Banque PT – Informatique et Modélisation 2015

PT - Durée : 4 heures

SUJET

Epreuve d'Informatique et Modélisation

Durée 4 h

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

L'usage de calculatrices est interdit.

AVERTISSEMENT

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

L'épreuve comporte deux parties qui peuvent être traitées indépendamment l'une de l'autre. Une première partie modélisation (durée conseillée 1H30) et une seconde informatique (durée conseillée 2 H30)

Remarques préliminaires importantes : il est rappelé aux candidat(e)s que

Les explications des phénomènes étudiés interviennent dans la notation au même titre que les développements analytiques et les applications numériques ; les résultats exprimés sans unité ne seront pas comptabilisés.

Tout au long de l'énoncé, les paragraphes en italiques ont pour objet d'aider à la compréhension du problème.

Tout résultat fourni dans l'énoncé peut être admis et utilisé par la suite, même s'il n'a pas été démontré par le (la) candidat(e).

Les applications numériques, effectuées sans calculatrice, pourront supporter des arrondis ou simplifications judicieux.

Ce problème traite de systèmes d'identification par radio fréquence (RFID). Il est constitué de deux parties totalement indépendantes : ***l'étude d'un système de type navigo*** *pour la* $1^{ère}$ *partie. Aucune connaissance particulière sur les antennes n'est demandée.*

En 1948, Harry Stockman décrivit dans un article un moyen de communiquer grâce à la réflexion des ondes, annonçant ainsi les futurs systèmes autonomes communiquant par ondes électromagnétiques. Leur but est, par exemple, d'identifier des objets à distance, d'où le nom d'identification radio-fréquence (RFID : Radio Frequency Identification). Quelques années plus tard, les premiers systèmes d'identification utilisant des ondes réfléchies étaient développés.
Les progrès réalisés dans le domaine de l'électronique ont permis son développement et son intégration dans de nombreux domaines. Aujourd'hui, la RFID (du fait de son prix et de la taille des étiquettes) prend une place de plus en plus importante dans la vie courante, et d'un simple fonctionnement en mode tout-ou-rien, au stockage et au traitement d'informations les applications couvrent des domaines allant de la télédétection (identification d'animaux, antivols, localisation...) aux transactions de la vie courante : par exemple dans les systèmes de contrôle d'accès aux transports en commun, type passe Navigo de la RATP pour le métro parisien. Les puces RFID tentent aujourd'hui de supplanter les codes à barres en jouant de leurs avantages, à savoir qu'il est possible d'écrire, d'effacer et de réécrire les données stockées dans une puce un grand nombre de fois, que leur portée peut être supérieure aux lecteurs optiques utilisés pour les codes à barres, et que la communication peut se faire à travers certains obstacles contrairement aux systèmes à lecture optique.

Un système RFID passif est composé de deux entités qui communiquent entre elles (Fig.1) :

- *Un TAG passif (dénommé TAG par la suite) ou étiquette intelligente (aussi appelé transpondeur), associé à l'élément à identifier. ll est capable de répondre à une demande venant d'un lecteur. Le TAG n'a pas d'alimentation de type batterie ou pile mais est autoalimenté par l'onde électromagnétique reçue.*
- *Une station de base ou lecteur RFID qui a pour mission d'identifier le TAG. Le lecteur envoie une onde électromagnétique en direction de l'élément à identifier, cette onde alimente le TAG qui peut alors communiquer avec le lecteur grâce à sa puce électronique interne. En retour, le lecteur reçoit l'information renvoyée par le TAG.*

La figure 1, page suivante, présente le fonctionnement général d'un système RFID. Le lecteur relié à une antenne émettrice agit en maître par rapport au TAG : si le TAG est dans la zone de lecture du lecteur, ce dernier l'active en lui envoyant une onde électromagnétique et entame la communication. Le TAG est quant à lui, constitué d'une antenne et d'une puce électronique qui module l'onde réémise vers le lecteur. En démodulant le signal reçu, le lecteur relié à une application interne récupère l'information pour la traiter, il est chargé de l'interface et de la gestion de l'identification des TAGs qui se présentent à lui.
ll existe plusieurs familles de systèmes RFID dont le principal critère de différentiation est la fréquence de fonctionnement. Les système RFID utilisent des bandes de fréquence à 125 kHz (bande BF), 13,56 MHz (bande HF), 860-960 MHz (bande UHF) et 2,45 GHz. On précise que

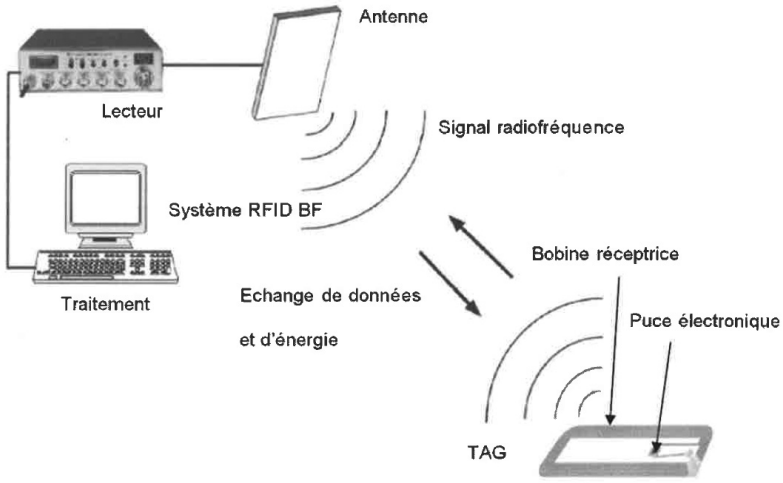


Figure 1

la puissance rayonnée par l'antenne d'émission est une fonction croissante de la fréquence est varie en $\frac{1}{\lambda^4}$.

Bande	*Fréquence*	*Portée*	*Pouvoir de Pénétration dans un conducteur*
BF	*125 kHz*	+	++++
HF	*13.56 MHz*	++	+++
UHF	*860-960 MHz*	+++	++
UHF	*2,45 GHz*	++	+

+ : faible , ++ : Assez bon , + + + : Bon, + + ++ : Excellent

En fonction des différentes fréquences, les principes physiques mis en œuvre ne sont pas les mêmes et le problème aborde certains aspects de la communication.

Première partie : Système RFID à 13.56 MHz en couplage magnétique

La première partie n'est pas reprise ici. Elle ne concerne pas le programme d'informatique.

Deuxième partie : Informatique. Modélisation d'un système de titres de transport sans contact

La première partie a permis de montrer comment la puce RFID peut être alimentée à distance par le lecteur, et comment un signal peut être transmis de l'une à l'autre. L'objectif des parties suivantes est de mettre en place quelques algorithmes visant à :

- simuler la récupération d'un message binaire à partir de la tension captée par le lecteur (Partie 1) ;
- contrôler l'intégrité du message récupéré et corriger les erreurs éventuelles (Partie 2),
- déterminer si un voyageur est autorisé ou non à franchir un point de contrôle (Partie 3),
- traiter les informations recueillies par le système afin d'améliorer le service (Partie 4).

Ces quatre parties sont indépendantes.

Les algorithmes demandés seront réalisés, au choix, dans le langage Python ou le langage Scilab. **Le candidat doit précsier en tête de partie le langage choisi et s'y tenir.**On supposera que tout module nécessaire à l'utilisation des fonctions usuelles (pi, sin...) a été importé.

Simulation numérique de la démodulation d'amplitude par le lecteur

La première partie a montré que la puce et le lecteur peuvent tous deux recevoir des messages contenus dans des tensions alternatives d'amplitudes variables. Nous allons maintenant modéliser une telle tension puis simuler numériquement le comportement d'un dispositif permettant d'en extraire une information binaire.

La tension $e(t)$ reçue par le lecteur peut être modélisée par une sinusoïde de fréquence $f = 13{,}56$ MHz dont l'amplitude E_0 est modulée par le signal binaire $b(t)$ transmis par la puce :

$$e(t) = E_0(t)\sin(2\pi f t) \text{ avec } \begin{cases} E_0(t) = E_{\min} = 1{,}3 \text{ V lorsque } b(t) = 0 \\ E_0(t) = E_{\max} = 1{,}5 \text{ V lorsque } b(t) = 1 \end{cases}$$

La puce transmet un nouveau bit toutes les 16 périodes de la porteuse. A titre d'illustration, la Figure 1 (ci dessous) représente la tension $e(t)$ correspondant à la séquence de bits (0, 1, 0). La tension est en volts et le temps en secondes.

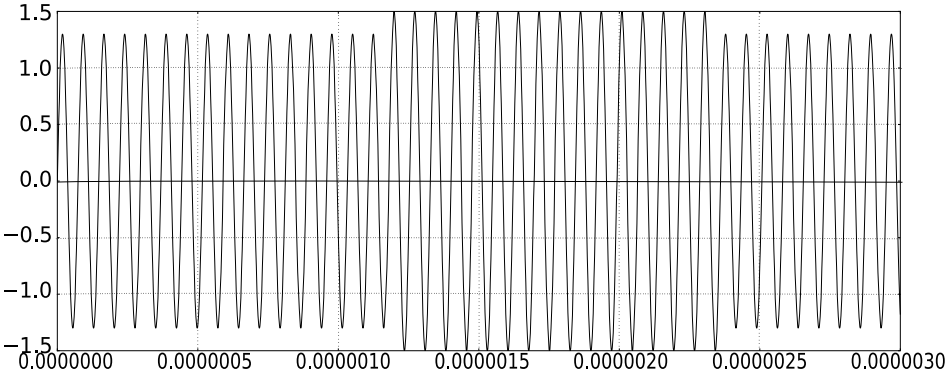


FIGURE 1 – tension d'entrée du démodulateur (sinusoïde modulée en amplitude par un signal binaire)

La simulation porte sur un intervalle de temps $[0, T_{\max}]$. Le temps sera représenté numériquement par une liste de N instants régulièrement espacés que l'on écrira, compte tenu des conventions de numérotation propres à chaque langage, $T = [t_0 = 0,...,t_{N-1} = T_{\max}]$ sous Python ou $T = [t_1 = 0,...t_N = T_{\max}]$ sous Scilab.

Q 1. Écrire une fonction `init_T(Tmax,dt)` prenant pour arguments la durée `Tmax` de la simulation et le pas de temps `dt` et retournant la liste `T`. Si `Tmax` n'est pas multiple de `dt`, on arrondira la durée de la simulation au multiple entier de dt immédiatement supérieur.

La tension d'entrée $e(t)$ sera de même représentée par une liste `E` telle que `E[i]`$=e(t_i)$.

Q 2. Écrire une fonction `init_E(T,f)` prenant pour arguments la liste `T` des instants de la simulation et la fréquence `f` de la porteuse et retournant la liste des valeurs $e(t_i)$ de la tension e aux instants `T[i]`, d'après la définition de $e(t)$ et pour le message binaire (0,1,0) (on considèrera que la simulation ne se poursuit pas au-delà).

Pour récupérer l'information binaire contenue dans une telle tension, il faut en extraire l'amplitude. On utilise pour cela le dispositif de la Figure 2, appelé **détecteur d'enveloppe**.

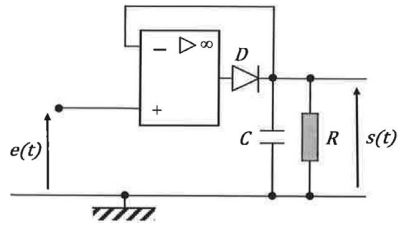


FIGURE 2 – Détecteur d'enveloppe

La modélisation du comportement de ce montage permet d'exprimer la tension de sortie $s(t)$ en fonction de la tension d'entrée $e(t)$ de la façon suivante (τ est la constante de temps du circuit RC) :

1. Si la diode est *bloquée*, alors $\dfrac{ds}{dt}(t) + \dfrac{1}{\tau}s(t) = 0$ et il faut vérifier que $s(t) > e(t)$
2. Si la diode est *passante*, alors $s(t) = e(t)$ et il faut vérifier que $\dfrac{ds}{dt}(t) + \dfrac{1}{\tau}s(t) > 0$

Autrement dit, la diode peut prendre deux états (bloquée ou passante), chacun soumis à une condition (inégalité) et muni d'une équation d'évolution (égalité) ; lorsque la condition cesse d'être vérifiée, la diode change d'état. Nous allons utiliser ces équations pour simuler numériquement l'évolution de la tension $s(t)$. Pour cela, on propose d'utiliser une variable (par exemple booléenne) indiquant l'état de la diode et, à chaque pas de temps t_i :

- de calculer $s(t_{i+1})$ en utilisant l'équation correspondant à la valeur de la variable à l'instant t_i,
- *puis* de tester la condition correspondante à partir de la valeur de $s(t_{i+1})$ et, si elle n'est plus vérifiée de mettre à jour la variable.

Le résultat est stocké dans une liste `S` avec `S[i]`$= s(t_i)$.

Q 3. Donner une approximation de $\dfrac{ds}{dt}(t_i)$ en fonction de $s(t_i)$, $s(t_{i+1})$ et $\Delta t = t_{i+1} - t_i$ en utilisant la formule d'Euler explicite. En déduire, dans le cas où la diode est *bloquée* à l'instant t_i, la relation de récurrence donnant $s(t_{i+1})$ en fonction de $s(t_i)$, τ et Δt.

Dans le cas où la diode est *passante* à l'instant t_i, on ne peut pas utiliser cette formule pour tester l'état de la diode à t_{i+1}. On utilise donc l'approximation *d'Euler arrière* ou *Euler implicite*, qui consiste à utiliser la même formule mais pour approcher $\dfrac{ds}{dt}(t_{i+1})$ au lieu de $\dfrac{ds}{dt}(t_i)$.

Q 4. Pourquoi ne peut-on pas utiliser la formule d'Euler explicite pour effectuer le test ici ? Donner, en utilisant la démarche proposée, une condition portant sur $s(t_{i+1})$, $s(t_i)$, τ et Δt permettant de déterminer si la diode se bloque ou non à l'instant t_{i+1}.

Q 5. Écrire alors une fonction `solve(T,E,tau)` prenant pour arguments la liste `T` des instants de la simulation, la liste `E` des tensions d'entrée et la constante de temps `tau` et retournant la liste `S` des tensions de sortie. Les conditions initiales seront prises nulles et, si nécessaire, l'état initial de la diode sera supposé passant.

Les Figures 3, 4 et 5 donnent les résultats (entrées et sorties numériques) obtenus pour trois pas de temps différents choisis parmi 1 ns, 10 ns et 100 ns. Sur ces trois graphes, la tension d'entrée *avant discrétisation* est celle de la Figure 1 et la constante de temps du circuit est $\tau = 1\mu$s. Seul le pas de temps change d'une simulation à l'autre.

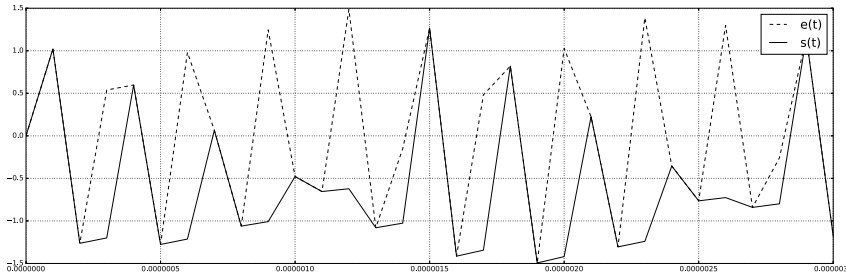


FIGURE 3 – résultats de la simulation pour le pas de temps Δt_1

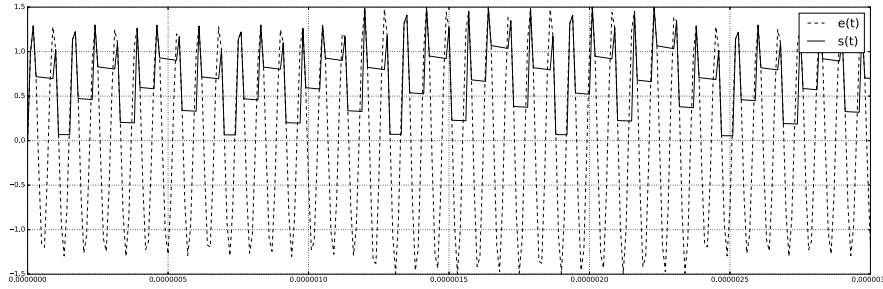
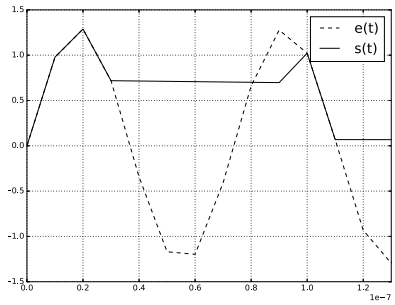


FIGURE 4 – résultats de la simulation pour le pas de temps Δt_2 (avec zoom à l'origine disponible)

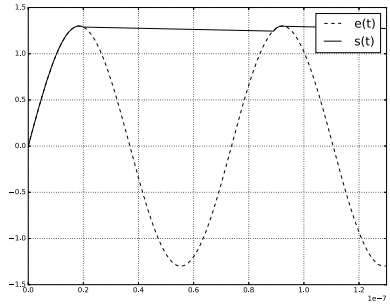
Q 6. Indiquer la valeur du pas de temps (1, 10 ou 100 ns) correspondant à chacune de ces trois simulations, en justifiant vos réponses.

Q 7. Expliquer en quelques phrases les causes des différences obtenues entre les trois résultats $s(t)$. Repérer en particulier les instants auxquels la diode change d'état. Que constatez-vous ?

Pour distinguer l'état "haut" de l'état "bas" de la tension $s(t)$ et ainsi extraire les 0 et les 1 du message binaire transmis par modulation, il est nécessaire de déterminer un seuil séparant les deux niveaux.



(a) zoom de la figure 4 à l'origine



(b) zoom de la figure 5 à l'origine

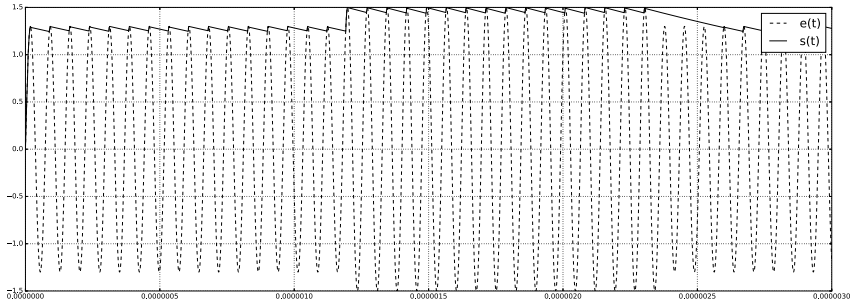


FIGURE 5 – résultats de la simulation pour le pas de temps Δt_3 (avec zoom à l'origine disponible)

Q 8. Indiquer, pour chacun des trois résultats, s'il est possible d'identifier un tel seuil, et donc si la récupération du message binaire semble réalisable *si l'on se base uniquement sur ce résultat*. Conclure sur le critère que doit respecter le pas de temps d'une simulation temporelle pour que les résultats de celle-ci aient une chance d'être pertinents.

Vérification de l'intégrité des données et correction des erreurs

Le signal transmis par une liaison RFID 13,56 MHz peut être perturbé par toutes sortes de facteurs pouvant provoquer des erreurs dans les données : autres signaux électromagnétiques, masses métalliques, imperfections du matériel électronique... En pratique, il est donc indispensable de pouvoir détecter ces erreurs et, dans la mesure du possible, les corriger sans que cela ne nécessite une nouvelle transmission ; l'objet de cette partie est de mettre en place quelques algorithmes dans ce but.

2.1 Bit de parité

Une technique simple et très répandue pour s'assurer qu'une donnée binaire sera lue correctement par son récepteur est de lui adjoindre un **bit de parité**, égal par définition à :

- 0 si la donnée contient un nombre pair de 1 (et, donc, si ses bits sont de somme paire),
- 1 si la donnée contient un nombre impair de 1 (et, donc, si ses bits sont de somme impaire).

Après réception de la donnée, le récepteur recalcule le bit de parité et le compare à celui que l'émetteur lui a adressé. Si la donnée n'a pas été altérée lors de la transmission, alors les deux bits de parité sont forcément identiques.

Q 9. Donner les bits de parité associés aux représentations binaires des entiers 5, 16 et 37.

Q 10. Écrire une fonction `parite(bits)` prenant pour argument une liste `bits` constituée d'entiers valant 0 ou 1 et retournant l'entier 0 ou 1 correspondant à son bit de parité.

Les techniques de vérification les plus simples consistent à découper la donnée en blocs et à joindre un bit de parité à chaque bloc. Par exemple, certains protocoles transmettent sept bits de données pour un bit de parité.

Q 11. Donner un exemple d'erreur n'étant pas détectable par cette technique. Si une erreur a été détectée, est-il possible de la corriger sans retransmettre la donnée ?

2.2 Code de Hamming

Le code de Hamming est un exemple d'utilisation des bits de parité pour détecter et corriger des erreurs. Nous nous intéressons ici au code dit (7,4), ainsi appelé car il consiste à joindre trois bits de parité à quatre bits de données, ce qui donne un message d'une longueur totale de sept bits. Ces trois bits de parité sont définis ainsi : si la donnée s'écrit (d_1,d_2,d_3,d_4) avec $d_i = 0$ ou 1, alors :

- p_1 est le bit de parité du triplet (d_1,d_2,d_4),
- p_2 est le bit de parité du triplet (d_1,d_3,d_4),
- p_3 est le bit de parité du triplet (d_2,d_3,d_4).

Le message encodé, que l'on transmet, s'écrit alors comme suit : (p_1,p_2,d_1,p_3,d_2,d_3,d_4).

Q 12. Écrire une fonction `encode_hamming(donnee)` prenant pour argument une liste `donnee` de quatre bits (représentés par des entiers valant 0 ou 1) et retournant une liste de bits contenant le message encodé. On pourra appeler la fonction `parite(bits)` précédemment définie.

Le contrôle après réception d'un message ainsi encodé est relativement simple. On pourrait naturellement recalculer les trois bits de parité de la donnée et les comparer aux valeurs transmises, mais la technique proposée par Hamming est de calculer les trois bits de contrôle suivants, notés (c_1, c_2, c_3), à partir du *message complet* (données et bits supplémentaires), noté $(m_1,..., m_7)$:

- c_1 est le bit de parité de l'ensemble (m_4, m_5, m_6, m_7),
- c_2 est le bit de parité de l'ensemble (m_2, m_3, m_6, m_7),
- c_3 est le bit de parité de l'ensemble (m_1, m_3, m_5, m_7).

On montre que si le message a bien été encodé selon les règles précédentes et n'a pas été altéré, alors les trois bits de contrôle doivent être à 0. Si ce n'est pas le cas, alors il y a eu une erreur ; l'intérêt de la technique de Hamming est que *dans le cas particulier où l'erreur est unique, le mot de contrôle donne la représentation binaire de la position de cette erreur en numérotant à partir de 1*. Par exemple, si (c_1, c_2, c_3) = (0,1,1), alors l'erreur porte sur le troisième bit du message. Il suffit ainsi d'inverser ce bit (le mettre à 1 s'il est à 0, et inversement) pour corriger l'erreur.

La donnée décodée est alors constituée des quatre bits (d_1,d_2,d_3,d_4) qui se trouvent respectivement en positions 3, 5, 6 et 7 (toujours en numérotant à partir de 1) conformément à la description de l'encodage donnée ci-dessus.