir le piston 3 qui transforme cette énergie mécanique en énergie électrique via la génératrice électrique. Afin d'optimiser le rendement, il est important de faire fonctionner les deux moteurs en opposition de phase et donc de contrôler le déplacement du piston 3.

Ainsi, le moteur Stirling " double effet " évite une grande partie des pertes mécaniques liées au système d' entraînement, des usures et notamment des bruits engendrés par des liaisons mécaniques externes.

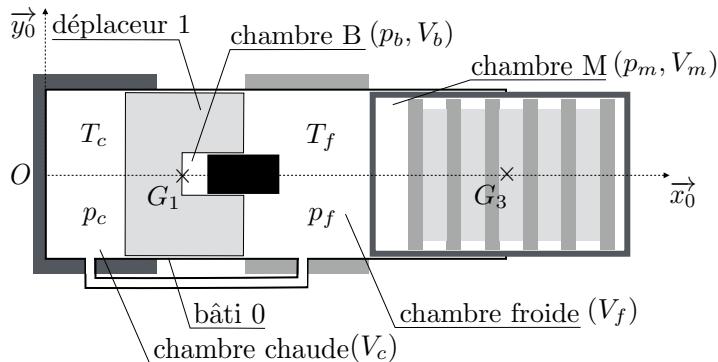
Le moteur Stirling à piston libre " double effet " présente les avantages suivants :

- la cinématique est réduite à l'extrême ; il n'y a que 3 solides en mouvement avec l'absence totale de liaisons mécaniques (ex. bielle, ressort...);
- la compacité du moteur Stirling " double effet " devrait permettre d'obtenir un meilleur rendement par rapport à des moteurs Stirling " simple effet ".

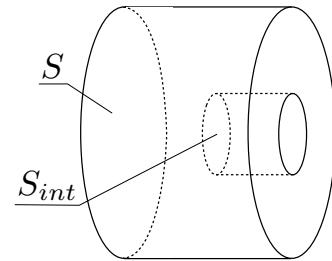
Néanmoins, cette technologie présente un certain nombre de défis :

- le fonctionnement en mode opposition de phase thermodynamique des deux moteurs élémentaires est difficile à maîtriser ;
- la position instantanée du piston 3 doit être maîtrisée par une commande dont la variable de commande est la force électromagnétique de la génératrice électrique ;
- les positions des deux déplaceurs 1 et 2 restent " libres " et résultent du couplage thermomécanique non-linéaire avec le piston 3.

### I.1 - Modélisation de la chaîne thermomécanique



**Figure 3 - Modèle thermomécanique du moteur 1**



**Figure 4 - Déplaceur seul**

#### Notations et hypothèses associées au modèle :

- $p_f, T_f$  et  $V_f$  sont respectivement la pression, la température et le volume de la chambre froide ;
- $p_c, T_c$  et  $V_c$  sont respectivement la pression, la température et le volume de la chambre chaude ;
- $p_b$  et  $V_b$  sont respectivement la pression et le volume de la chambre B ;
- $p_m$  et  $V_m$  sont respectivement la pression et le volume de la chambre M ;
- $F_{gen}\vec{x}_0$  l'action de la génératrice sur le piston 3 ;
- $\mathcal{R}_0 = (O, \vec{x}_0, \vec{y}_0, \vec{z}_0)$  le repère lié au bâti ;
- les liaisons sont supposées parfaites et l'action de la pesanteur est supposée négligeable devant les autres actions mécaniques.

#### Modèle mécanique d'un moteur Stirling " libre "

Pour établir le modèle du moteur Stirling " double effet ", nous ne considérons tout d'abord que le moteur 1 (**figures 2 et 3**). Ce modèle sera par la suite complété par analogie avec le moteur 2. Le système présente alors deux pièces mobiles :

- le déplaceur 1 de masse  $m_1$ , de section  $S$ , de section interne  $S_{int}$  et de centre de gravité  $G_1$ . On note  $\overrightarrow{OG_1} = (l_1 + x_1(t))\vec{x}_0$  avec  $l_1$  la position de référence du déplaceur et  $x_1(t)$  son déplacement par rapport à cette position de référence ;
- le piston 3 de masse  $m_3$ , de section  $S$  et de centre de gravité  $G_3$ . On note  $\overrightarrow{OG_3} = (l_3 + x_3(t))\vec{x}_0$  avec  $l_3$  la position de référence du piston et  $x_3(t)$  son déplacement par rapport à cette position de référence.

Un premier principe fondamental de la dynamique appliqué au piston 3 dans le repère  $\mathcal{R}_0$  a permis d'obtenir une première loi de mouvement pour le déplacement  $x_3(t)$  du piston :

$$\ddot{x}_3(t) = \frac{1}{m_3} (F_{gen}(t) + S \cdot (p_f(t) - p_m(t))). \quad (1)$$

- Q1.** Étant donné la forme cylindrique du déplaceur 1 et du bâti 0 (**figures 3 et 4**), proposer une liaison cinématique pour modéliser le contact entre 1 et 0, lorsqu'il n'est pas en butée. Argumenter ce choix. Donner alors la forme du torseur des actions mécaniques  $\{\mathcal{T}_{0 \rightarrow 1}\}$  dans la base  $(\vec{x}_0, \vec{y}_0, \vec{z}_0)$ .
- Q2.** Réaliser le bilan des actions mécaniques s'exerçant sur le déplaceur 1 en prenant soin d'exprimer chaque torseur d'action mécanique en fonction des paramètres du système.

**Q3.** Quelle équation du principe fondamental de la dynamique permet d'obtenir la relation suivante :

$$\ddot{x}_1(t) = \frac{1}{m_1} (S \cdot p_c(t) - (S - S_{int}) \cdot p_f(t) - S_{int} \cdot p_b(t)). \quad (2)$$

### Modèle thermodynamique des gaz

Le modèle mécanique est étroitement couplé au modèle thermodynamique des gaz des chambres chaude  $V_c(t)$  et froide  $V_f(t)$ . De plus, la quantité de gaz à l'intérieur de la chambre B est fixe. Il joue le rôle de "ressorts de rappel" pour le déplaceur 1. De la même façon, le gaz dans l'espace fermé de la chambre M fait office de ressort gazeux pour le piston 3.

On pose le paramétrage suivant pour définir les différents volumes gazeux du moteur (**figure 3**) :

- $V_0$  : volume de référence de la chambre froide et chaude ;
- chambre froide : volume  $V_f(t) = V_0 - (S - S_{int}) \cdot x_1(t) + S \cdot x_3(t)$ , de pression  $p_f$  et de température  $T_f$  ;
- chambre chaude : volume  $V_c(t) = V_0 + S \cdot x_1(t)$ , de pression  $p_c$  et de température  $T_c$  ;
- chambre adiabatique B : volume de référence  $V_b^0$  et de pression de référence  $p_b^0$  pour  $x_1 = 0$  ;
- chambre adiabatique M : volume de référence  $V_m^0$  et de pression de référence  $p_m^0$  pour  $x_3 = 0$ .

On négligera le volume de gaz présent dans le régénérateur.

### Modèle thermodynamique des chambres froide et chaude

On rappelle qu'un gaz parfait vérifie la relation :

$$p \cdot V = m \cdot r \cdot T \quad (3)$$

avec  $m$  la masse du gaz et  $r$  la capacité thermique massique du gaz. On note  $m_f$  la masse de gaz de la chambre froide et  $m_c$ , la masse de gaz de la chambre chaude. On supposera que la masse de gaz dans le régénérateur est négligeable.

**Q4.** On note ici  $m$ , la masse de l'ensemble des gaz du système : {chambre chaude, chambre froide, régénérateur}. Appliquer le principe de conservation de la masse et exprimer  $m$  en fonction de  $m_f$  et  $m_c$ , puis montrer que :

$$m \cdot r = p_f \frac{V_f}{T_f} + p_c \frac{V_c}{T_c}. \quad (4)$$

L'approche étudiée est la méthode de Schmidt qui est, en thermodynamique, une méthode d'ordre 1 très utilisée. Pour pouvoir utiliser cette méthode, on considère qu'en régime permanent :

- les variations des volumes sont sinusoïdales selon le temps, ce qui se traduit par :

- $x_3(t) = x_3^0 \cos(\omega_0 t)$  ;
- $x_1(t) = x_1^0 \cos(\omega_0 t + \varphi_1)$

avec  $x_3^0, x_1^0$  respectivement les amplitudes de la course du piston 3 et du déplaceur 1. On note  $\omega_0$  la pulsation des oscillations mécaniques et  $\varphi_1$  le déphasage entre le mouvement du piston et du déplaceur 1 ;

- les étapes de détente et de compression sont considérées isothermes ;
- le gaz est idéal : pas de perte de charge ni de fuite.

Si toutes les conditions sont satisfaites, la pression instantanée du gaz de travail au sein du moteur est identique dans les deux chambres soit :

$$p(t) = p_c = p_f. \quad (5)$$

**Q5.** En utilisant les équations (4) et (5), exprimer  $p(t)$  en fonction de  $m, r, V_f, T_f, V_c$  et  $T_c$ .

En prenant les expressions des volumes  $V_f$  et  $V_c$  définies précédemment, on montre que :

$$p(t) = \frac{m \cdot r}{A_0 \left( 1 + \frac{A_1}{A_0} x_1(t) + \frac{A_3}{A_0} x_3(t) \right)} \quad (6)$$

avec  $A_0 = \frac{V_0}{T_c} + \frac{V_0}{T_f}$ ,  $A_1 = \frac{S}{T_c} - \frac{S - S_{int}}{T_f}$  et  $A_3 = \frac{S}{T_f}$ .

**Q6.** En supposant de petits déplacements  $x_1(t)$  et  $x_3(t)$  tels que  $\left| \frac{A_1}{A_0} x_1(t) + \frac{A_3}{A_0} x_3(t) \right| \ll 1$ , réaliser un développement limité à l'ordre 1 de l'équation (6). On mettra le résultat sous la forme :

$$p(t) = p^0 - k_1 x_1(t) - k_2 x_3(t). \quad (7)$$

Préciser alors les expressions de  $p^0, k_1$  et  $k_2$  en fonction de  $m, r, A_0, A_1$  et  $A_3$ .

### Modèle thermodynamique des chambres adiabatiques B et M

Les chambres B et M subissent des transformations adiabatiques réversibles. Elles jouent le rôle de "ressorts gazeux" à l'intérieur du piston et du déplaceur 1. On a alors dans ces chambres :

$$p \cdot V^\gamma = cste \quad (8)$$

avec  $\gamma$  l'indice adiabatique du gaz.

**Q7.** Exprimer la pression instantanée du gaz  $p_b(t)$  en fonction du volume instantané  $V_b(t)$ , de  $\gamma$  et des grandeurs de références  $p_b^0$  et  $V_b^0$ .

**Q8.** Après avoir donné l'expression de  $V_b(t)$  en fonction de  $x_1(t)$  et des dimensions de la chambre B, exprimer  $p_b$  en fonction de  $V_b^0, p_b^0, \gamma, S_{int}$  et  $x_1$ .

**Q9.** Réaliser un développement limité à l'ordre 1 de l'expression pour des petits déplacements  $x_1(t)$  tels que  $\left| \frac{S_{int} x_1(t)}{V_b^0} \right| \ll 1$  et montrer que :

$$p_b(t) = p_b^0 - k_3 x_1(t). \quad (9)$$

Exprimer alors  $k_3$  en fonction des dimensions de la chambre B.

Pour la chambre M, on obtient de façon analogue :

$$p_m(t) = p_m^0 - k_4 x_3(t). \quad (10)$$

### Modèle thermomécanique du moteur Stirling "double effet"

Différentes sources de frottement (perte de charges, frottement visqueux/sec) sont également prises en compte dans le modèle. On peut alors rassembler les modèles mécaniques (équations (1) et (2)) avec les modèles thermodynamiques des différentes chambres (équations (7), (9) et (10)). Une étude analogue est également menée sur le deuxième moteur. Le modèle se résume à :

$$\ddot{x}_1(t) = K_{13} x_3(t) + K_{11} x_1(t) + D_{13} \dot{x}_3(t) + D_{11} \dot{x}_1(t); \quad (11)$$

$$\ddot{x}_3(t) = K_{33} x_3(t) + 2K_{31} x_1(t) + D_{33} \dot{x}_3(t) + \frac{1}{m_3} F_{gen}(t), \quad (12)$$

où les coefficients  $K_{ij}$  et  $D_{ij}$  sont des constantes.

**Q10.** Réécrire le système d'équation (11) et (12) dans le domaine de Laplace. On supposera que les conditions de Heaviside sont respectées.

La fonction de transfert globale  $H_{mot}(p) = \frac{X_3(p)}{F_{gen}(p)}$  du moteur Stirling " double effet " peut se mettre sous la forme :

$$H_{mot}(p) = K_{mot} \frac{1 + a_1 p + a_2 p^2}{1 + b_1 p + b_2 p^2 + b_3 p^3 + b_4 p^4}$$

où les coefficients  $K_{mot}$ ,  $a_i$  et  $b_i$  s'expriment en fonction des coefficients  $K_{ij}$ ,  $D_{ij}$  et  $m_3$ .

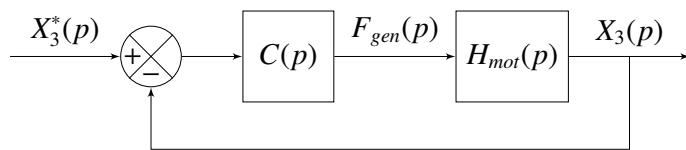
**Q11.** L'étude de  $H_{mot}(p)$  permet de faire apparaître quatre pôles :

$$p_{1,2} = 20 \pm 178i \quad p_{3,4} = -113 \pm 173i.$$

Conclure sur la stabilité du système.

### I.2 - Commande du moteur Stirling " double effet "

La variable de commande du modèle est la force électromagnétique instantanée. Les sorties du modèle sont les mouvements instantanés du piston. La force électromagnétique doit être asservie afin de stabiliser le moteur Stirling au point de fonctionnement nominal. On propose l'architecture suivante :



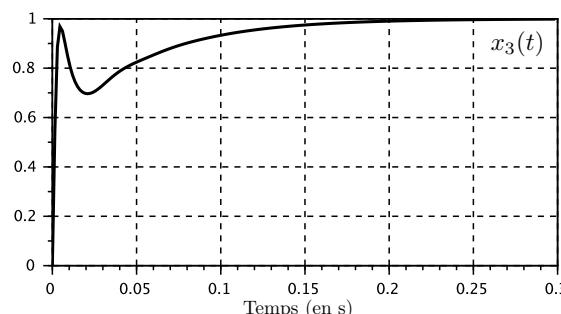
**Figure 5** - Schéma-bloc de l'asservissement en position du piston

Le correcteur  $C(p)$  présente une action intégrale et une double avance de phase, tel que :

$$C(p) = \frac{K_i}{p} \left( \frac{1 + T_2 p}{1 + T_1 p} \right)^2 \quad \text{avec} \quad T_2 > T_1.$$

**Q12.** Rappeler les avantages associés à ce correcteur.

Une simulation du système permet d'obtenir la réponse indicielle de  $x_3(t)$  (pour un échelon unitaire  $x_3^*(t)$ ) :



**Figure 6** - Réponse indicielle (adimensionnée) de l'asservissement

**Q13.** Conclure sur la stabilité de l'asservissement. Puis estimer sa rapidité, sa précision et son amortissement.

## Partie II - Étude d'un concentrateur solaire

Le rôle du miroir parabolique est de concentrer le rayonnement solaire au niveau du récepteur (ou absorbeur) du moteur afin d'obtenir une température élevée de la source chaude.

Dans la première sous-partie de l'étude, on justifie le choix d'une surface parabolique également nommée paraboloïde de révolution pour concentrer les rayons réfléchis.

Dans la deuxième sous-partie, on s'emploie à caractériser l'image du soleil, vu comme un disque lumineux, dans le plan focal du miroir.

Enfin la dernière sous-partie s'achève par une estimation de la puissance rayonnée (ou flux énergétique) reçue par l'absorbeur.

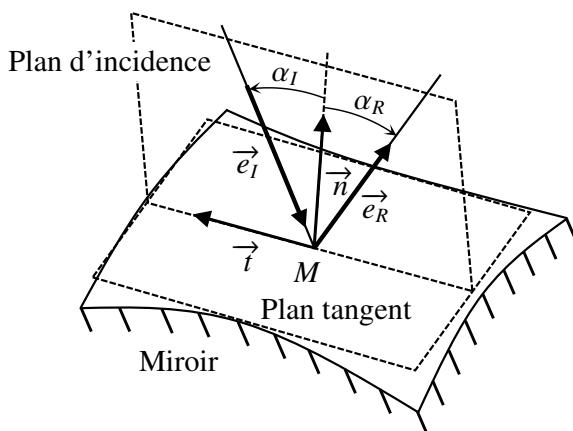
### II.1 - Équation de la surface réfléchissante

On recherche une équation de la surface d'un miroir parfait (c'est-à-dire sans absorption) vérifiant la propriété que les rayons du soleil incidents parallèles à une direction donnée possèdent un point de concours après réflexion.

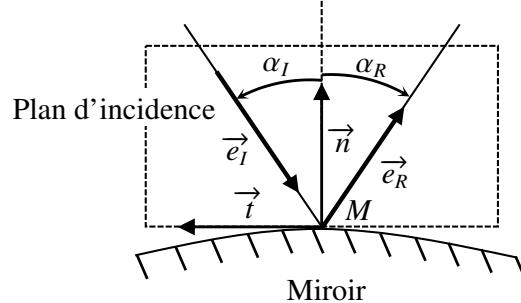
On considère ainsi un rayon lumineux dirigé selon le vecteur unitaire  $\vec{e}_I$  et incident en un point  $M$  d'un miroir de surface régulière quelconque. Le rayon réfléchi est lui dirigé selon le vecteur  $\vec{e}_R$  également unitaire.

On introduit le vecteur  $\vec{n}$  normal au plan tangent à la surface en  $M$  et de même sens que  $\vec{e}_R$ . Le plan  $(M, \vec{n}, \vec{e}_I)$  est alors appelé plan d'incidence (choisi arbitrairement dans le cas où  $\vec{n}$  et  $\vec{e}_I$  sont colinéaires). Avec  $\vec{t}$  vecteur directeur commun au plan d'incidence et au plan tangent,  $(\vec{n}, \vec{t})$  forme une base orthonormée directe du plan d'incidence.

On définit enfin l'angle d'incidence  $\alpha_I = (\vec{n}, -\vec{e}_I)$  ( $> 0$  sur la **figure 7**) et l'angle réfléchi  $\alpha_R = (\vec{n}, \vec{e}_R)$  ( $< 0$  sur la **figure 7**).



(a) Vue en perspective



(b) Vue en plan

**Figure 7** - Réflexion sur un miroir dans un cas général

**Q14.** Rappeler les lois de réflexion de Snell-Descartes. En déduire la relation vectorielle :

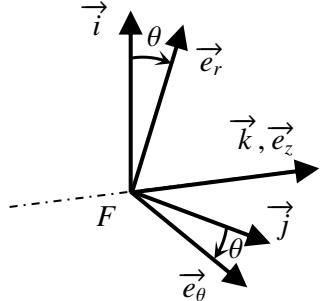
$$\vec{e}_R = \vec{e}_I - 2(\vec{e}_I \cdot \vec{n})\vec{n}. \quad (13)$$

On munit l'espace d'un repère cartésien orthonormé direct  $(F, \vec{i}, \vec{j}, \vec{k})$  où l'origine  $F$  est supposée concentrer, après réflexion, les rayons lumineux initialement dirigés selon  $-\vec{k}$ .

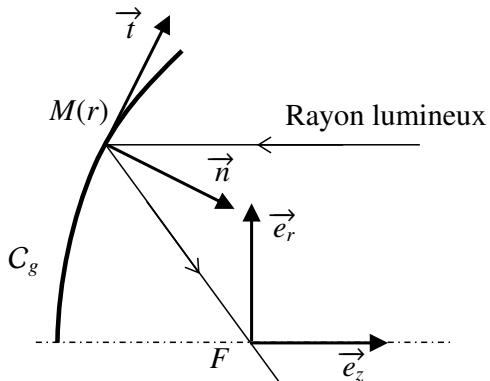
$(F, \vec{e}_r, \vec{e}_\theta, \vec{e}_z)$  est le repère cylindrique associé (**figure 8**) tel que  $\vec{k} = \vec{e}_z$  et  $\theta = (\vec{i}, \vec{e}_r) = (\vec{j}, \vec{e}_\theta) \in [0; 2\pi[$ .

Un point géométrique  $M$  est alors caractérisé aussi bien par ses coordonnées cartésiennes  $(x, y, z)$  que cylindriques  $(r, \theta, z)$  et sa position est donnée par  $\overrightarrow{FM} = x\vec{i} + y\vec{j} + z\vec{k} = r\vec{e}_r(\theta) + z\vec{e}_z$  ( $r \in \mathbb{R}_+$ ).

On suppose enfin la surface du miroir (notée  $S_G$ ) suffisamment régulière pour pouvoir être décrite en premier lieu par l'équation cartésienne  $G(x, y, z) = 0$  où  $G$  est une fonction  $\mathcal{C}^1$  de  $\mathbb{R}^3$  vers  $\mathbb{R}$ . Par abus de notation, on écrira également dans la suite  $G(x(r, \theta), y(r, \theta), z) = G(r, \theta, z)$ .



**Figure 8** - Repères utilisés



**Figure 9** - Courbe et plan d'étude

**Q15.** Soit  $M(x, y, z)$  un point de  $S_G$ .

- Rappeler les expressions de  $x$  et  $y$  en fonction de  $r$  et  $\theta$  puis de  $\vec{e}_r$  et  $\vec{e}_\theta$  en fonction de  $\vec{i}$  et  $\vec{j}$ .
- Démontrer, par dérivation de fonction composée, la relation  $\frac{\partial G}{\partial \theta}(r, \theta, z) = r\nabla G(x, y, z) \cdot \vec{e}_\theta$  où  $\nabla G(x, y, z)$  désigne le vecteur gradient de  $G$  dirigeant la normale  $\vec{n}$  à la surface  $S_G$  au point considéré.
- Montrer que  $\vec{n}$  est dans le plan  $(F, \vec{e}_r, \vec{e}_z)$  et en déduire que  $S_G$  est une surface de révolution d'axe  $(F, \vec{e}_z)$ .

On suppose désormais que  $G$  peut s'écrire sous forme séparable  $G(r, \theta, z) = z - g(r)$  où  $g$  est une fonction  $\mathcal{C}^1$  de  $\mathbb{R}_+$  vers  $\mathbb{R}$  et on restreint l'étude de  $S_G$  à une demi-méridienne  $C_g$  c'est-à-dire la courbe (**figure 9**) obtenue par l'intersection de  $S_G$  et d'un plan méridien  $(F, \vec{e}_r, \vec{e}_z)$  pour  $\theta \in [0; 2\pi[$  fixé et  $r \in \mathbb{R}_+$ .  $S_G$  peut donc être vue comme la surface engendrée par la rotation de  $C_g$  autour de l'axe de révolution  $(F, \vec{e}_z)$ .

La courbe  $C_g$  est l'ensemble des points  $M$  de paramètre  $r \in \mathbb{R}_+$  tels que  $\overrightarrow{FM} = r\vec{e}_r + g(r)\vec{e}_z$ . On définit en tout point de  $C_g$  une base orthonormée directe  $(\vec{t}, \vec{e}_\theta, \vec{n})$  où  $\vec{t}$  est choisi tangent à la courbe, de sorte que  $\vec{t}$  et  $\frac{d\overrightarrow{FM}}{dr}$  soient colinéaires et de même sens; quant à  $\vec{n}$ , il vérifie par construction la relation  $\vec{n} = \vec{t} \wedge \vec{e}_\theta$ .

On considère dans la suite un rayon lumineux parallèle à  $(F, \vec{e}_z)$  incident au point  $M(r)$  appartenant à  $C_g$  et réfléchi en direction du point  $F$ .

Par abus de notation, on notera  $g$  au lieu de  $g(r)$  et  $g'$  au lieu de  $g'(r)$ .

**Q16.** Calculer  $\vec{t} = \frac{\frac{d\overrightarrow{FM}}{dr}}{\left\| \frac{d\overrightarrow{FM}}{dr} \right\|}$ . En déduire que  $\vec{n} = \frac{-g'}{\sqrt{1+g'^2}} \vec{e}_r + \frac{1}{\sqrt{1+g'^2}} \vec{e}_z$ .

**Q17. a)** Justifier que  $\frac{\overrightarrow{FM}}{\left\| \overrightarrow{FM} \right\|} = \vec{e}_z - 2(\vec{e}_z \cdot \vec{n})\vec{n}$ .

**b)** Calculer  $\vec{e}_z - 2(\vec{e}_z \cdot \vec{n})\vec{n}$  en fonction de  $g'$ .

**c)** Calculer  $\frac{\overrightarrow{FM}}{\left\| \overrightarrow{FM} \right\|}$  en fonction de  $r$  et  $g$ .

**d)** Déduire des questions précédentes le signe de  $g'$  et que  $g'$  vérifie l'équation :

$$g'^2 - 2\frac{g}{r}g' - 1 = 0, \text{ pour tout } r \in \mathbb{R}_+^*. \quad (14)$$

**Q18. a)** Trouver l'unique solution positive de  $X^2 - 2\frac{g}{r}X - 1 = 0$ . On justifiera la réponse.

**b)** Sachant que  $X = g'$ , déterminer une expression de  $g'$  en fonction de  $g$  et  $r$ .

**c)** Montrer à l'aide du changement de fonction  $h(r) = \frac{g(r)}{r}$  que  $h$  est solution sur  $\mathbb{R}_+^*$  de l'équation différentielle :

$$\frac{h'}{\sqrt{1+h^2}} = \frac{1}{r}. \quad (15)$$

**Q19.** Pour faciliter les calculs et par abus de notation, on écrira  $h$  au lieu de  $h(r)$ .

**a)** Soit  $k \in \mathbb{R}_+^*$ . Vérifier que la fonction  $r \mapsto \frac{kr}{2} - \frac{1}{2kr}$  est solution sur  $\mathbb{R}_+^*$  de l'équation (15).

**b)** Déterminer l'ensemble des primitives sur  $\mathbb{R}_+^*$  de  $r \mapsto \frac{1}{r}$ .

**c)** Résoudre l'équation  $Y + \sqrt{1+Y^2} = A$  d'inconnue  $Y$  où  $A \in \mathbb{R}_+^*$ .

**d)** On considère la fonction  $H : r \mapsto \ln(h + \sqrt{1+h^2})$  définie sur  $\mathbb{R}_+^*$  où  $h$  est solution de (15). Calculer sa dérivée  $H'$ .

**e)** Montrer que  $\ln(h + \sqrt{1+h^2}) = \ln(r) + \lambda$  où  $\lambda \in \mathbb{R}$ , puis en déduire  $h$  en fonction de  $r$ .

**f)** Conclure à propos des solutions de l'équation (15).

**Q20.** Soit  $M(0)$  le point de  $C_g$ , situé sur l'axe  $(F, \vec{e}_z)$  et défini par  $\overrightarrow{FM} = -f \vec{e}_z$  où  $f$ , appelée distance focale du miroir, est un réel  $> 0$ . On rappelle que  $h(r) = \frac{g(r)}{r}$  pour tout  $r > 0$ . Montrer que  $g$ , fonction  $\mathcal{C}^1$  sur  $\mathbb{R}_+$ , peut y être définie par :

$$g(r) = \frac{r^2}{4f} - f. \quad (16)$$

$C_g$  est donc un arc de parabole. Cette expression de  $g$  sera conservée dans toute la suite du sujet.

**Q21.** On souhaite représenter la courbe  $C_g$  pour  $r \in [0; f]$ . Déterminer, puis tracer sur la copie des points de  $C_g$  en utilisant un pas  $\Delta r = \frac{f}{5}$  à partir de  $r = 0$ . En déduire l'allure de la courbe.

On prendra  $f = 5$  m pour les calculs et on adoptera une échelle de 1/100 pour le tracé. Les coordonnées  $(r, z)$  des points seront présentées dans un tableau et, concernant  $z$ , écrites sous forme fractionnaire et décimale approchée à 0,1 m près. Enfin, on indiquera sur la figure, en justifiant la construction, la direction de la normale au point de paramètre  $r = 3$ .

## II.2 - Caractérisation de l'image du soleil

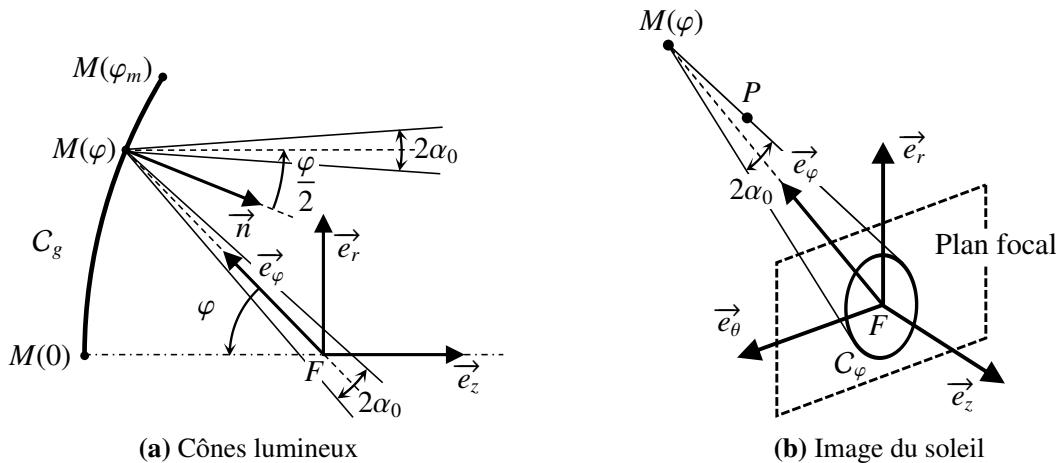
Compte tenu de la distance de la terre au soleil ( $1,496 \cdot 10^8$  km), ce dernier peut être assimilé à une source lumineuse étendue de la forme d'un disque de rayon  $6,955 \cdot 10^5$  km associé en première approximation à une densité de flux uniforme sur toute sa surface.

Par conséquent, on considère qu'en chaque point du miroir les rayons incidents et réfléchis forment un cône de révolution d'angle au sommet noté  $2\alpha_0 \approx 32'$  (une minute d'arc valant  $1' = 1/60^\circ$ ). En supposant que l'axe du miroir pointe vers le centre du soleil, l'axe du cône incident est donc parallèle à l'axe optique  $(F, \vec{e}_z)$  et l'axe du cône réfléchi passe, comme vu en II.1 et représenté sur la **figure 10a**, par son foyer  $F$ .

On cherche alors à déterminer les caractéristiques de l'image formée lorsque le cône réfléchi rencontre le plan focal  $(F, \vec{e}_r, \vec{e}_\theta)$  (**figure 10b**).

Soit  $M$  un point de la courbe  $C_g$  où  $g$  est la fonction définie sur  $\mathbb{R}_+$  par l'équation (16). On introduit l'angle  $\varphi = (\overrightarrow{FM}, -\vec{e}_z) \in [0; \pi[$  et on pose  $\overrightarrow{FM} = \rho(\varphi)\vec{e}_\varphi$  avec  $\rho(\varphi) > 0$  et  $\|\vec{e}_\varphi\| = 1$ .

$C_g$  peut donc être considérée comme une courbe paramétrée indifféremment par  $r$  ou  $\varphi$ .



**Figure 10** - Réflexion du soleil sur le miroir

**Q22.** Par abus de notation et pour faciliter les calculs  $\rho(\varphi)$  sera noté  $\rho$ .

a) Exprimer les coordonnées  $(r, z)$  du point  $M$  appartenant à  $C_g$  en fonction de ses coordonnées  $(\rho, \varphi)$ .

b) Montrer que  $\rho$  est solution de  $\rho^2 \frac{\sin^2 \varphi}{4f} + \rho \cos \varphi - f = 0$ .

c) En déduire que  $\rho$  est une fonction de  $\varphi$  définie sur  $[0; \pi[$  par :

$$\rho(\varphi) = \frac{2f}{1 + \cos \varphi}. \quad (17)$$

**Q23.** On rappelle que  $\overrightarrow{FM} = r\overrightarrow{e}_r + g(r)\overrightarrow{e}_z = \rho(\varphi)\overrightarrow{e}_\varphi$ . On note  $r_m$  le rayon d'ouverture de la parabole que l'on prend égal à  $f$  ( $r_m = f$ ) ; le point  $M$  de paramètre  $r_m$  (ou de manière équivalente  $\varphi_m$ ) est donc situé à l'extrémité de la courbe  $C_g$ , c'est-à-dire sur le bord du miroir. En déduire une expression de  $\rho_m = \rho(\varphi_m)$  en fonction de  $f$  ainsi que les valeurs de  $\cos \varphi_m$  et  $\sin \varphi_m$  correspondantes. On vérifiera également que  $\tan \varphi_m = \frac{4}{3}$ . Ces valeurs seront conservées dans la suite.

**Q24.** Soit  $P$  un point du cône de sommet  $M$  appartenant à  $C_g$ , d'axe  $(M, -\overrightarrow{e}_\varphi)$  et de demi-angle au sommet  $\alpha_0$  (**figure 10b**).

a) Montrer que  $P$  vérifie la relation  $(\overrightarrow{MP} \cdot \overrightarrow{e}_\varphi)^2 = \|\overrightarrow{MP}\|^2 \cos^2 \alpha_0$ .

On recherche une équation de la courbe notée  $C_\varphi$  issue de la section du cône par le plan focal. On pose à cet effet  $\overrightarrow{FP} = X\overrightarrow{e}_r + Y\overrightarrow{e}_\theta$  ( $P$  est dans le plan focal) et on rappelle que  $\overrightarrow{FM} = \rho\overrightarrow{e}_\varphi$ .

b) Montrer que  $(\overrightarrow{MP} \cdot \overrightarrow{e}_\varphi)^2 = (X \sin \varphi - \rho)^2$ .

c) Montrer également que l'on a  $\|\overrightarrow{MP}\|^2 = X^2 - 2\rho X \sin \varphi + Y^2 + \rho^2$ .

d) En déduire après simplification l'équation homogène vérifiée par  $X$  et  $Y$ .

$C_\varphi$  représente donc le contour de l'image du soleil issue d'un point  $M(\varphi)$  de la parabole  $C_g$  dans le plan focal ; le plan focal ayant pour vecteur normal  $\overrightarrow{e}_z$ , sa projection est une ellipse.

L'équation précédente peut s'écrire sous la forme  $\frac{(X - X_c)^2}{a^2} + \frac{Y^2}{b^2} = 1$  où les paramètres  $a$ ,  $b$  et  $X_c$  sont fonctions de  $\varphi$  sur l'intervalle d'étude  $[0; \varphi_m]$ , pour lequel on admettra que  $\varphi_m < \frac{\pi}{2} - \alpha_0$ , avec :

$$a(\varphi) = \frac{\rho(\varphi) \sin \alpha_0 \cos \alpha_0 \cos \varphi}{\cos^2 \alpha_0 - \sin^2 \varphi}, b(\varphi) = \frac{\rho(\varphi) \sin \alpha_0 \cos \varphi}{\sqrt{\cos^2 \alpha_0 - \sin^2 \varphi}} \text{ et } X_c(\varphi) = -\frac{\rho(\varphi) \sin^2 \alpha_0 \sin \varphi}{\cos^2 \alpha_0 - \sin^2 \varphi}. \quad (18)$$

On propose par la suite d'approximer ces fonctions par d'autres d'expressions plus simples.

Pour  $a$  on considère par exemple l'approximation  $\tilde{a}$  définie par  $\tilde{a}(\varphi) = \frac{\rho(\varphi) \tan \alpha_0}{\cos \varphi}$  obtenue en écrivant

$$a(\varphi) = \frac{\rho(\varphi) \tan \alpha_0 \cos \varphi}{1 - \frac{\sin^2 \varphi}{\cos^2 \alpha_0}} \approx \frac{\rho(\varphi) \tan \alpha_0 \cos \varphi}{1 - \sin^2 \varphi}. \text{ On montre alors que l'erreur relative commise sur } a(\varphi),$$

donnée pour tout  $\varphi \in [0; \varphi_m]$  par  $\left| \frac{\tilde{a} - a}{a} \right|(\varphi) = \tan^2 \alpha_0 \tan^2 \varphi$ , est négligeable (moins de 0,01%).

En procédant de la même manière pour  $b$  et  $X_c$ , on produit les approximations  $\tilde{b}$  et  $\tilde{X}_c$  telles que  $\tilde{b}(\varphi) = \rho(\varphi) \tan \alpha_0$  et  $\tilde{X}_c(\varphi) = 0$  pour lesquelles on ne demande pas de justification.

La courbe  $C_\varphi$  est donc désormais approchée par une courbe  $\tilde{C}_\varphi$  d'équation  $\frac{X^2}{\tilde{a}^2} + \frac{Y^2}{\tilde{b}^2} = 1$  dans le repère  $(F, \overrightarrow{e}_r, \overrightarrow{e}_\theta)$ .

### II.3 - Performances du concentrateur

Une mesure de la puissance rayonnée reçue par l'absorbeur du moteur situé dans le plan focal s'obtient en sommant les contributions en flux lumineux de l'ensemble des points du paraboloïde  $S_G$  sur la surface délimitée par  $\tilde{C}_0$ , c'est-à-dire le disque de centre  $F$  et de rayon  $f \tan \alpha_0$  correspondant à l'image du soleil réfléchie depuis le sommet  $M(0)$  d'une parabole  $C_g$ .

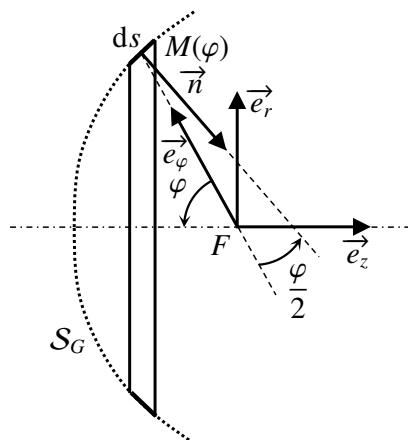
Seule une fraction de la puissance émise depuis un point de la surface  $\mathcal{S}_G$  est reçue au niveau du disque. Cette quantité est proportionnelle au rapport  $\frac{\tilde{A}_0}{\tilde{A}_\varphi}$  ( $\leq 1$ ) où  $\tilde{A}_0$  et  $\tilde{A}_\varphi$  désignent respectivement les aires délimitées par  $\tilde{C}_0$  et  $\tilde{C}_\varphi$ .

La **figure 11** représente un quart de la courbe  $\tilde{C}_\varphi$  pour un  $\varphi \in [0; \varphi_m]$ . Par symétries, l'aire  $\tilde{A}_\varphi$  est donc le quadruple de l'aire délimitée par l'arc de courbe et les axes  $X$  et  $Y$ .

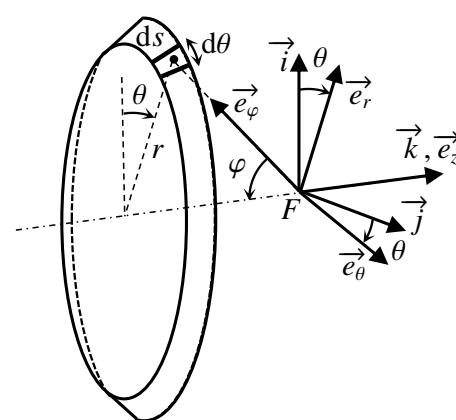
On propose le paramétrage suivant pour l'arc considéré :

$$\begin{cases} X = \tilde{a} \cos t \\ Y = \tilde{b} \sin t \end{cases} \text{ pour tout } t \in [0; \frac{\pi}{2}]. \quad (19)$$

**Q25.** Donner une interprétation de la quantité  $\int_0^{\tilde{a}} Y(X) dX$  puis la calculer. En déduire que  $\tilde{A}_\varphi = \pi \tilde{a} \tilde{b}$ .



(a) Vue en plan



(b) Vue en perspective

**Figure 12 - Anneau élémentaire**

On considère maintenant la puissance émise par un anneau élémentaire situé sur le paraboloïde, paramétré par l'angle  $\varphi$  et de longueur différentielle  $ds$  (**figure 12**).

L'anneau en question est constitué de facettes élémentaires de normale  $\vec{n}$  telle que  $(\vec{n}, \vec{e}_z) = \frac{\varphi}{2}$  et d'aire  $dS = r d\theta ds$  avec  $ds = \left\| \frac{d\vec{F}M}{d\varphi}(\varphi) \right\| d\varphi$  la longueur de l'élément de courbe associé.

La puissance émise par une facette est donnée par l'expression  $\eta p_0 dS \vec{n} \cdot \vec{e}_z = \eta p_0 \cos \frac{\varphi}{2} dS$  où  $dS \vec{n} \cdot \vec{e}_z$  représente sa section apparente,  $p_0$  est l'éclairement solaire au sol (en  $\text{W/m}^2$ ) et  $\eta < 1$  est un facteur correctif prenant en compte les pertes diverses (absorption atmosphérique, ...).

Tous les points de l'anneau générant une image du soleil de même aire  $\tilde{A}_\varphi$ , l'absorbeur reçoit une fraction de la puissance émise par l'anneau donnée par  $dP = 2\pi\eta p_0 \frac{\tilde{A}_0}{\tilde{A}_\varphi} \cos \frac{\varphi}{2} r ds$ .

**Q26.** Par abus de notation, on écrira  $\rho$  pour  $\rho(\varphi)$  et  $\rho'$  pour  $\rho'(\varphi)$ .

- a) Montrer que  $\frac{d\vec{e}_\varphi}{d\varphi}$  est unitaire et vérifie  $\vec{e}_\varphi \cdot \frac{d\vec{e}_\varphi}{d\varphi} = 0$ .
- b) Montrer que  $ds = \sqrt{\rho^2 + \rho'^2} d\varphi$ .
- c) Montrer également que  $\rho' = \rho \tan \frac{\varphi}{2}$ .
- d) En déduire que  $dP = 2\pi\eta p_0 f^2 \sin \varphi \cos \varphi d\varphi$ .
- e) Calculer en fonction de  $\eta$ ,  $p_0$ ,  $f$  et  $\varphi_m$  la puissance totale reçue par l'absorbeur en supposant la parabole totalement éclairée.
- f) Effectuer l'application numérique pour  $p_0 = 10^3$  W/m<sup>2</sup>,  $f = 5$  m et  $\eta = 0,7$ .

**FIN**







## ÉPREUVE SPÉCIFIQUE - FILIÈRE TPC

### MODÉLISATION

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.*

#### RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot *FIN* à la fin de votre composition.

Les calculatrices sont autorisées.

Le sujet est composé de quatre parties (pages 2 à 12)  
et d'une annexe (page 13).

## Modélisation de la fuite de matière d'un réservoir rempli de dioxyde de carbone gazeux

### Présentation générale

Ce sujet porte sur l'étude théorique et numérique d'une fuite de matière au sein d'une cuve contenant du dioxyde de carbone ( $\text{CO}_2$ ) gazeux. Il est constitué de **quatre parties indépendantes**.

### Quelques précisions concernant les notations utilisées

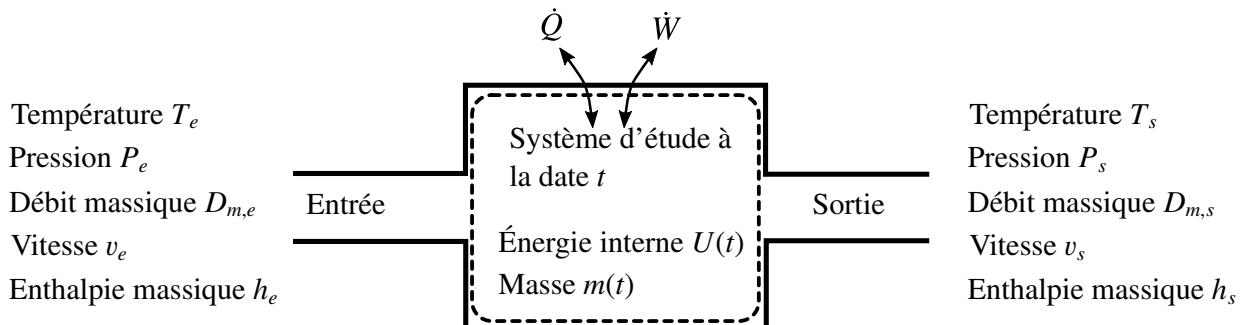
Symbol	Nom	Unité
$C_{p,m}$	Capacité thermique <b>molaire</b> à pression constante	$\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
$c_p$	Capacité thermique <b>massique</b> à pression constante	$\text{J}\cdot\text{K}^{-1}\cdot\text{kg}^{-1}$
$D_m$	Débit <b>massique</b>	$\text{kg}\cdot\text{s}^{-1}$
$e_c$	Énergie cinétique <b>massique</b>	$\text{J}\cdot\text{kg}^{-1}$
$H$	Enthalpie	J
$h$	Enthalpie <b>massique</b>	$\text{J}\cdot\text{kg}^{-1}$
$M$	Masse molaire	$\text{kg}\cdot\text{mol}^{-1}$
$m$	Masse	kg
$P$	Pression	Pa
$\dot{Q} = \delta Q/dt$	Puissance thermique	
$R$	Constante des gaz parfaits	$\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
$S$	Section de fuite	$\text{m}^2$
$T$	Température	K
$T_c$	Température critique	K
$U$	Énergie interne	J
$u$	Énergie interne <b>massique</b>	$\text{J}\cdot\text{kg}^{-1}$
$V$	Volume	$\text{m}^3$
$v$	Vitesse des courants de matière	$\text{m}\cdot\text{s}^{-1}$
$\dot{W} = \delta W/dt$	Puissance utile	

On prendra garde à bien distinguer les grandeurs **molaires** des grandeurs **massiques**. Certaines de ces grandeurs peuvent être évaluées en entrée ou en sortie du système, ce que l'on identifie par un indice  $e$  en entrée ou  $s$  en sortie.

## Partie I - Modélisation de la fuite d'un réservoir : mise en équation

### Généralités sur les bilans de matière et d'énergie en système ouvert

On s'intéresse au système ouvert décrit par la **figure 1**. Ce système possède une entrée et une sortie de matière où la température  $T$ , la pression  $P$ , la vitesse  $v$ , le débit massique  $D_m$  et l'enthalpie massique  $h$  sont définis. Le système échange algébriquement, avec le milieu extérieur, la puissance thermique  $\dot{Q}$  et la puissance utile  $\dot{W}$ . Les débits massiques  $D_{m,e}$  et  $D_{m,s}$  sont positifs. Il n'y a pas de travaux dérivant d'une énergie potentielle.



**Figure 1** - Représentation schématique d'un système ouvert

- Q1.** Rappeler l'unité des grandeurs  $\dot{W}$ ,  $\dot{Q}$  et  $D_m$  mentionnées sur la **figure 1**.
- Q2.** Par une méthode de votre choix (issue de la thermodynamique ou de la mécanique des fluides ou autres), établir l'équation locale de conservation de la matière :

$$\frac{dm}{dt} = D_{m,e} - D_{m,s}. \quad (1)$$

- Q3.** À l'aide d'un bilan énergétique, établir l'expression du premier principe de la thermodynamique en régime stationnaire pour le système ouvert représenté sur la **figure 1**. On mettra le résultat sous la forme :

$$D_m(\Delta h + \Delta e_c) = \dot{Q} + \dot{W}. \quad (2)$$

Donner la relation liant  $D_m$ ,  $D_{m,e}$  et  $D_{m,s}$  et expliciter  $\Delta e_c$  en fonction des données.

On admet dans la suite l'écriture du premier principe en système ouvert, étendue aux systèmes en régime transitoire :

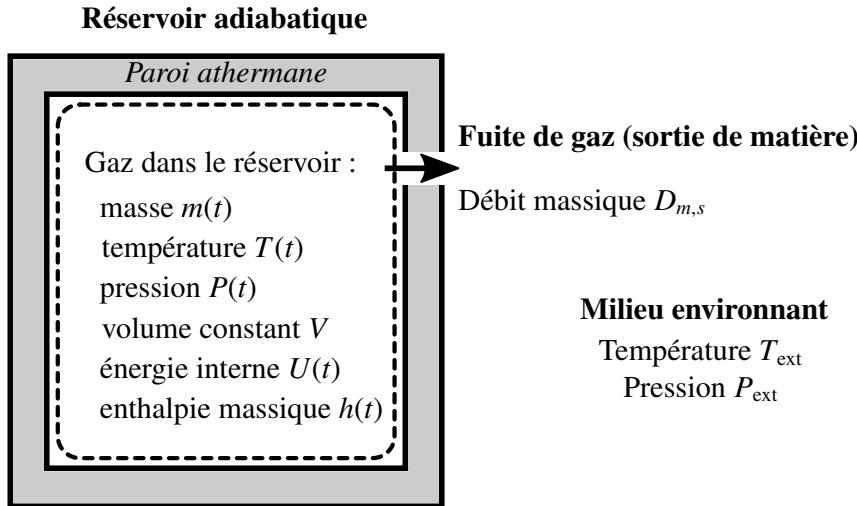
$$\frac{dU}{dt} = \left( \begin{array}{c} \text{Débit d'énergie} \\ \text{entrant} \end{array} \right) - \left( \begin{array}{c} \text{Débit d'énergie} \\ \text{sortant} \end{array} \right) \quad (3)$$

où  $dU/dt$  désigne la dérivée de l'énergie interne du système étudié par rapport au temps.

- Q4.** Proposer une interprétation qualitative du bilan d'énergie traduit par l'équation (3).

### Écriture d'un modèle décrivant la fuite d'un réservoir adiabatique contenant du CO<sub>2</sub>

On s'intéresse à présent à un réservoir contenant un gaz supposé parfait. Ce réservoir indéformable (donc de volume  $V$  constant) est le siège d'une fuite vers le milieu environnant à la température  $T_{\text{ext}}$  et à la pression  $P_{\text{ext}}$  supposées constantes dans tout le problème. Il n'est pas agité mécaniquement. Toutefois, les propriétés du gaz dans le réservoir sont supposées uniformes à chaque instant. Ce réservoir est représenté sur la **figure 2**.



**Figure 2** - Réservoir adiabatique renfermant du gaz et sujet à une fuite de gaz vers le milieu environnant

**Q5.** On suppose que les grandeurs intensives du gaz dans le réservoir sont uniformes et que celles du gaz sortant sont identiques à celles du réservoir. On suppose négligeable l'énergie cinétique massique de la matière quittant le système. On s'intéresse au système *{volume contenant le gaz à  $T$  et  $P$  dans le réservoir, paroi non comprise}* représenté par des pointillés sur la **figure 2**. Justifier que l'application du bilan de matière et du premier principe (équations (1) et (3)) au système en pointillés, décrit par la **figure 2**, permet d'obtenir le système d'équations :

$$\begin{cases} \frac{dm}{dt} = -D_{m,s} \\ \frac{dU}{dt} = -D_{m,s} h \end{cases} . \quad (4)$$

**Q6.** *Le modèle gaz parfait*

- a) Dans quelle situation limite un gaz réel s'identifie-t-il exactement à un gaz parfait ?
- b) Donner la valeur du rapport  $C_{P,m}/R$  pour un gaz parfait monoatomique.
- c) On observe expérimentalement que pour le CO<sub>2</sub> gaz parfait, le rapport  $C_{P,m}/R$  dépend de la température. Donner un argument physique expliquant pourquoi.
- d) On note  $u$  l'énergie interne massique du gaz parfait,  $n$  le nombre de moles de gaz,  $m$  sa masse et  $M$  sa masse molaire. Justifier la relation  $h = u + \frac{nRT}{m}$  et en déduire l'équation :

$$\frac{du}{dT} = \frac{R}{M} [f(T) - 1] \quad (5)$$

dans laquelle  $f(T) = C_{P,m}/R$ .

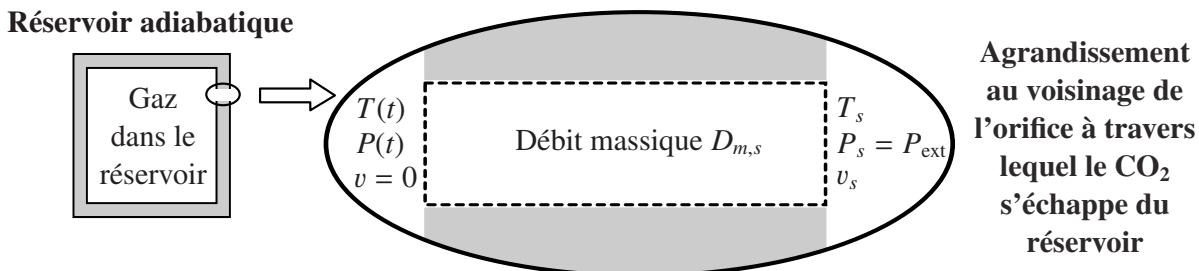
- Q7.** À partir de la relation  $u = U/m$  liant l'énergie interne massique  $u$ , l'énergie interne  $U$  et la masse  $m$  du système étudié, montrer que le système d'équations (4) se ramène à l'équation :

$$\frac{du}{dt} = \frac{R T}{M} \frac{1}{m} \frac{dm}{dt}. \quad (6)$$

Déduire alors de l'équation (5) la relation :

$$[f(T) - 1] \frac{dT}{dt} = \frac{T}{m} \frac{dm}{dt}. \quad (7)$$

On cherche à présent à estimer le débit  $D_{m,s}$  de gaz quittant le réservoir. Pour ce faire, on s'intéresse à la zone de l'espace dans laquelle se produit la fuite.



**Figure 3** - Fuite de CO<sub>2</sub> gazeux à travers l'orifice dans la paroi du réservoir

On considère le système  $\{\text{gaz dans la paroi du réservoir}\}$  (**figure 3**) et on suppose que :

- l'écoulement de gaz à travers l'orifice est **adiabatique** et **stationnaire** ;
  - la vitesse du gaz en entrée du système est nulle, elle est notée  $v_s$  en sortie ;
  - la température et la pression du gaz en entrée sont notées  $T$  et  $P$  (ce sont celles du gaz à l'intérieur du réservoir) ; en sortie, elles sont notées  $T_s$  et  $P_s = P_{\text{ext}}$ .
- Q8.** Montrer que le débit massique  $D_{m,s}$  de fluide à travers la section de l'orifice est donné par :

$$D_{m,s} = \frac{S v_s M P_{\text{ext}}}{R T_s} \quad (8)$$

où  $S$  désigne la section de passage du fluide à travers l'orifice.

- Q9.** On s'intéresse au système  $\{\text{gaz contenu dans la paroi du réservoir}\}$ . Justifier que l'équation (2) s'écrit alors  $\Delta h + \Delta e_c = 0$  et en déduire que la vitesse de sortie du CO<sub>2</sub> a pour expression :

$$v_s = \sqrt{2 \frac{R}{M} \int_{T_s}^T f(T) dT}. \quad (9)$$

**Q10. Méthode de calcul de  $T_s$**

- a) L'écoulement de fuite du gaz est supposé isentropique. Expliquer comment procéder expérimentalement pour rendre un écoulement pratiquement isentropique.
  - b) L'isentropicité permet d'obtenir l'équation admise  $\frac{C_{P,m}}{R} \frac{dT}{T} - \frac{dP}{P} = 0$ .
- En utilisant l'équation d'état des gaz parfaits, en déduire que  $T_s$  est solution de l'équation :

$$\int_T^{T_s} \frac{f(T)}{T} dT + \ln \left( \frac{m R T}{M V P_{\text{ext}}} \right) = 0. \quad (10)$$

## Partie II - Modélisation de la fuite d'un réservoir : traitement numérique

On suppose que le code suivant (**figure 4**) est écrit en préambule de toute réponse s'appuyant sur l'écriture d'un code Python. Il est inutile de le recopier. Toutes les variables définies sont globales et accessibles dans toutes les fonctions, sans avoir à être passées en argument.

```
from math import log, sqrt # logarithme népérien, racine carrée

# constantes générales
M = 44.01e-3      # Masse molaire du CO2 (kg/mol)
Pext = 1.013e5    # Pression extérieure (Pa)
R = 8.314          # Constante des gaz parfaits (J/K/mol)
S = 1.000e-4       # Section de l'orifice de fuite (m2)
Text = 293.0        # Température extérieure (K)
V = 1.000          # Volume de l'enceinte (m3)
K = S * Pext * sqrt(2 * M / R) # constante présente dans les équations du
→ modèle

# constantes pour le modèle A de f(T)
A1 = 8.303
A2 = -2810
A3 = 485.6

# initialisation
m0 = 11.00         # Masse initiale du CO2 dans l'enceinte (kg)
T0 = 473.2         # Température initiale du CO2 dans l'enceinte (K)
```

**Figure 4** - Préambule 1 Python

On s'intéresse ici à la fuite du gaz dans le réservoir et plus précisément à la température et la masse du gaz en sortie. Pour cela, on dispose des équations suivantes issues du modèle précédent :

$$g(T, T_s, m) = \int_T^{T_s} \frac{f(T)}{T} dT + \ln\left(\frac{m R T}{M V P_{\text{ext}}}\right) = 0 \quad (11)$$

et

$$\begin{cases} \frac{dm}{dt} = -\frac{K}{T_s} \sqrt{\int_{T_s}^T f(T) dT} & \text{avec } K = S P_{\text{ext}} \sqrt{\frac{2M}{R}} \\ \frac{dT}{dt} = \frac{T}{m[f(T) - 1]} \frac{dm}{dt} \end{cases} \quad . \quad (12)$$

On associe à ce modèle la fonction  $f(T) = C_{p,m}/R$  qui suit la loi du modèle A :

$$f(T) = A_1 + \frac{A_2}{T + A_3} \quad \text{avec } T \text{ en K et} \quad \begin{cases} A_1 = 8,303 \\ A_2 = -2\,810 \text{ K} \\ A_3 = 485,6 \text{ K} \end{cases} \quad . \quad (13)$$

Pour déterminer les paramètres du gaz en sortie de l'enceinte, il nous faut d'abord trouver la température de sortie  $T_s$ , ce que nous allons faire grâce à l'équation (11), en 4 étapes :

- définir la fonction  $f(T)$ ;
- calculer l'intégrale  $\int [f(T)/T] dT$  par la méthode des rectangles;
- définir la fonction  $g(T, T_s, m)$ ;
- résoudre l'équation  $g(T, T_s, m) = 0$  par la méthode de Newton pour obtenir  $T_s$ .

**Q11.** Écrire le code Python de la fonction  $f(T)$  associée à l'équation (13), qui prend en argument la température  $T$  et renvoie la valeur  $f(T)$  à cette température.

**Q12.** On souhaite déterminer numériquement l'expression  $\int_{T_1}^{T_2} \frac{f(T)}{T} dT$  en utilisant la méthode des rectangles sur un ensemble de  $N$  sous-intervalles.

- À l'aide d'une représentation graphique, justifier que l'expression  $\frac{T_2 - T_1}{N} \times \frac{f(T)}{T}$  correspond à l'aire d'un des rectangles dont un coin est commun à la courbe représentative de la fonction  $f(T)/T$ . Donner la valeur de  $N$  pour un calcul sur 100 sous-intervalles.
- En déduire le code de la fonction Python `integ1(T1, T2)` qui prend en argument 2 températures  $T_1$  et  $T_2$  correspondant aux bornes de l'intervalle d'intégration et renvoie la valeur numérique de l'intégrale de  $f(T)/T$  calculée par la méthode des rectangles, sur un ensemble de 100 sous-intervalles.

**Q13.** Donner le code Python de la fonction  $g(T, T_s, m)$  qui prend en argument les températures  $T$ ,  $T_s$  et la masse  $m$  et renvoie la valeur de la fonction  $g$  définie par l'équation (11). On pourra utiliser toutes les fonctions déjà définies.

**Q14.** Enfin, pour déterminer la valeur numérique de  $T_s$ , on va résoudre l'équation (11) par la méthode de Newton. Les variables  $T$  et  $m$  sont constantes pour l'étude de la fonction  $g(T, T_s, m)$ , qui se comporte donc comme une fonction d'une seule variable,  $T_s$ .

Pour implémenter la méthode de Newton, on utilise la suite  $T_{n+1} = T_n - g(T_n)/g'(T_n)$ , fabriquée à partir de la fonction  $g$  et de sa dérivée  $g'$ , dont on admet qu'elle converge vers un zéro de la fonction  $g$  qui est la valeur de  $T_s$  recherchée.

- À l'aide d'une représentation graphique la plus complète possible, illustrer quelques étapes du principe de convergence de la suite  $T_n$  vers un zéro de la fonction  $g$ .
- La fonction `chercheTs(T, m)` partiellement fournie ci-dessous renvoie, pour des valeurs connues des variables  $T$  et  $m$ , la valeur correspondante de  $T_s$  obtenue par résolution de l'équation (11) selon la méthode de Newton.

```
def chercheTs(T, m):
    "Résolution de l'équation g(T, Ts, m) = 0 par la méthode de
    → Newton"
    Ts = 300.0
    [instruction1]                      # À modifier
    while residu > 1e-10:
        gn = [instruction2]              # À modifier
        gpn = f(Ts) / Ts
        Tsold = Ts
        Ts = [instruction3]              # À modifier
        residu = abs(Ts - Tsold)
    return Ts
```

Indiquer par quelles instructions il convient de remplacer les séquences `[instruction1]`, `[instruction2]` et `[instruction3]` présentes dans le code ci-dessus.

Ensuite, pour résoudre le système d'équations (12), il faut :

- calculer l'intégrale  $\int f(T) dT$  qui intervient dans l'une d'elles ;
- déterminer les solutions des 2 équations différentielles par la méthode d'Euler.

**Q15.**

- a) Déterminer l'expression littérale de  $I = \int_{T_1}^{T_2} f(T) dT$ .
- b) Donner le code de la fonction `integ2(T1, T2)` retournant la valeur de  $I$  en fonction des arguments d'entrée  $T1$ ,  $T2$  et des variables globales  $A1$ ,  $A2$  et  $A3$ .

Pour résoudre une équation différentielle mise sous la forme  $y'(t) = f(t, y(t))$ , on utilise le schéma d'Euler suivant :

$$y(t + \Delta t) = y(t) + f(t, y(t)) \times \Delta t. \quad (14)$$

Ce schéma est implémenté au sein de l'**algorithme 1** présenté ci-dessous, que l'on souhaite appliquer pour résoudre le système d'équations (12) entre les dates  $t = 0$  s et  $t = 10$  s avec un pas de temps de  $\Delta t = 0,10$  s :

- Initialisation des variables `dt`, `N`, `T`, `m` et `P`
- Début d'une boucle avec test d'un ou plusieurs critères d'arrêt
  - calcul de `Ts`
  - calcul de `mp` et `Tp` par le schéma d'Euler
  - actualisation des variables
- Renvoie des grandeurs attendues

**Algorithme 1** - Résolution du système d'équations (12) par la méthode d'Euler

- Q16.** Préciser la signification et les valeurs initiales des variables présentes dans la phase d'initialisation de l'**algorithme 1** (on fera référence au préambule Python).
- Q17.** Préciser la signification des variables `mp` et `Tp` présentes dans la boucle de l'**algorithme 1**. Donner la syntaxe Python du calcul de `mp` et `Tp` respectant le schéma d'Euler proposé. On pourra faire appel à toute fonction précédemment définie, ainsi qu'au préambule Python.
- Q18.** Écrire le code Python de la fonction `Euler()` traduisant l'**algorithme 1** proposé ci-dessus. On pourra faire appel à toute fonction précédemment définie.
- Q19.** Intuitivement, quand la fuite s'arrêtera-t-elle en pratique ?

### Partie III - Développement de corrélations pour la capacité thermique à pression constante molaire du dioxyde de carbone

On s'intéresse maintenant au modèle  $B$  pour la fonction  $f(T) = C_{p,m}/R$ , dont on suppose qu'elle suit une loi polynomiale de la forme suivante :

$$f(T) = \sum_{i=0}^n B_i \left(\frac{T}{T_c}\right)^i \quad (15)$$

dans laquelle  $T_c$  est la température critique connue du CO<sub>2</sub>. Pour déterminer les coefficients réels  $B_i$  de ce polynôme de degré  $n$ , on dispose de 6 données expérimentales de la valeur de  $C_{p,m}/R$  en fonction de la température  $T$  en kelvin (voir **tableau 1**).

$T$	$C_{p,m}/R$
$T_1 = 100,0$	$C_1 = 3,513$
$T_2 = 500,0$	$C_2 = 5,367$
$T_3 = 1\,000$	$C_3 = 6,532$
$T_4 = 2\,000$	$C_4 = 7,257$
$T_5 = 3\,000$	$C_5 = 7,475$
$T_6 = 4\,000$	$C_6 = 7,586$

**Tableau 1** - Série de 6 données expérimentales pour le CO<sub>2</sub> gazeux

#### Préambule

- Q20.** Les valeurs des capacités thermiques données dans le **tableau 1** ont été mesurées par calorimétrie. Expliquer, à l'aide d'un schéma, comment réaliser expérimentalement cette mesure pour un liquide ou un gaz. Préciser comment la valeur de la capacité thermique est déduite de la mesure.
- Q21.** Montrer, par une analyse dimensionnelle, que le rapport  $C_{p,m}/R$  est sans dimension.

#### Développement du modèle polynomial

On souhaite déterminer les coefficients  $B_i$  d'un polynôme  $f(T)$  de degré  $n = 5$  donné par l'équation (15) dont la courbe représentative passe exactement par les 6 points expérimentaux.

- Q22.** Écrire l'expression littérale de l'équation que doivent vérifier les coefficients  $B_i$  (pour  $0 \leq i \leq 5$ ) du polynôme pour passer par le point expérimental ( $T_3, C_3$ ).
- Q23.** En déduire que l'on peut rassembler les données expérimentales sous la forme du produit matriciel :

$$\mathbf{Y} = \mathbf{M} \mathbf{B} \quad (16)$$

où  $\mathbf{Y}$  est le vecteur des mesures expérimentales de  $C_{p,m}/R$ ,  $\mathbf{B}$  celui des coefficients  $B_i$  (pour  $0 \leq i \leq 5$ ) du polynôme et  $\mathbf{M}$  une matrice carrée dont les coefficients ne dépendent que des valeurs expérimentales  $T_j/T_c$  (pour  $1 \leq j \leq 6$ ).

Donner l'expression d'une des lignes de la matrice  $\mathbf{M}$  en fonction de  $T_c$ ,  $T_j$  et  $C_j$  pour  $1 \leq j \leq 6$ .

- Q24.** Quelle opération mathématique faut-il effectuer sur la matrice  $\mathbf{M}$  pour accéder au vecteur  $\mathbf{B}$  ?

On suppose que le code suivant (**figure 5**) est écrit en préambule de toute réponse s'appuyant sur l'écriture d'un code Python. Il est inutile de le recopier. Toutes les variables définies sont globales et accessibles dans toutes les fonctions, sans avoir à être passées en argument.

```
import matplotlib.pyplot as plt # tracés
import numpy as np             # travail matriciel

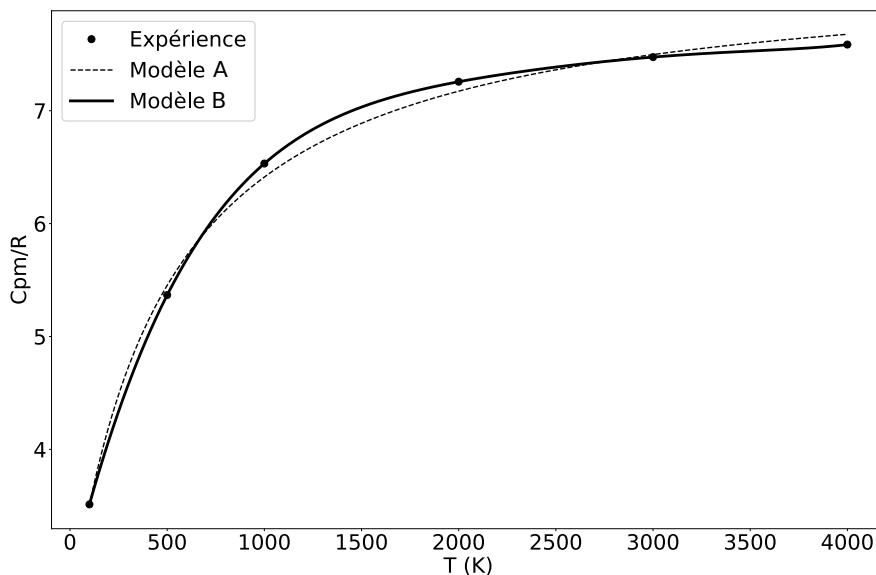
# Données expérimentales
T = np.array([100.0, 500.0, ...])      # température (K)
CpmR = np.array([3.513, 5.367, ...])    # Cpm/R

# modèle B
Tc = 304.2           # Température critique du CO2 (K)
Y = np.array(...)     # Vecteur Y du modèle B
M = np.array(...)     # Matrice M du modèle B
```

**Figure 5** - Préambule 2 Python

- Q25.** On suppose que la matrice  $M$  et le vecteur  $Y$  ont été entièrement saisis sous forme de tableaux numpy (voir **figure 5**). Donner le code Python permettant d'en déduire le vecteur  $B$  contenant les coefficients  $B_i$  à l'aide d'une fonction du module numpy.

Les points expérimentaux ainsi que les deux courbes associées aux modèles  $A$  et  $B$  ont été représentés sur la **figure 6**.



**Figure 6** - Confrontation des points expérimentaux et des deux modèles

- Q26.** Donner le code permettant de représenter le nuage des points expérimentaux ou le modèle  $A$  ou le modèle  $B$ , en précisant quel est le tracé choisi.
- Q27.** Commenter le graphe de la **figure 6**. Finalement, parmi les deux modèles proposés, lequel retiendriez-vous et pourquoi ?

## Partie IV - Traitement informatique des données expérimentales

On suppose que le code suivant (**figure 7**) est écrit en préambule de toute réponse s'appuyant sur l'écriture d'un code Python. Il est inutile de le recopier.

```
def tri(L):
    "Trie la liste L"
    n = len(L)
    if n <= 1:
        return L
    a = L[-1]
    L1 = []
    L2 = []
    for i in range(n - 1):
        if L[i] <= a:
            L1.append(L[i])
        else:
            L2.append(L[i])
    return tri(L1) + ... + ...      # À compléter
```

**Figure 7** - Préambule 3 Python

Les données expérimentales doivent éventuellement être triées. Pour cela, on utilise la fonction `tri(L)` qui prend en argument la liste à trier (voir **figure 7**).

**Q28.** Préciser, sans justification, s'il s'agit du *tri rapide*, du *tri par fusion* ou du *tri par insertion*. Expliquer pourquoi ce tri est récursif. Donner sa complexité temporelle dans le meilleur cas.

**Q29.** Les variables `a`, `L1` et `L2` ont une signification bien précise que l'on ne retrouve pas dans leur nom. Donner le nom usuel de la variable `a` et expliquer ce qui différencie les listes `L1` et `L2`. Proposer des noms plus pertinents pour ces 3 variables.

**Q30.** Compléter la dernière ligne de la fonction `tri`.

Après leur acquisition, les résultats expérimentaux sont stockés dans une base de données. Les molécules y sont identifiées par une clé primaire appelée `idcas`.

On s'intéresse aux **2 tables** suivantes contenues dans la base de données :

- la table `general` contenant des informations diverses sur les molécules ;
- la table `capa` contenant les données expérimentales de mesure du rapport  $C_{p,m}/R$  en fonction de la température pour différentes molécules.

<code>idcas</code>	<code>nom</code>	<code>symbole</code>	<code>Tc</code>	<code>Pc</code>	...
7727-37-9	diazote	N2	126.19	33.978	
124-38-9	dioxyde de carbone	CO2	304.18	73.825	
...					

**Table 1** - Extrait de la table `general`

<b>idcas</b>	<b>CpmR</b>	<b>T</b>
124-38-9	5.367	500.0
124-38-9	7.475	3000.0
7727-37-9	3.499	500.0
124-38-9	3.513	100.0
<hr/>		
...		

**Table 2** - Extrait de la table capa

- Q31.** Donner le code SQL permettant d'afficher le nom et le symbole de toutes les molécules.
- Q32.** Donner le code SQL permettant d'afficher toutes les informations de la table **capa** pour le dioxyde de carbone (repéré par son **idcas**).
- Q33.** Expliquer ce que renvoie la requête suivante :

```
SELECT g.symbole, count(*) AS nb
  FROM general AS g
  JOIN capa    AS c ON g.idcas = c.idcas
 GROUP BY c.idcas
```

- Q34.** Donner le code SQL permettant d'afficher, pour chaque molécule, son identifiant et la valeur moyenne de  $C_{p,m}/R$  pour les températures inférieures ou égales à 500,0 K.

## ANNEXE

### Bibliothèque numpy de Python

La bibliothèque numpy est importée de la façon suivante :

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

La création d'un tableau numpy à 2 lignes et 3 colonnes, appelé M, est réalisée ainsi :

```
>>> tab = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M)
[[1.5 2. 3.]
 [4. 5. 6.]]
```

L'inversion d'une matrice M carrée se fait à l'aide de la commande np.linalg.inv(M) :

```
>>> M = np.array([1.5, 2], [4, 5])
>>> N = np.linalg.inv(M)
>>> print(N)
[[-10., 4.],
 [ 8., -3.]]
```

### Bibliothèque matplotlib de Python

La bibliothèque matplotlib est importée de la façon suivante :

```
>>> import matplotlib.pyplot as plt
```

La courbe représentative de la fonction  $x \mapsto x^2$  peut être tracée de la façon suivante :

```
>>> X = [0, 1, 2, 3, 4, 5]
>>> Y = [0, 1, 4, 9, 16, 25]
>>> plt.plot(X, Y)
>>> plt.show()
```

### Mémento de commandes sql

<b>SELECT ...</b>	- <i>sélection d'attributs</i>
<b>FROM ...</b>	- <i>choix de tables</i>
<b>JOIN ... ON ...</b>	- <i>jointure entre tables</i>
<b>WHERE ...</b>	- <i>condition</i>
<b>GROUP BY ...</b>	- <i>groupements</i>
<b>HAVING ...</b>	- <i>condition sur les agrégations</i>
<b>ORDER BY ...</b>	- <i>classement</i>
<b>AVG(...), MIN(...), MAX(...), SUM(...), COUNT(...)</b>	- <i>commandes d'agrégation</i>
<b>AS</b>	- <i>renommage</i>

**FIN**









## Epreuve d'Informatique et Modélisation de Systèmes Physiques

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.

### **CONSIGNES :**

- Composer lisiblement sur les copies avec un stylo à bille à encre foncée : bleue ou noire.
- L'usage de stylo à friction, stylo plume, stylo feutre, liquide de correction et dérouleur de ruban correcteur est interdit.
- Remplir sur chaque copie en MAJUSCULES toutes vos informations d'identification : nom, prénom, numéro inscription, date de naissance, le libellé du concours, le libellé de l'épreuve et la session.
- Une feuille dont l'entête n'a pas été intégralement renseignée, ne sera pas prise en compte.
- Il est interdit aux candidats de signer leur composition ou d'y mettre un signe quelconque pouvant indiquer sa provenance.

**Il est conseillé de répartir votre temps de composition comme suit :**

- 1h30 pour lire et traiter la partie modélisation (partie II).
- 2h30 pour lire et traiter la partie informatique (partie III).

**À rendre en fin d'épreuve avec la copie un document réponse.**

# Étude d'un système autofocus d'appareil photo numérique

## I Présentation

Nous nous intéressons dans ce sujet à l'étude de différents méthodes permettant de réaliser l'auto-focus sur les appareils photos numériques. L'auto-focus consiste à régler de manière automatique la netteté de l'image avant d'effectuer la prise de vue.

Les applications numériques seront données avec 1 chiffre significatif, sauf contre ordre.

## II Modélisation : principe de la méthode de l'auto-focus.

Un formulaire d'optique géométrique est rappelé en fin de cette partie.

Un appareil photo est modélisé par une lentille mince convergente ( $L$ ), l'objectif, de focale  $f'_0 = 10\text{ cm}$  et un plan récepteur ( $P$ ) placé orthogonalement à l'axe optique. Ce plan récepteur est de taille  $20\text{ mm} \times 30\text{ mm}$  et contient 6 mégapixels.

Lorsque le réglage de l'appareil est optimal l'image de l'objet à photographier se trouve sur le plan ( $P$ ). Sinon, il convient de modifier la position de ce plan.

Prenons l'exemple d'un objet réduit à un point objet  $A_0$  qui donne un point image  $A$ . En cas de défaut de réglage on a la situation décrit sur la figure 1.

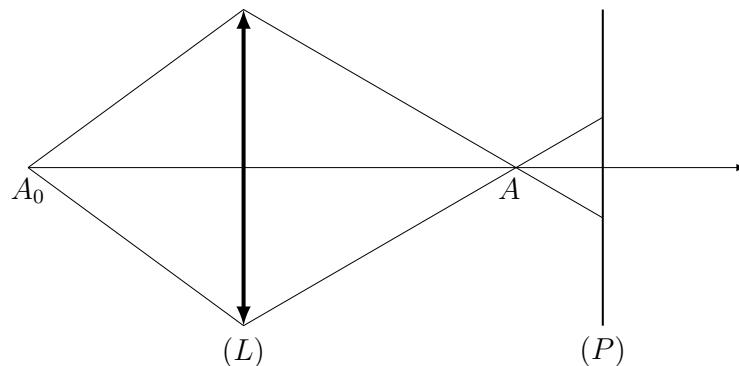


FIGURE 1 – Défaut de réglage

Sur ( $P$ ) se forme alors une tache à la différence d'un point.

Il faut donc déplacer le plan ( $P$ ) d'une certaine distance pour obtenir une image nette. Pour cela la méthode la plus rapide consiste à calculer cette distance à partir de la différence d'ordonnées des points inférieurs et supérieurs de la tâche.

Il y a cependant une difficulté du fait que l'on obtient la même tache que ( $P$ ) soit placé devant ou derrière l'image.

Nous allons étudier un dispositif astucieux qui permet de calculer algébriquement le déplacement à opérer partant d'un défaut de réglage.

## II.1 Mise au point.

Dans cette partie on attend, pour chaque question, une expression littérale puis une valeur numérique.

On souhaite photographier un objet de hauteur  $h = 10 \text{ cm}$ , transverse à l'axe de l'objectif et situé à une position  $x_0 = -2 \text{ m}$ , l'origine étant prise au centre de la lentille ( $L$ ). L'axe optique est orienté de la gauche vers la droite.

- Q1.** À quelle distance  $d'$  du centre de la lentille ( $L$ ) faut-il placer ( $P$ ) pour avoir une image nette (ceci définit le plan  $P_0$ ) ? Donner la taille  $h'$  de l'image.

On se place dans le cas où l'objet précédent se ramène à un point situé sur l'axe toujours à la position  $x_0 = -2 \text{ m}$ . ( $P$ ) est placé à une distance  $\delta = +0,5 \text{ cm}$  derrière  $P_0$ . La lentille a un rayon  $a = 5 \text{ cm}$  (figure 2). On observe alors une tache lumineuse sur ( $P$ ).

- Q2.** Faire un schéma et tracer les rayons qui parviennent à l'extrémité de cette tâche.

Déterminer le rayon  $a'$  de la tâche lumineuse formée sur ( $P$ ).

NB : le rayon de la lentille  $a$  est défini par la hauteur de lentille par rapport à l'axe optique (figure 2).

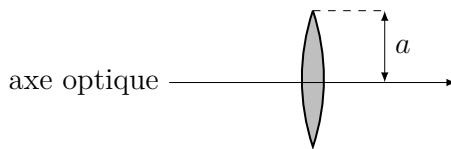


FIGURE 2 – Illustration de la lentille

- Q3.** Après avoir déterminé la taille d'un pixel, supposé carré, donner un critère sur  $a'$ , puis sur  $\delta$  pour que l'image transmise par le capteur soit nette.

## II.2 Principe simplifié de l'auto focus.

Dans un souci de simplification on considère un objet réduit à un point objet situé sur l'axe de la lentille ( $L$ ) qui donne un point image noté  $A$ .

Pour déterminer si  $A$  se trouve ou non sur ( $P$ ) on utilise deux lentilles annexes ( $L_1$ ) et ( $L_2$ ), situées à une distance  $d_0$  de ( $P$ ), de focale  $f'$  et de rayon  $a$  auxquelles sont associées des capteurs plans ( $P_1$ ) et ( $P_2$ ) situés à une distance  $d$  de chaque lentille. ( $P_1$ ) (resp ( $P_2$ )) est conjugué de ( $P$ ) par ( $L_1$ ) (resp ( $L_2$ )). Attention : en réalité les deux lentilles sont désaxées par rapport à l'axe optique de ( $L$ ), les rayons étant déviés par des miroirs. Ici, on considère simplement que tout se passe comme si ( $P$ ) est transparent.

Le schéma est représenté figure 3.

- Q4.** Exprimer  $d$  en fonction de  $d_0$  et  $f'$ .

- Q5.** On se place dans le cas où  $A$  est sur ( $P$ ). On note  $A_1$  son image par ( $L_1$ ).

Faire un schéma représentant  $A$ ,  $A_1$ ,  $F'_1$  (le foyer image de ( $L_1$ )) et les rayons issus de  $A$  passant par les bords inférieurs et supérieurs de ( $L_1$ ).

- Q6.** Déterminer l'ordonnée  $y_1$  de  $A_1$  en prenant l'origine  $O'$  située sur l'axe ( $Ox$ ) (on pourra préalablement déterminer l'ordonnée par rapport à l'axe optique de la lentille ( $L_1$ )).

En déduire l'expression de  $y_2$  l'ordonnée de  $A_2$  l'image de  $A$  par ( $L_2$ ) en prenant également l'origine en  $O'$ .

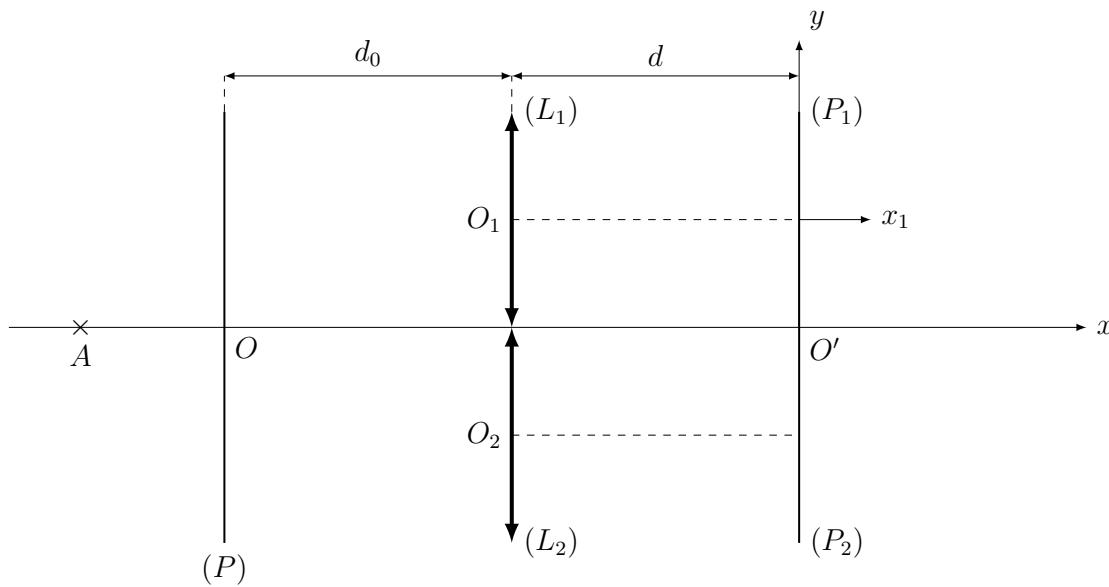


FIGURE 3 – Principe des lentilles de l’auto-focus

- Q7.** Calculer  $\Delta\Phi_0 = y_1 - y_2$  (appelé différence de phase, même si c'est une longueur) que l'on exprimera en fonction de  $d_0$ ,  $d$  et  $a$ .

On se place dans le cas où  $A$  est avant le plan  $(P)$  (cas de la figure 3) et on pose  $\overline{OA} = p$  (mesure algébrique). On note à nouveau  $A_1$  l'image de  $A$  par  $(L_1)$ .

- Q8.** Construire  $A_1$  sur le document réponse.

On note  $x_1$  l'abscisse de  $A_1$  mesurée sur l'axe  $(O_1x_1)$  que l'on ne cherchera pas à exprimer et qui sera donc considérée comme une donnée. Déterminer son ordonnée  $y_1$  mesurée à partir de l'axe  $(Ox)$  en fonction de  $a$ ,  $x_1$ ,  $d_0$  et  $p$ .

- Q9.** Sur le document réponse, tracer les rayons issus de  $A$  passant par les extrémités des lentilles  $(L_1)$  et  $(L_2)$ .

On obtient donc une tâche lumineuse. On note  $y_s$  l'ordonnée du point supérieur de la tâche lumineuse sur  $(P_1)$  et  $y_i$  l'ordonnée du point inférieur de cette tâche, l'origine étant en  $O'$  sur l'axe  $(Ox)$ .

- Q10.** Exprimer  $y_s$  en fonction de  $y_1$ ,  $x_1$  et  $d$  et montrer que  $y_i = 2a \left(1 - \frac{d}{x_1}\right) + \frac{dy_1}{x_1}$ .

Le principe de la méthode est de mesurer ce qui est appelé « la différence de phase » définie par  $\Delta\Phi = y_s - y'_s$  où  $y'_s$  est l'ordonnée du point supérieur de la tache lumineuse sur  $(P_2)$ . Ce qui revient à déterminer  $\Delta\Phi = y_s + y_i$ .

- Q11.** Pourquoi a-t-on  $y_i = -y'_s$ ? Montrer que  $\Delta\Phi = 2a + \frac{2ad}{d_0 - p}$ .

- Q12.** Évaluer la différence de différence de phase entre le cas où la mise au point n'est pas réalisée et celui où elle l'est. Soit  $\Delta^2\Phi = \Delta\Phi - \Delta\Phi_0$ .

Le principe de l'auto focus consiste donc à mesurer la différence de phase et d'en déduire  $p$ . Il suffit ensuite de déplacer la lentille  $(L)$  afin de faire coïncider  $A$  sur  $(P)$ .

Le déplacement de la lentille se fait au moyen d'un moteur pas à pas.

**Q13.** On mesure  $\Delta^2\Phi = 0,66 \text{ cm}$ . Donner la distance de laquelle on doit translater ( $P$ ) pour obtenir une image nette. On précisera la direction de la translation. On a  $d = 2f'$  et  $d_0 = 2f'$  avec  $f' = 10 \text{ cm}$  et  $a = 3 \text{ cm}$ .

Pour un objet étendu orthogonal à l'axe, on peut montrer que la différence de phase due à chaque point source de l'objet est identique.

Le principe de la méthode peut alors être généralisé.

### II.3 Moteur pas à pas

Un moteur pas à pas est constitué d'un rotor qui peut tourner autour d'un axe ( $\Delta$ ). On place un ensemble de bobines plates, ( $B1$ )...( $B6$ ) qui sont susceptibles de créer un champ magnétique lorsqu'on les alimente (figure 4). Celles-ci ont alimentées par un courant continu dans le sens des orientations, ou non alimentées.

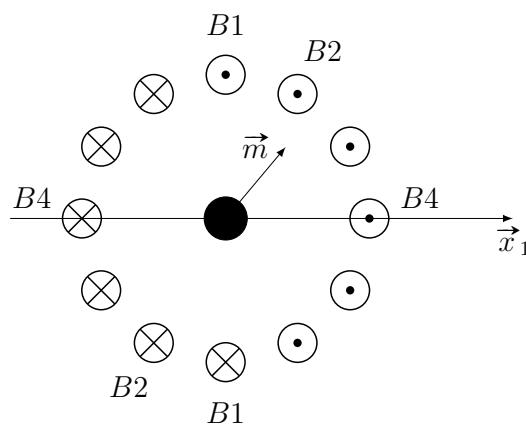


FIGURE 4 – Vue en coupe : 6 bobines au centre desquelles se trouve le rotor.

En commandant les bobines les unes après les autres, on fait tourner le moteur de pas successifs. Le rotor est en général caractérisé par un moment magnétique  $\vec{m}$  qui peut tourner librement autour de l'axe  $\Delta$  orthogonal au plan de la figure.

**Q14.** Déterminer la direction et sens du champ magnétique créé par la bobine  $B_1$  en un point de son plan.

Donner la direction d'un moment magnétique à l'équilibre dans un champ magnétique uniforme (à justifier).

La bobine  $B_1$  est alimentée seule ; on coupe l'alimentation et on alimente la bobine  $B_2$ . Représenter les positions successives du moment magnétique en se limitant aux deux bobines considérées.

#### Moment magnétique.

**Q15.** On considère une spire plane rectangulaire  $ABCD$  de cotés  $a$  et  $b$ , parcourue par un courant  $i$ . Donner l'expression de son moment magnétique.

À un aimant on associe également un moment magnétique. Pourquoi ? Quel est l'ordre de grandeur d'un moment magnétique associé à un aimant ?

Nous étudions le cas où le rotor est un cadre sur lequel sont bobinées  $N$  spires parcourues par un courant  $i$ .

### Moment des forces de Laplace.

La spire précédente peut tourner autour de l'axe ( $\Delta$ ) = ( $Oz$ ). Elle est placée dans un champ magnétique uniforme  $\vec{B}$  sur le cadre ce qui donne vue de dessus la figure 5(a) où  $\vec{n}$  est le vecteur normal à la spire. Dans le plan du cadre on a le schéma de la figure 5(b).

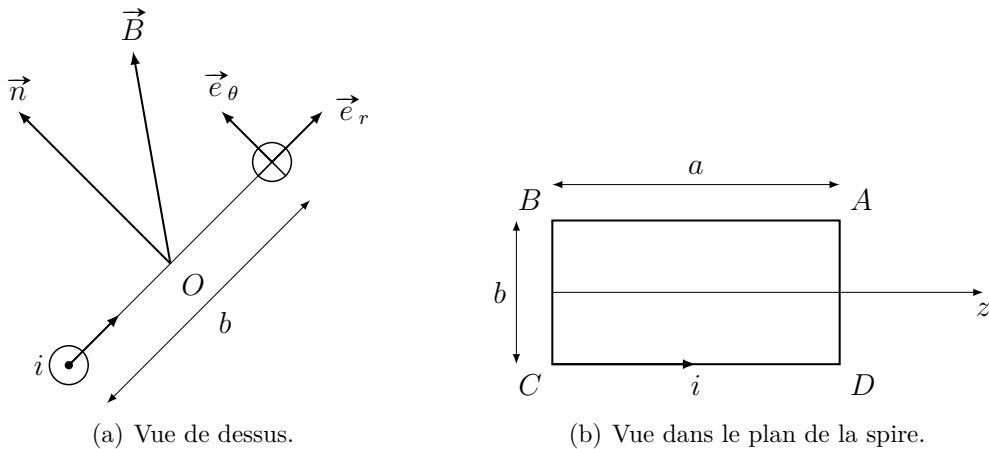


FIGURE 5 – Modélisation de la spire.

On pose  $\vec{B} = B_r \vec{e}_r + B_\theta \vec{e}_\theta$  avec la base locale définie sur la figure 5(a).

**Q16.** Déterminer le moment des forces qui s'exerce sur la spire par rapport à l'axe ( $\Delta$ ).

Retrouver ce résultat à partir de la forme vectorielle du moment donnée en cours.

### Équation électrique

**Q17.** En utilisant la conversion de puissance  $P_{\text{Laplace}} + P_{\text{fem induite}} = 0$ , déterminer la force électromotrice induite dans le cadre lors de son mouvement.

Le cadre de résistance  $R$  est parcouru par un courant constant  $i_0$ . Déterminer la force électro-motrice du générateur qui doit alimenter le cadre.

### Rappel des lois de Descartes.

On considère un point objet  $A$  situé sur l'axe d'une lentille mince de focale  $f'$ ,  $A'$  le point image.

On a les relations :

$\frac{1}{OA'} - \frac{1}{OA} = \frac{1}{f'}$  et la relation de grandissement transverse :  $\frac{\overline{A'B'}}{\overline{AB}} = \frac{\overline{OA'}}{\overline{OA}}$  où  $AB$  est un objet transverse et  $A'B'$  son image.

### III Partie informatique

On supposera dans cette partie que tous les modules nécessaires ont été importés. Par exemple, toutes les fonctions de `numpy` ou celles de `matplotlib.pyplot`. Les fonctions importées peuvent être utilisées sans préfixe (`plot`, `arange`...).

Pour réaliser l'autofocus, il existe deux méthodes principales :

- la mesure du contraste,
- la mesure de détection de phase.

#### III.1 Mesure du contraste

Le principe de cette méthode repose sur le fait qu'une image bien mise au point présente un maximum de contraste. Plus précisément l'écart de valeur entre les pixels est la plus grande.

En fonctionnement, l'appareil mesure le contraste, puis par tâtonnement, va déplacer l'objectif jusqu'à détecter le maximum.

**Représentation d'une image** Une image matricielle est représentée sous la forme d'une matrice. Chaque élément de la matrice correspondra à un pixel qui est généralement représentée par un liste de 3 entiers naturels, représentant les composantes rouge, vert et bleu; c'est le codage RVB.

Chaque valeur est représentée par un entier allant de 0 à 255. On suppose, dans cette partie, travailler avec un appareil photo doté d'un capteur de 48 MPixels, c'est-à-dire que la photo sera composé de 48 millions de pixel.

On suppose que le stockage des photos se fait en mode RAW sans compression, c'est-à-dire que l'on stocke directement ce que récupère le capteur.

L'image sera représentée en mémoire dans une variable `I`, de type liste de liste de liste Python classique ou de type `numpy.ndarray`. Dans les deux cas, on accédera aux données RVB d'un pixel de coordonnée  $(i,j)$ , par l'instruction `I[i][j]`. Le candidat choisira librement le type qu'il préfère manipuler.

**Q18.** Préciser l'espace mémoire nécessaire pour stocker la valeur d'une composante, puis celle d'un pixel et enfin celle d'une image en Mo (= 1000 ko) ou Mio (=1024 kio).

**Conversion en niveau de gris** La première étape de la détection du contraste est de convertir l'image en niveau de gris pour n'obtenir qu'une valeur par pixel.

Cette transformation s'opère en plusieurs étapes :

— chaque composante R, V et B qui ne sont pas linéaires en terme d'intensité lumineuse sur le rendu, sont d'abord transformée dans un espace linéaire. Une composante C sera transformée selon la définition suivante :

$$\bullet \quad C_{\text{lin}} = \frac{C}{12,92} \text{ si } C \leq 0,04045 ;$$

$$\bullet \quad C_{\text{lin}} = \left( \frac{C + 0,055}{1,055} \right)^{2,4} \text{ sinon.}$$

— la valeur du niveau de gris  $Y_{\text{lin}}$  en échelle linéaire sera calculée à partir des valeurs linéarisées par :  $Y_{\text{lin}} = 0,2126R_{\text{lin}} + 0,7152V_{\text{lin}} + 0,0722B_{\text{lin}}$

— on repasse dans l'espace non linéaire avec le calcul suivant :

$$\bullet \quad Y = 12,92Y_{\text{lin}} \text{ si } Y_{\text{lin}} \leq 0,0031308 ;$$

$$\bullet \quad Y = 1,055Y_{\text{lin}}^{1/2,4} - 0,055 \text{ sinon.}$$

**Q19.** Écrire une fonction `Clinear(val)`, qui prend en argument une valeur de l'espace non linéaire et qui renvoie la valeur linéarisée.

**Q20.** Écrire une fonction `Y(pix)` qui prend en argument une liste de trois valeurs correspondant à un pixel au format RVB et qui renvoie la valeur  $Y$  du niveau de gris dans l'espace non linéaire.

**Q21.** Écrire une fonction `NiveauxGris(I)` prenant en argument une image  $I$  au format RVB et qui renvoie une image de même dimension en niveau de gris.

Pour détecter le contraste d'une image en niveau de gris, on va comparer pour chaque pixel les valeurs autour de celui-ci. Plus l'écart est grand (ce qui est le cas quand l'image est nette), plus les pixels autour du pixel de référence ont une valeur différente ; on calcule d'une certaine manière la dérivée en chaque pixel.

Cette opération est réalisée par un filtre de Sobel. Partant d'une image  $I$ , on extrait les pixels

autour du pixel  $(i,j)$  sous la forme d'une matrice  $3 \times 3$ , notée  $I_e = \begin{pmatrix} I_{i-1,j-1} & I_{i-1,j} & I_{i-1,j+1} \\ I_{i,j-1} & I_{i,j} & I_{i,j+1} \\ I_{i+1,j-1} & I_{i+1,j} & I_{i+1,j+1} \end{pmatrix}$ .

On réalise ensuite une convolution entre cette matrice et la matrice de filtration. On définit la convolution entre deux matrices  $A$  et  $B$  de taille  $3 \times 3$  par  $A \otimes B = \sum_{i=0}^2 \sum_{j=0}^2 A_{ij} B_{ij}$ .

On réalise une filtration selon les deux directions de l'image. On donne les deux matrices

de filtration :  $G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$  et  $G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$ .

Enfin on calcule une norme euclidienne des deux convolutions dont on prend la partie entière pour obtenir la valeur du contraste  $c$  du pixel  $(i,j)$  :  $c = \text{Ent}(\sqrt{(I_e \otimes G_x)^2 + (I_e \otimes G_y)^2})$ .

Pour terminer, on calcule le contraste de cette manière pour chaque pixel intérieur à l'image (on ne calcule rien sur les bords pour simplifier), puis on réalise la moyenne des valeurs pour obtenir l'indice de contraste de référence  $c_{ref}$ .

**Q22.** Écrire une fonction `convolution(A,B)` prenant en argument deux matrices de taille  $3 \times 3$  et qui renvoie la valeur du produit de convolution.

**Q23.** Écrire une fonction `contraste_pixel(I,i,j)` prenant en argument une image  $I$  au format niveaux de gris et les coordonnées du pixel  $(i,j)$  qui renvoie la valeur du contraste défini précédemment par la quantité  $c$ .

**Q24.** Écrire une fonction `contraste(I)` prenant en argument une image  $I$  au format niveau de gris et qui renvoie la valeur du contraste de référence  $c_{ref}$ .

Pour régler la netteté de l'image, l'objectif est déplacé à l'aide d'un moteur pas à pas. L'objectif est déplacé d'un pas, une photo est prise, la valeur du contraste de référence est calculée puis comparée à la valeur obtenue au pas précédent ; l'algorithme s'arrête dès que la valeur du contraste de référence diminue. Le moteur pas à pas recule d'un pas pour retourner à la position précédente où le contraste était maximale.

Pour cela, on utilise une fonction `position_objectif(val)` qui prend en argument un entier `val` allant de 0 à 1000 correspondant à la position en pas demandée et qui déplace l'objectif à cette position. Une fonction `prise()` permet de prendre un cliché et de retourner une image au format RGB.

**Q25.** Écrire une fonction `reglage`, dont les arguments et les valeurs de retour sont à définir, répondant au comportement décrit en partant de la position 0 en pas. On supposera que le maximum de contraste existe.

### III.2 Détection de phase

La méthode par détection de phase (décrise à la partie I) consiste à mesurer une petite partie de l'image par deux capteurs différents. Quand les deux mesures sont identiques, l'image est nette. En calculant la différence de phase entre les deux mesures, il est possible de calculer directement la valeur dont doit se déplacer l'objectif.

On suppose que l'on dispose de deux capteurs de longueurs 100 pixels (dont les valeurs seront stockées dans des listes) et que l'on a exactement la même séquence de valeurs mais décalées de quelques pixels. La différence de phase est le nombre de pixels (donc le nombre d'indice de décalage entre les deux listes) permettant de retrouver les mêmes séquences de pixels.

L'objectif est donc de comparer deux listes de valeurs, de même dimension, extraites des listes de données des deux capteurs. Trois cas peuvent se poser en fonction de la valeur du décalage noté `dec` et sont illustrés sur la figure 6. Les sous listes extraites sont grisées.

	<code>dec=0</code>	<code>dec=2</code>	<code>dec=-3</code>
Capteur 1 :	a   b   c   d   e   f   g   h   i   j	a   b   c   d   e   f   g   h   i   j	a   b   c   d   e   f   g   h   i   j
Capteur 2 :	a   b   c   d   e   f   g   h   i   j	y   z   a   b   c   d   e   f   g   h	d   e   f   g   h   i   j   k   l   m

FIGURE 6 – Illustration du décalage (sur l'alphabet)

**Q26.** Écrire une fonction `extraction(L1,L2,dec)` prenant en argument deux listes `L1` et `L2` ainsi qu'une valeur entière de décalage et qui renvoie deux sous-listes à comparer de longueur `len(L1)-dec`, conformément à aux règles présentées ci-dessus. On supposera que les deux listes `L1` et `L2` sont de même taille.

**Q27.** Écrire une fonction `comparaison(L1,L2)` (n'utilisant pas la comparaison interne de Python entre les listes) prenant en argument deux listes `L1` et `L2` qui renvoie `True` si les listes sont identiques et `False` sinon.

On suppose que le décalage est compris entre  $-80$  et  $80$ .

**Q28.** Écrire une fonction `recherche_decalage(L1,L2)` prenant en argument deux listes `L1` et `L2` et qui renvoie la valeur du décalage ou `None` s'il n'existe pas.

**Q29.** Évaluer la complexité de la fonction `recherche_decalage(L1,L2)` en prenant en compte le nombre de comparaison en fonction de  $n$  la taille des listes et de  $m$  le nombre de décalage maximal à prendre en compte (161 dans notre exemple) dans le meilleur et dans le pire des cas.

En pratique, il n'y a pas de raisons que les deux capteurs CCD mesurent exactement les mêmes valeurs décalées. Les valeurs mesurées décalées seront approximativement les mêmes. Pour résoudre ce problème, on va calculer une erreur quadratique moyenne entre les deux sous-listes et puis chercher le minimum en fonction du décalage.

Pour deux sous-listes  $L1$  et  $L2$  de taille  $n$ , l'erreur sera définie par :

$$\text{erreur} = \frac{1}{n} \sum_{i=0}^{n-1} (L1_i - L2_i)^2$$

**Q30.** Écrire une fonction `erreur(L1,L2)` qui retourne l'erreur quadratique définie ci-dessus.

**Q31.** Écrire une fonction `recherche_decalage_2(L1,L2)` qui cherche le minimum de l'erreur pour des décalages de  $-80$  à  $80$  et qui retourne la valeur de ce décalage.

### III.3 Comparaison des deux méthodes

Deux méthodes de réglage de l'objectif ont été introduites dans les parties précédentes.

**Q32.** Décrire en 5 lignes maximum les avantages et les inconvénients de ces deux méthodes et laquelle vous semble être la plus pertinente.

### III.4 Commande du moteur pas à pas

L'objectif de mise au point est motorisé par un moteur pas à pas bipolaire commandé en « demi pas ».

Les paragraphes suivants permettent de décrire le principe de fonctionnement et la commande du moteur mais leurs compréhensions détaillées n'est pas utile pour répondre à la suite des questions.

Le principe du moteur pas à pas consiste à positionner le rotor sur lequel se trouvent des aimants permanents polarisé nord ou sud régulièrement espacé. Sur le stator se trouvent deux demi-bobines. La commande des demi-bobines se fait de tel sorte que le rotor va s'aligner sur le stator par pôle opposée (un nord en face d'un sud et un sud en face d'un nord).

Prenons un exemple simplifié de la figure 7 avec d'un moteur avec deux bobines ( $AB$  pour la première et  $CD$  pour la seconde). Les bobines sont en deux parties situées de part et d'autres du rotor. Sur le rotor se trouve un aimant permanent avec deux pôles.

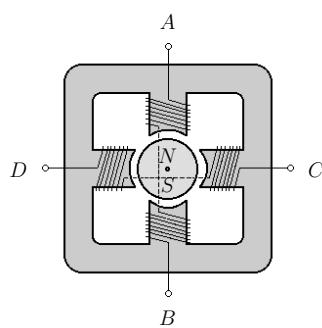


FIGURE 7 – Structure élémentaire d'un moteur pas à pas

Les deux demi-bobines étant positionné à  $90^\circ$ , la commande en demi pas consiste à faire des :

- pas entier : les bobines sont alimentées l'un après l'autre
- demi pas : les deux bobines sont alimentées et le rotor s'aligne entre deux bobines

La figure 8 montre les différentes positions associées à la séquence d'alimentation présentée dans la suite.

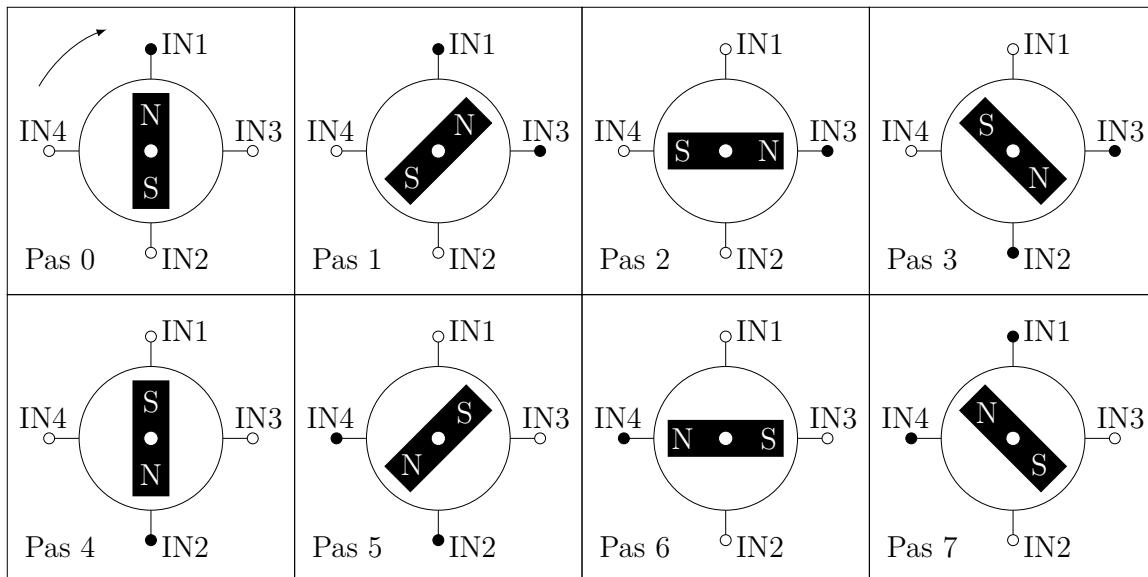


FIGURE 8 – Séquence d'alimentation des bobines : —● IN à 1 ; —○ IN à 0

La séquence d'alimentation du préactionneur des bobines (non détaillé ici) est donné par le chronogramme de la figure 9 et donne les 8 cas d'alimentation possibles pour obtenir la rotation dans le sens positif. Pour obtenir la rotation dans le sens négatif, il suffit de parcourir la séquence d'alimentation dans l'autre sens.

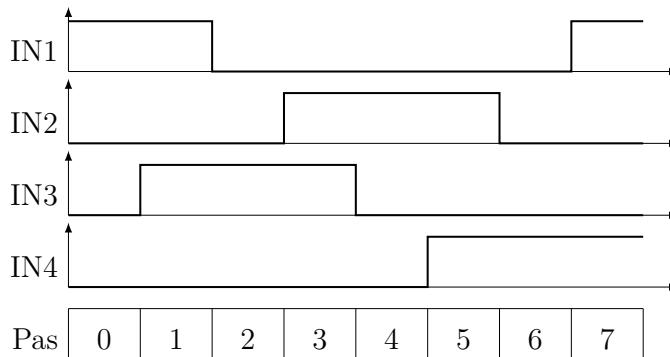


FIGURE 9 – Chronogramme d'alimentation des 4 entrées.

On se propose pour la suite de programmer le déplacement pas à pas de l'objectif. Pour cela, on supposera que les variables `IN1`, `IN2`, `IN3` et `IN4` ont été déclarées de manière globale et représentent les sorties à piloter sur un microcontrôleur. Pour piloter ces sorties, on utilisera la fonction `modif_sortie(IN, valeur)` où `IN` sera la sortie à modifier et `valeur` la valeur de type booléen souhaitée pour la sortie.

Exemple : `modif_sortie(IN4, True)` passera la sortie `IN4` à 1 et `modif_sortie(IN4, False)` la passera à 0.

On remarquera que la séquence d'alimentation présente les 8 premier pas, les autres pas étant obtenus à un modulo près.

**Q33.** Écrire une fonction `faire_un_pas_positif(pas_actuel)` qui prend en argument la valeur du pas courant qui modifie l'état des sorties `IN1` à `IN4` et qui renvoie la nouvelle valeur du pas. On veillera à ne changer l'état que des sorties nécessaires.

On supposera l'existence d'une variable globale `pas_courant` comprise entre 0 et 1000 correspondant à la position courante du moteur pas à pas, ainsi qu'une fonction `faire_un_pas_négatif(pas_actuel)` qui permet de réaliser un pas négatif.

**Q34.** Écrire une fonction `position_objectif(pas)` qui prend en argument la position en pas à atteindre à partir de la position courante réalisant le déplacement demandé pas à pas. Il peut y avoir plus d'un pas à effectuer. On vérifiera que le pas demandé est atteignable (compris entre 0 et 1000).

## IV Gestion des photographies

Une société emploie des photographes pour réaliser des photographies, qu'ils déposent ensuite sur un site et qui sont gérées à l'aide d'une base de données.

La base de données est composée de plusieurs tables, notamment :

- table `PHOTOS` contenant notamment les attributs :
  - `id` : identifiant de la photographie, clé primaire, de type entier ;
  - `date` : date à laquelle a été prise la photographie, de type texte, au format YYYYMMDD.  
Par exemple, le 22 juin 2021 sera stocké sous la forme 20210622 ;
  - `heure` : heure à laquelle a été prise la photographie, de type texte ;
  - `idp` : identifiant du photographe, de type entier ;
  - `orientation` : orientation de l'appareil lors de la prise de photographie, de type entier ;
  - ...
- table `PHOTOGRAPHES` contenant notamment les attributs :
  - `id` : identifiant du photographe, clé primaire, de type entier ;
  - `nom` : nom du photographe, de type texte ;
  - `prenom` : prénom du photographe, de type texte ;
  - ...

**Q35.** Donner la définition d'une clé primaire.

**Q36.** Donner une requête en SQL permettant de sélectionner les identifiants de toutes les photos prises le 11 novembre 2018.

**Q37.** Donner une requête en SQL permettant de sélectionner les noms et prénoms des photographes ayant pris des photographies le 11 novembre 2018.

**Q38.** Donner une requête en SQL permettant de sélectionner les noms, prénoms des photographes et heure de la prise de vue des photographies prises le 11 novembre 2018.

Afin d'afficher correctement sur le site les clichés sélectionnés, il faut réorienter l'image en fonction de l'orientation initiale du cliché. Les réorientations automatiques classiques à gérer sont le pivotement de 90 degrés dans le sens horaire ou le sens trigonométrique ainsi que la rotation de 180 degrés.

**Q39.** Illustrer sur un schéma les nouvelles coordonnées d'un pixel de coordonnées  $(i,j)$  après une rotation de 180 degrés. Écrire une fonction `rotation_180(image)` qui prend en argument une image au format RVB et qui renvoie une nouvelle image pivotée de 180 degrés.

**Q40.** Illustrer sur un schéma les nouvelles coordonnées d'un pixel de coordonnées  $(i,j)$  après une rotation de 90 degrés. Écrire une fonction `rotation_90(image)` qui prend en argument une image au format RVB et qui renvoie une nouvelle image pivotée de 90 degrés dans le sens trigonométrique.









Modèle CMEN-DR v2 ©EXATECH

A horizontal row of 20 empty white squares, each outlined in black, arranged in a single row.



**Prénom(s) :**



**Numéro  
Inscription :**

 Né(e) le :  /  / 

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

(Remplir cette partie à l'aide de la notice)

**Concours / Examen :** .....

**Section/Specialité/Série :** .....

**Epreuve :** .....

Matière : ..... Session : .....

- Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.
  - Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.
  - Numéroté chaque PAGE (cadre en bas à droite de la page) et placer les feuillets dans le bon sens et dans l'ordre.
  - Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.
  - N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.

060

# Document réponse

## À rendre en fin d'épreuve

B

Tournez la page S.V.P.

1 / 4

**NE RIEN Ecrire DANS CE CADRE**

Document réponse partie modélisation, questions 8 et 9

