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 renseigné, 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.

Présentation de la problématique

SUIVI DE LA CONSOMMATION D'EAU D'UNE HABITATION

La consommation d'eau, au même titre que la consommation d'électricité ou de gaz, représente une part non négligeable du budget de nombreux ménages, et contribue de façon significative à leur empreinte environnementale. La facturation de ces consommations étant souvent annuelle, leur suivi tout au long de l'année présente un intérêt aussi bien économique qu'écologique.

1. Principe du suivi

Le suivi de la consommation d'une habitation est réalisé à l'aide d'un compteur d'eau (Figure 1). Il s'agit d'un dispositif monté en série sur l'arrivée d'eau d'une habitation, qui comporte un dispositif mécanique mis en rotation par le passage de l'eau. Si le compteur a été correctement posé et étalonné, à chaque tour de la partie mobile correspond un volume d'eau connu. Un ensemble d'engrenages et de roues à cliquet permet d'afficher le volume consommé sur un cadran.

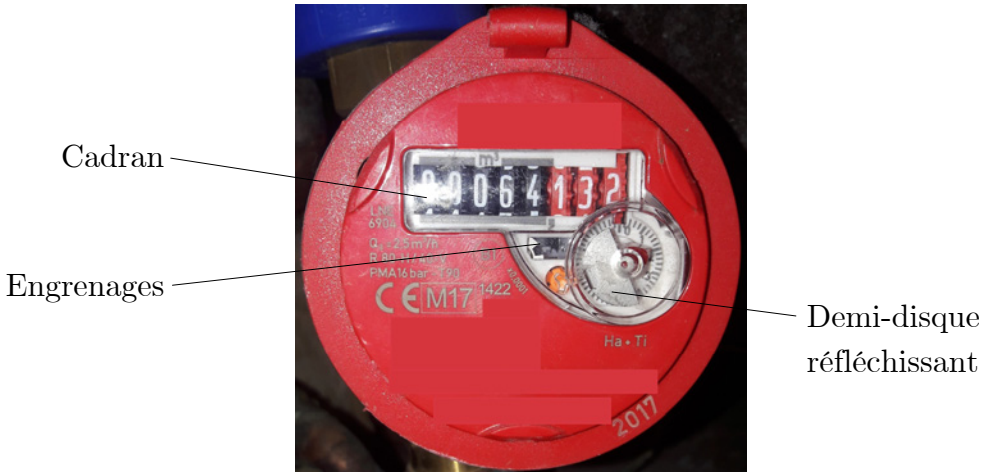


FIGURE 1 – Compteur d'eau installé dans une habitation

Le relevé fréquent d'un compteur étant une opération fastidieuse, son automatisation est d'un intérêt certain pour faciliter le suivi. On peut pour cela utiliser des compteurs dits "intelligents" ou "communicants", ou encore instrumenter un compteur d'eau classique. Dans les deux cas, le principe est le même : il s'agit de compter les tours d'un demi-disque métallique tournant avec la partie mobile au moyen d'un détecteur optique (phototransistor) ou magnétique (interrupteur à lame souple). Ces dispositifs, alimentés sous une tension continue, fournissent l'un comme l'autre une tension présentant une impulsion par tour de la partie mobile (Figure 2).

La mesure en temps réel de la consommation d'eau d'une habitation peut ainsi être réalisée en comptant les impulsions issues du capteur au moyen, par exemple, d'un microcontrôleur. En

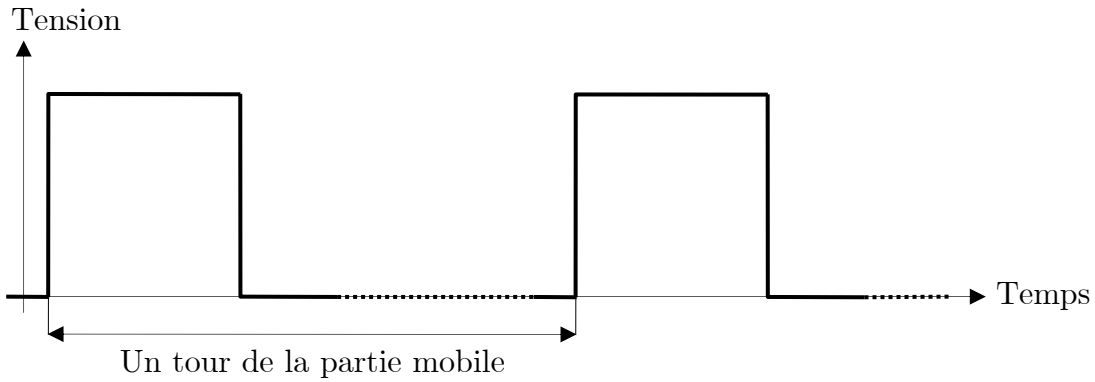


FIGURE 2 – Schématisation du signal en sortie du détecteur

pratique, le signal obtenu peut être affecté par divers phénomènes indésirables (rebond des interrupteurs à lame souple, saturation progressive des phototransistors) et il est donc nécessaire de mettre préalablement en forme les impulsions pour éviter les erreurs de comptage.

2. Travail demandé

Ce sujet comporte deux parties.

La première partie (durée conseillée : 1h30) porte sur l'étude d'un montage électronique, utilisant un amplificateur opérationnel, destiné à mettre en forme les impulsions avant comptage.

La seconde partie (durée conseillée : 2h30) porte sur le stockage des impulsions dans le microcontrôleur, leur traitement informatique (sur un ordinateur classique) pour reconstruire les débits et volumes consommés, la transmission cryptée des mesures vers un serveur distant mettant en oeuvre une base de données, et l'exploitation de celle-ci afin d'accéder à distance au suivi de la consommation.

Ces deux parties sont indépendantes.

Tournez la page S.V.P.

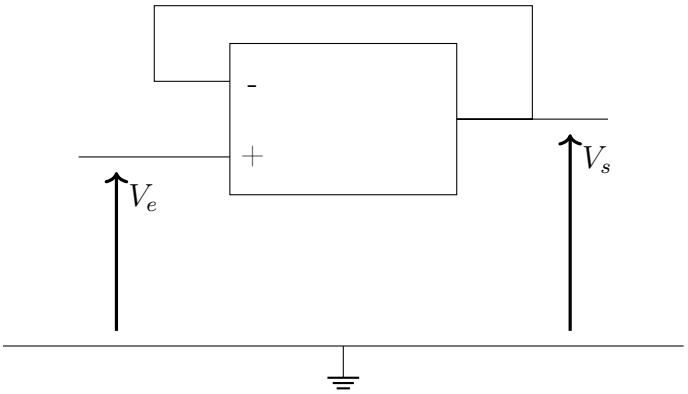
PREMIERE PARTIE
COMPTEUR D'IMPULSIONS ANALOGIQUE

A. GENERALITES SUR LES ALI (15%)

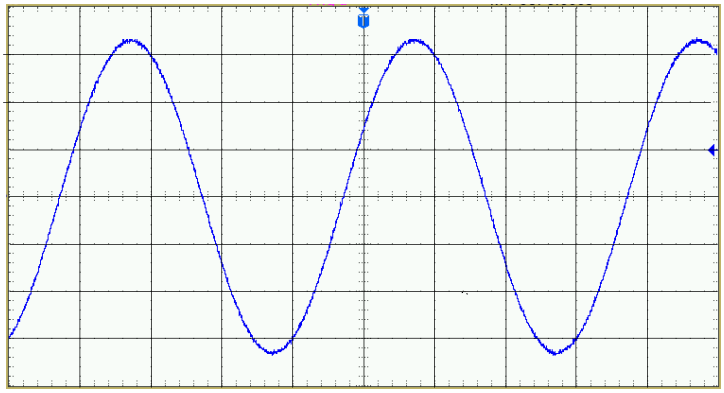
Dans cette partie, on considère un ALI alimenté en $+15/-15$ *Volts par une alimentation à point milieu. On admettra que les tensions de saturation haute et basse sont* $+/-15$ *Volts.*

1. Représenter la tension de sortie en fonction de la tension différentielle d'entrée, en indiquant clairement les ordres de grandeur considérés (on indiquera la partie correspondant au régime linéaire et celle correspondant au régime saturé).

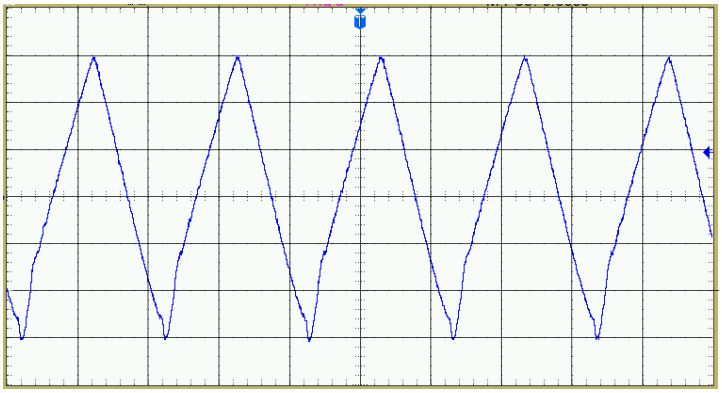
2. On s'intéresse au montage représenté ci-dessous. Montrer que $V_s = V_e$. Comment s'appelle ce montage ? Quel est son intérêt ? (on considérera le gain de l'ALI comme infini)



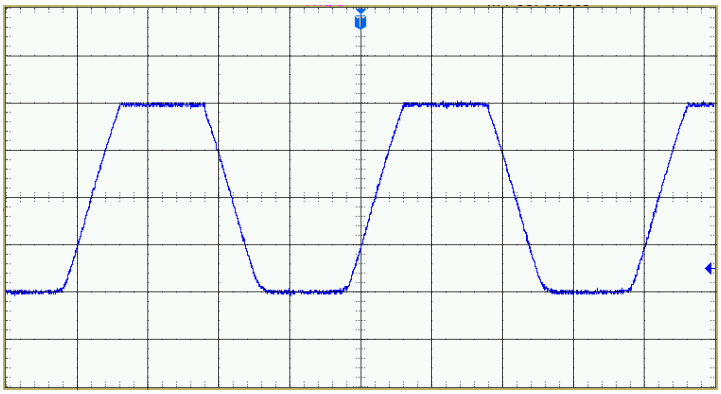
3. On alimente ce montage avec en entrée la tension dont l'oscillogramme est donné ci-dessous. Les réglages sont $2V/div$ et $100\mu s/div$, quelles sont les caractéristiques de cette tension ? Peut-on raisonnablement penser observer la même chose en sortie ?



4. Toutes choses égales par ailleurs, on augmente la fréquence et on observe en sortie la tension ci-dessous. Les réglages sont $2V/div$ et $1\mu s/div$. Quelle caractéristique de l'ALI est ainsi mise en évidence ? Evaluer sa valeur numérique.



5. On revient à la fréquence de la question 3, et on ajoute une résistance de charge $R_0 = 50\Omega$ entre la sortie et la masse. Les réglages sont $2V/div$ et $100\mu s/div$. Quelle caractéristique de l'ALI est ainsi mise en évidence ? Evaluer sa valeur numérique.



6. Donner le schéma d'un montage amplificateur non inverseur utilisant un ALI et 2 résistances. Etablir l'expression du gain de ce montage.

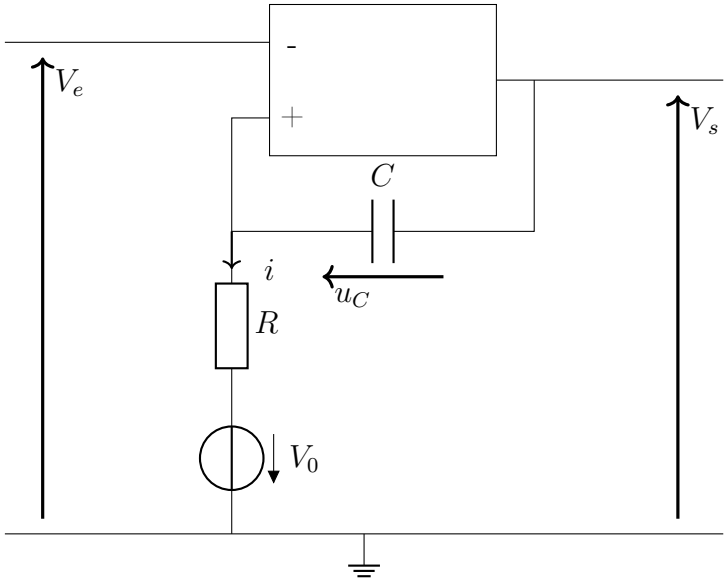
7. Proposer des valeurs pour les résistances pour avoir des gains de 10, 100 et 1000. Jusqu'à quelle valeur de gain peut-on aller avec un tel montage avec une tension d'entrée continue ? Avec une tension d'entrée sinusoidale de fréquence $10kHz$? (de simples ordres de grandeur sont attendus)

8. On alimente ce montage, en prenant un gain de 10, avec la tension d'entrée de la question 3. Dessiner l'allure de la tension attendue en sortie.

Tournez la page S.V.P.

B. COMPTEUR D'IMPULSIONS (25%)

Le montage ci-dessous permet de réaliser un compteur d'impulsions analogique. L'ALI est alimenté en +V_{cc} / -V_{cc} avec $V_{cc} = 7$ Volts par une alimentation à point milieu. Dans toute cette partie, il fonctionne en régime saturé et les tensions de saturation +V_{sat} et -V_{sat} sont considérées comme égales aux tensions d'alimentation +V_{cc} et -V_{cc}. On considérera que le temps de réponse de l'ALI est négligeable (on bascule de +/-V_{sat} à son opposé de manière instantanée). On prend pour ce montage $R = 10\text{k}\Omega$, $C = 650\text{nF}$ et $V_0 = 1\text{V}$.



9. Quelle est la valeur de i en régime stationnaire ? De V_+ ? Justifier le fait que $V_s = -V_{sat}$ en régime stationnaire si $V_e = 0$. Quelle est alors la valeur de u_C ?

10. A $t = 0$, on envoie en entrée une impulsion *très brève* de durée Δt : V_e passe *instantanément* de 0 à -5 volts puis, un temps Δt plus tard, repasse à 0 (toujours *instantanément*). Représenter cette impulsion.

11. Que signifie *très brève* pour Δt ? (Avec quelle grandeur caractéristique du circuit faut-il comparer). En déduire une condition sur Δt. Expliquer pourquoi la valeur de u_C ne varie quasiment pas entre $t = 0$ et $t = \Delta t$.

12. Expliquer pourquoi le passage de V_e de 0 à -5 volts à $t = 0$ fait basculer la sortie à $+V_{sat}$, et pourquoi le retour à 0 à $t = \Delta t$ ne provoque pas un autre basculement.

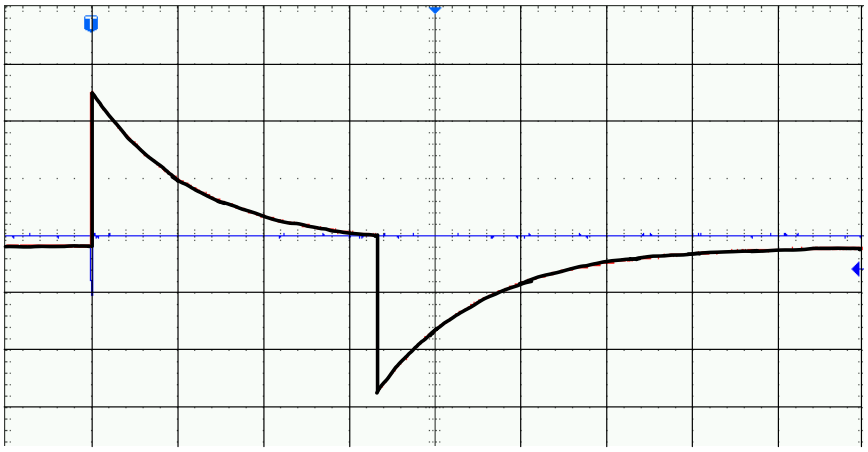
13. Montrer que, suite au basculement (on prend $t = 0$ au moment du basculement), u_C évolue de la manière suivante :

$$u_C = 2V_{sat}\, e^{(-t/\tau)} - (V_{sat} + V_0)$$

Donner l'expression de τ ainsi que son sens physique.

14. A quel instant t_1 la sortie va-t-elle repasser en saturation basse ? Donner l'expression en fonction de R, C, V_{sat} et V_0 et faire l'application numérique. On donne $ln(14) \simeq 2,6$.

On donne ci-dessous un enregistrement de la tension V_+ suite à une impulsion. L'instant $t = 0$ est décalé à une division après la gauche de l'écran. Les réglages sont $5V/div$ et $5ms/div$.



15. Expliquer les deux phases observées dans l'évolution de cette tension (expliquer en particulier la valeur minimale prise par V_+).

16. Sur la même échelle de temps que l'enregistrement précédent, représenter l'évolution des tensions u_C et V_s.

Le montage reçoit en entrée des impulsions périodiques, toujours de largeur Δt et avec une période T, dans le but de pouvoir mesurer la fréquence f de ces impulsions.

17. Quelle condition doit respecter T vis-à-vis des caractéristiques du montage ?

18. Représenter sur un même graphe, sans échelle mais en respectant les conditions des questions 11 et 17, les tensions V_e et V_s (sur au moins une période T et au plus deux) . Indiquer clairement les trois temps caractéristiques.

19. Donner l'expression de la valeur moyenne $< V_s >$ de V_s en fonction de V_{sat}, f et t_1.

20. Quel type de filtre pourrait-on utiliser en aval du montage précédent pour obtenir $< V_s >$? Préciser comment il doit être branché (faire un schéma) et proposer des valeurs pour ses composants si $f = 10\ Hz$.

21. On souhaite obtenir avec un voltmètre une tension directement proportionnelle à f. Expliquer comment compléter le montage en aval du filtre pour obtenir ce résultat.

Tournez la page S.V.P.

SECONDE PARTIE
EXPLOITATION INFORMATIQUE DES MESURES

Dans cette partie, les fonctions et programmes attendus devront être écrits en langage Python (sauf, bien entendu, lorsqu'une requête SQL est demandée). On prendra garde à la lisibilité du code, et notamment aux indentations. Le code devra être documenté. Toute fonction définie dans le sujet pourra être utilisée, même sans avoir été codée.

C. ESTIMATION DES DEBITS ET VOLUMES (30%)

La sortie du dispositif précédent est raccordée à une entrée logique d'un microcontrôleur. À chaque fois qu'une impulsion se produit, le microcontrôleur mémorise l'instant auquel l'impulsion a été détectée. Cette information est ensuite utilisée pour reconstituer l'évolution temporelle du volume d'eau et du débit consommés. On se propose ici d'implanter partiellement cette reconstitution.

C.1. Justification du format des données

Si l'on fait l'hypothèse que le débit volumique Q traversant le compteur est constant dans l'intervalle de temps séparant deux impulsions, alors on l'obtient par la relation :

$$Q = \frac{c}{\tau} \tag{1}$$

où $c = 7\times10^{-5}$ m^3 est le volume d'eau consommé à chaque impulsion, et τ la durée séparant deux impulsions. Le débit maximal pouvant traverser le compteur ne dépasse pas $Q_{\max} = 2,5\ \mathrm{m}^3\cdot\mathrm{h}^{-1}$.

22. Estimer la plus petite valeur numérique que prend τ.

Le microcontrôleur effectue le chronométrage et le stockage en mémoire des instants avec une résolution limitée. On suppose donc que la durée τ est connue avec une incertitude $\delta\tau$. Par suite, le débit Q est déterminé avec une incertitude δQ. On souhaite que l'incertitude relative $\delta Q/Q$ soit d'au plus 1% en valeur absolue (toujours sous l'hypothèse simplificatrice d'un débit constant).

23. Exprimer, en valeur absolue, $\delta Q/Q$ en fonction de $\delta\tau/\tau$. Estimer alors la valeur maximale que doit avoir $\delta\tau$ pour que la condition ci-dessus soit satisfaite au débit $Q_{\max}$.

Les instants des impulsions sont représentés au format *timestamp*, défini comme le nombre de secondes écoulées depuis le 1er janvier 1970 à minuit. Par exemple, lors d'une acquisition effectuée début 2021, des impulsions ont été détectées aux *timestamp* suivants, exprimés en secondes :

$$1609825201,145 \quad 1609825202,948 \quad 1609825206,532 \quad \ldots$$

Le microcontrôleur utilisé pour l'acquisition stocke les nombres sur au plus 32 bits. Il propose des types entier et flottant définis comme suit :

— les entiers peuvent prendre des valeurs comprises entre -2^{31} et $2^{31} - 1$;

— les flottants sont au format simple précision de la norme IEEE 754, qui prévoit une mantisse codée sur 23 bits et un exposant codé sur 8 bits.

Quelle que soit la valeur obtenue précédemment, on admet que les instants des impulsions doivent être mémorisés à 1 ms près. Pour les applications numériques, on pourra considérer que $2^{10} \approx 1000$.

24. Justifier qu'il n'est pas possible de stocker directement les instants des impulsions dans le microcontrôleur avec la résolution demandée, que ce soit avec le stockage entier (en exprimant les instants en millisecondes) ou avec le stockage flottant.

C.2. Récupération des données

Compte tenu de la limitation mise en évidence dans la partie précédente, on décide de représenter les instants des impulsions de la façon suivante :

— on synchronise le microcontrôleur à l'aide d'une horloge extérieure : on repère un instant où le *timestamp* est un nombre *entier* de *secondes*, et on relève la valeur entière correspondante ;

— on relève ensuite, à chaque impulsion, le nombre entier de *millisecondes* écoulées depuis la synchronisation.

Ces valeurs sont stockées en mémoire et, à intervalles de temps réguliers, transmises à un ordinateur où elles sont écrites dans un fichier texte. Par exemple, les *timestamp* donnés dans la partie précédente :

$$1609825201,145 \quad 1609825202,948 \quad 1609825206,532 \quad \ldots$$

pourraient être représentés par le fichier texte suivant :

```
1609825200
1145
2948
6532
...
```

On souhaite, à partir de ce fichier, construire en Python une liste de flottants contenant les *timestamp*. Supposons par exemple que le fichier précédent s'appelle `log.txt` et se limite aux quatre premières lignes ci-dessus. On exécute alors les instructions suivantes, dont on donne le résultat affiché dans l'interpréteur :

```
>>> f = open("log.txt", "r")
>>> lignes = f.readlines()
```


Tournez la page S.V.P.

```
>>> f.close()
>>> print(lignes)
['1620115200\n', '1145\n', '2948\n', '6532\n']
```

On précise que la séquence de caractères `\n` indique le passage à la ligne et ne perturbe pas les fonctions de conversion. On précise également que le type flottant de Python est au format double précision sur la plupart des ordinateurs, ce qui suffit largement à stocker les instants des impulsions avec la résolution exigée.

25. À l'aide de l'exemple ci-dessus, écrire une fonction `recup(nomFichier)` qui prend pour argument une chaîne de caractères `nomFichier` donnant le nom du fichier texte contenant les instants stockés au format décrit dans cette partie, et renvoie une liste de flottants contenant les instants des impulsions exprimés en seconde.

C.3. Première reconstruction : débit constant par morceaux

On suppose dans cette partie que l'on dispose de la liste des instants des impulsions, notés T_i pour $0 <= i <= n$ (il y en a donc $n+1$). Par construction, cette liste est croissante au sens strict, deux impulsions ne pouvant pas survenir simultanément.

On envisage tout d'abord l'hypothèse d'un débit constant par morceaux. On pose alors, pour tout i tel que $0 \leq i \leq n-1$:

$$q_i = \frac{c}{T_{i+1} - T_i} \tag{2}$$

où c est le volume consommé entre deux impulsions.

26. Écrire une fonction `debits1(lImp,c)` qui prend pour arguments la liste `lImp` des instants des impulsions et un flottant `c`, et renvoie la liste des débits définis par l'équation (2).

À partir de ces deux listes, on souhaite reconstruire l'évolution temporelle du débit volumique consommé sur l'intervalle de temps correspondant à l'enregistrement, de la façon suivante :

$$q(t) = \begin{cases} 0 & \text{si} \quad t < T_0 \quad \text{ou} \quad t \geq T_{n-1} \\ q_i & \text{si} \quad T_i \leq t < T_{i+1}\,,\ 0 \leq i < n-1 \end{cases} \tag{3}$$

27. Écrire une fonction `indice(lImp,t)` qui prend pour arguments la liste `lImp` des instants T_i des impulsions et un flottant `t`, tels que `t` est compris entre le premier et le dernier élément de `lImp` (premier inclus, dernier exclu), et recherche l'entier i tel que $T_i \leq t < T_{i+1}$. On ne demande pas de vérifier la condition précédente (ce sera fait lors de l'appel de la fonction).

L'évaluation précédente est menée en m instants de calcul utilisés pour le tracé et l'interpolation, stockés dans une liste `lTemps`.

28. Écrire une fonction `temporel(lQ,lImp,lTemps)` qui prend pour arguments trois listes `lQ` (débits moyens calculés par l'équation (2)), `lImp` (instants des impulsions) et `lTemps` (instants de calcul), et renvoie la liste des débits moyens aux instants de `lTemps`. Ceux-ci ne sont pas forcément compris entre le premier et le dernier élément de `lImp`. Lorsqu'ils le sont, on appellera la fonction `indice`.

29. Donner et justifier la complexité asymptotique de la fonction `temporel` en fonction du nombre m d'instants de calcul et du nombre n d'impulsions dans le pire des cas.

En pratique, la liste des impulsions et celle des instants de calcul sont toutes deux triées par ordre croissant. En optimisant le parcours des deux listes, il est possible de réduire la complexité des calculs.

30. Justifier qu'il n'est pas possible de réduire la complexité tout en conservant l'appel à la fonction `indice`. Expliquer, en deux ou trois phrases éventuellement complétées par un schéma, comment passer en revue les éléments des deux listes pour minimiser le nombre d'opérations si l'on suppose que les deux tailles m et n sont voisines. Donner la complexité obtenue pour $m = n$.

En plus du débit volumique, on souhaite pouvoir visualiser le volume consommé, défini comme l'intégrale temporelle du débit volumique. Pour tout j tel que $0 \leq j < m$, on note t_j les instants de calcul, $q_j = q(t_j)$ les débits et $v_j = v(t_j)$ les volumes consommés. Les instants de calcul étant supposés régulièrement espacés, on notera $\Delta t = t_{j+1} - t_j$.

31. Donner la relation de récurrence permettant d'exprimer, par la méthode des trapèzes, v_j en fonction de v_{j-1}, q_j, q_{j-1} et Δt.

32. Écrire une fonction `volumes(lTemps,lQT,v0)` qui prend pour arguments deux liste `lTemps` (instants de calcul) et `lQT` (débits aux instants de `lTemps`) et un flottant `v0`, et renvoie la liste des volumes consommés, calculés par la méthode des trapèzes. Le volume initial (avant le premier instant de `lTemps`) sera pris égal au paramètre `v0`.

On dispose d'un relevé d'impulsions contenu dans un fichier nommé `2021-01-31.txt`. On souhaite faire appel aux fonctions précédentes pour extraire les impulsions stockées dans ce fichier, déterminer les débits et préparer les deux listes "temporelles" nécessaires au tracé.

33. Écrire une suite d'instructions appelant entre autres les fonctions précédentes de sorte à :
- placer le contenu du fichier `2021-01-31.txt` dans une liste `lImp` ;
- construire la liste des débits `lQ` associés, en litres par seconde, si à chaque impulsion correspond 0,07 litre ;
- générer une liste `lTemps` de 1000 flottants régulièrement espacés, allant de la première à la dernière valeur de `lImp`, tous deux inclus ;
- construire la liste des débits `lQT` (tirés de `lQ`) aux instants de `lTemps` ;
- en déduire la liste `lVT` des volumes exprimés en litres, toujours aux instants de `lTemps`, pour un volume initial nul.

A partir des listes venant d'être construites, on peut tracer l'évolution temporelle du débit et du volume, donnée Figure 3.

Tournez la page S.V.P.

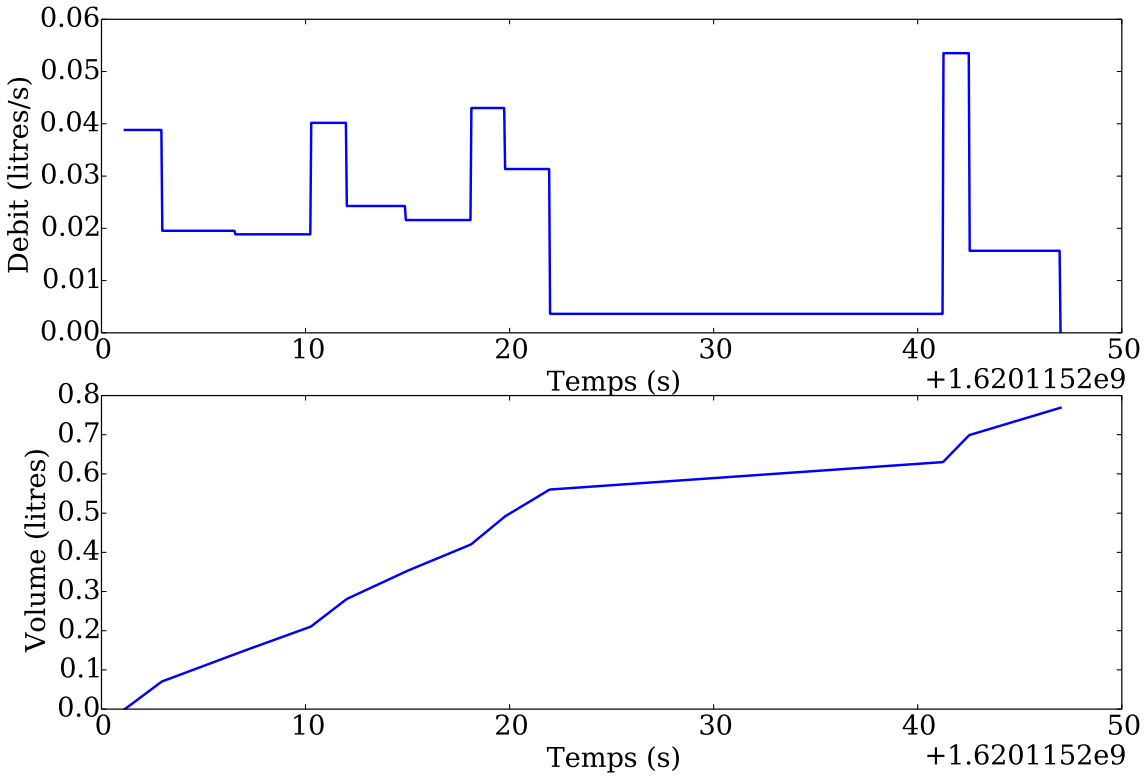


FIGURE 3 – Évolution temporelle du débit et du volume consommé

Une limitation de cet algorithme est qu'il ne permet pas, en l'état, de détecter l'absence de consommation d'eau pendant un laps de temps donné. En effet, un tel cas de figure se manifeste par la présence de deux impulsions très espacées, et l'algorithme estime alors, de façon logique, un débit volumique constant et très faible entre ces deux impulsions. On se propose donc de modifier le mode d'évaluation des débits pour permettre la détection des périodes sans consommation, selon le principe schématisé Figure 4.

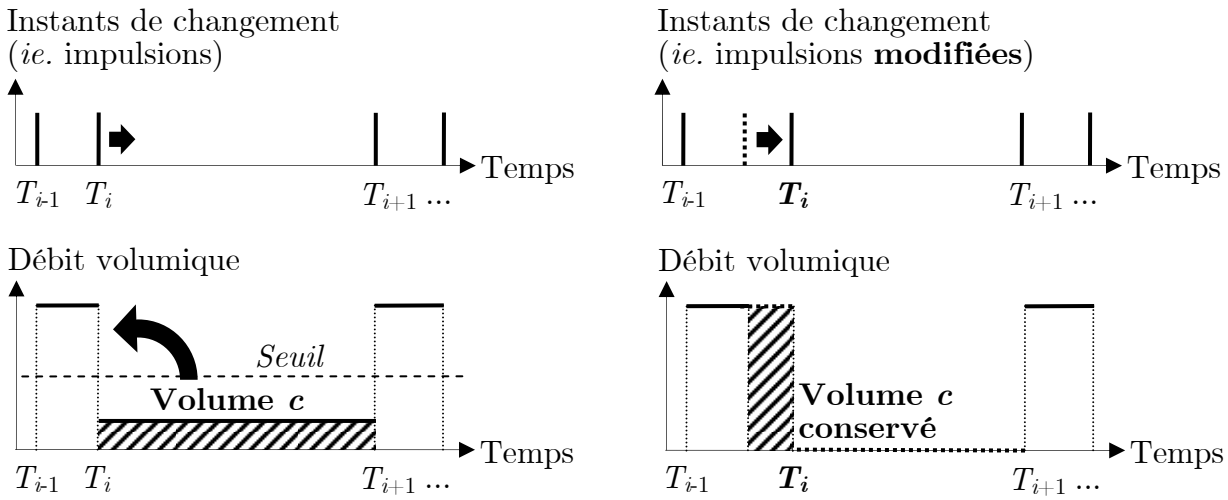


FIGURE 4 – Principe de la détection des périodes sans consommation

Pour cela, on se donne un seuil, ou débit minimal admissible ; le débit q_i calculé par l'équation (2) ne sera retenu que s'il est supérieur à ce seuil. Si le débit q_i est inférieur au seuil, on le met à zéro, et on retarde l'instant T_i de l'impulsion précédente de sorte à conserver le volume total consommé. On remarquera que le volume consommé entre deux impulsions est toujours égal à c.

Enfin, on n'effectuera ce processus que sur *un échantillon à la fois* : si plusieurs débits vérifiant le critère précédent se suivent, on ne décalera que la *première* impulsion intervenant dans le calcul de ces débits, sans modifier les suivantes. Un tel cas de figure signifie en effet qu'il existe soit un faible débit continu, soit une consommation intermittente de faibles volumes ; il n'est pas possible de distinguer ces deux cas en l'état, mais il n'est pas non plus souhaitable de les négliger.

La fonction ne devra pas modifier la liste des impulsions. On rappelle que lorsque l'argument d'une fonction est une liste, alors la fonction peut modifier son argument. Pour éviter cela, il est possible de copier une liste avec une instruction du type `Lcopie = L[:]`.

34. Écrire une fonction `debits2(Limp,c,seuil)` prenant pour arguments la liste `Limp` des instants des impulsions et deux flottants `c` et `seuil`, qui renvoie un tuple contenant :
- la liste des instants des impulsions modifiés comme indiqué ci-dessus (l'argument `Limp` ne doit pas être modifié),
- et la liste des débits calculés comme indiqué ci-dessus.

Comme indiqué précédemment, on ne modifiera pas plus d'une impulsion d'affilée (mais on pourra modifier plusieurs impulsions dans la liste).

Pour l'exemple précédent, le résultat obtenu est donné Figure 5.

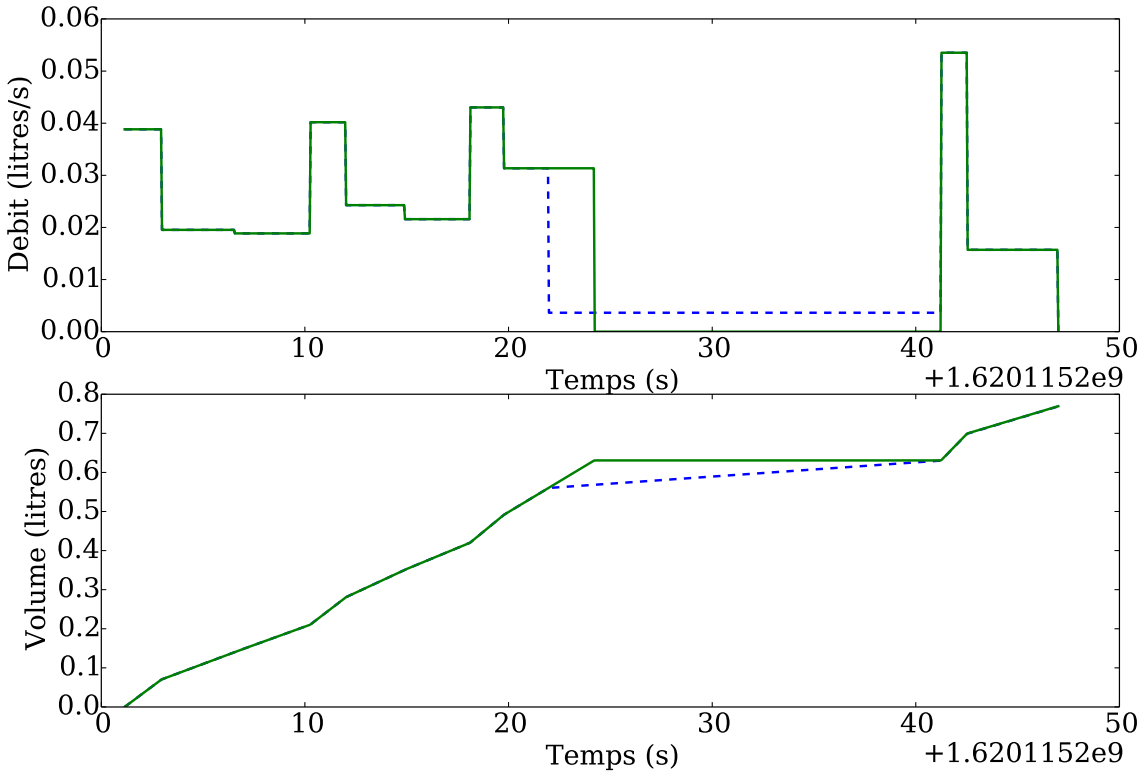


FIGURE 5 – Évolution temporelle avec prise en compte des périodes sans consommation

Tournez la page S.V.P.

D. TRANSMISSION DES DONNÉES ET CONSULTATION A DISTANCE (30%)

Les données sur la consommation élaborées dans la partie précédente font ensuite l'objet d'un chiffrement, puis sont transmises à un serveur dans le but de permettre un suivi à distance de la consommation. Ces opérations sont l'objet de cette partie.

D.1. CHIFFREMENT ET DÉCHIFFREMENT DES DONNEES

Les données sur la consommation d'eau ayant un caractère confidentiel, elles font l'objet d'un chiffrement avant d'être transmises au serveur distant. L'algorithme utilisé est le chiffrement XOR, ou "OU exclusif".

On rappelle que le OU exclusif, noté $\oplus$ et représenté par l'instruction `^` en Python, est un opérateur logique binaire qui renvoie 1 (ou Vrai) si les deux opérandes sont différents, 0 (ou Faux) s'ils sont identiques. Ainsi, `0^0` et `1^1` renvoient tous deux `0`, tandis que `1^0` et `0^1` renvoient tous deux `1`.

L'opérateur `^` permet d'appliquer un OU exclusif *bit à bit* entre deux entiers : le OU exclusif est alors appliqué sur les écritures binaires respectives des deux entiers, entre chaque paire de bits de même rang. Considérons par exemple l'instruction `6^3`. Si l'on écrit les bits de poids fort en premier (le résultat ne dépend pas de la convention choisie) :

- `6` a pour écriture binaire $(110)_2$;
- `3` a pour écriture binaire $(011)_2$;
- le OU exclusif bit à bit donne alors `1^0` (poids fort) puis `1^1` puis `0^1` (poids faible), soit $(101)_2$, soit `5`.

35. Déterminer le résultat renvoyé par l'instruction `13^7`.

On considère dans cette partie que les données à transmettre consistent en une liste d'entiers, et on se donne un autre entier nommé *clé*. Le cryptage XOR consiste à appliquer un OU exclusif bit à bit entre les données à transmettre et la clé.

La façon la plus simple de procéder consiste à appliquer un OU exclusif bit à bit entre chacun des entiers à transmettre et la clé.

36. Écrire une fonction `code1(L,cle)` prenant pour arguments une liste `L` d'entiers et un entier `cle`, et renvoyant la liste des éléments de `L` chiffrés à l'aide de `cle`.

37. Justifier que, si a et b sont deux bits valant 0 ou 1, $(a \oplus b) \oplus b = a$ (on utilisera par exemple une table de vérité). Indiquer alors comment décoder une liste codée par l'instruction `Lc = code1(L,cle)` et préciser pourquoi cet algorithme de chiffrement est dit *symétrique*.

En pratique, la fonction `code1` n'est pas un moyen de chiffrement sûr car tous les éléments de la liste sont cryptés de la même façon. S'il existe une ressemblance entre ces éléments (par exemple s'ils sont proches les uns des autres) et si l'on dispose d'informations sur les données, il est

possible d'identifier la clé puis d'en déduire la totalité du message. Pour y remédier, on se propose d'appliquer le chiffrement non pas à chaque élément de la liste (ce que l'on appelle le chiffrement *par blocs*) mais à la totalité des données mises bout à bout, en répétant pour cela la clé autant de fois que nécessaire (ce que l'on appelle le chiffrement *par flux*).

Considérons par exemple la suite d'entiers positifs (12,3,5,8) codés sur 4 bits, avec la clé 2 codée sur 3 bits. Écrivons ces entiers en binaire les uns à la suite des autres, puis la clé (010) répétée tout au long du message (la dernière répétition est ici partielle), et enfin le résultat du OU exclusif bit à bit :

```
1100 0011 0101 1000
0100 1001 0010 0100
1000 1010 0111 1100
```

Enfin, on interprète le résultat binaire final comme une suite d'entiers codés sur 4 bits (comme la liste de départ). On obtient donc (8,10,7,12).

Dans l'exemple ci-dessus, la clé étant écrite sur 3 bits et les données sur 4 bits, chaque entier est codé de façon différente. Ce principe permet d'obtenir un chiffrage relativement sûr si les nombres de bits utilisés sont premiers entre eux et suffisamment élevés.

On souhaite programmer cet algorithme pour transmettre des entiers positifs codés sur 64 bits. Les représentations binaires seront codées sous la forme de chaînes de caractères contenant des `0` et des `1`. On rappelle que :

— la fonction `bin` permet de convertir un entier en binaire : `bin(10)` renvoie `'0b1010'`, le préfixe `'0b'` indiquant une écriture binaire. On remarquera que l'écriture utilise le nombre minimal possible de bits, de sorte que le bit de poids fort est toujours 1 ;

— la fonction `int` permet de convertir une représentation binaire en entier, si l'on indique la base de numération (ici 2) en second argument : `int('0b1010',2)` et `int('1010',2)` renvoient tous deux `10`.

On suppose que tous les entiers dont il est question dans cette partie sont positifs et représentables sur 64 bits.

38. Écrire une fonction `bin64(n)` prenant pour argument un entier positif `n`, et retournant une chaîne de caractères contenant son écriture binaire sur 64 bits (c'est-à-dire complétée par autant de zéros que nécessaire), sans le préfixe `'0b'`. On utilisera la fonction `bin`.

39. Écrire une fonction `bits(L)` prenant pour argument une liste `L` d'entiers positifs, et retournant une chaîne de caractères constituée des écritures binaires des éléments de `L` sur 64 bits, concaténées les unes aux autres. On utilisera la fonction `bin64`.

40. Écrire une fonction `entiers(flux)` prenant pour argument une chaîne `flux` au format précédent (constituée des caractères `'0'` et `'1'` et de longueur multiple de 64), et renvoyant la liste des entiers dont les codages respectifs sur 64 bits se trouvent bout à bout dans la chaîne `flux`. On utilisera la fonction `int`.

Tournez la page S.V.P.

41. Si l'on appelle l la longueur de la clé, quelle est la position du bit de la clé (entre 0 et $l-1$) correspondant au bit de rang i (compté à partir de zéro) de la liste à chiffrer ? On donnera la réponse sous la forme d'une opération arithmétique simple.

Pour que l'algorithme précédent réalise un chiffrement sûr, on souhaite que le nombre de bits sur lequel est écrit la clé soit premier avec 64, c'est-à-dire impair.

42. Écrire une fonction `code2(L,cle)` prenant pour arguments une liste `L` d'entiers et un entier `cle`, et renvoyant une liste contenant le résultat du chiffrement par flux de `L` à l'aide de `cle` selon l'algorithme décrit ci-dessus. L'entier `cle` sera écrit en binaire sur un nombre de bits impair (on ajoutera donc un 0 en tête de son écriture binaire si cela est nécessaire). On utilisera entre autres les fonctions `bits` et `entiers`.

Une limitation importante du chiffrement par OU exclusif est que si l'on dispose d'un message *et* de sa version chiffrée, alors il est très facile de retrouver la clé de chiffrement. Pour cette raison, sur les applications présentant un risque particulier (ce qui n'est pas le cas ici), la clé de chiffrement est à *usage unique* (l'algorithme est alors dit à *masque jetable*).

43. À partir des résultats précédents et en vous aidant notamment de la réponse à la question 37, expliquer simplement comment il est possible de reconstruire la clé si l'on dispose d'un message et de son chiffrement par la fonction `code1`. Faire de même pour la fonction `code2`.

D.2. CONSULTATION DES DONNÉES

Un particulier souhaite suivre précisément sa consommation d'eau et enregistre à cet effet chaque jour la quantité consommée. Les données ainsi recueillies sont enregistrées dans une base de données. On donne ci-dessous un extrait de la table `eau` utilisée.

Table `eau` (extrait)

`id`	`date`	`conso`
...	...	...
36	2017-11-11	497
37	2017-11-12	156
38	2017-11-13	392
39	2017-11-14	274
40	2017-11-15	343
41	2017-11-16	296
42	2017-11-17	153
...	...	...

Cette table contient :

— un identifiant associé à chaque mesure, de type entier,

— la date, au format `aaaa-mm-jj`, de type chaine de caractères,

— la consommation (en litres), de type flottant.

44. Écrire en SQL les requêtes permettant d'obtenir :

1. la consommation le 17 novembre 2017,
2. les dates des consommations supérieures à 500 litres.

45. L'opérateur `LIKE` permet de faire des comparaisons partielles entre chaînes de caractères, et en particulier de remplacer une partie d'une chaîne par un `%`, la comparaison étant vraie quel que soit la partie de chaîne comparée avec le `%` (par exemple `'informatique' LIKE 'info%'` est vrai). Écrire les requêtes permettant d'obtenir :

1. les consommations de tous les jours du mois de novembre 2017,
2. les consommations totale et moyenne du mois de novembre 2017.

46. On envisage maintenant de gérer les consommations de tout un ensemble d'habitations (par exemple, la mairie enregistre les informations de chaque logement de la commune). Expliquer comment organiser les données en utilisant, en plus de la table `eau`, une table `habitations` dont on précisera les attributs les plus importants. Quel attribut faudra-t-il ajouter à la table `eau` ? Expliquer soigneusement quelle sera la condition de jointure utilisée pour faire des requêtes sur ces deux tables.

Fin de l'épreuve