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 strictement interdit. Les surveillants et surveillantes se réservent le droit de les confisquer.
- 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. La présence d'une information d'identification en dehors du cartouche donnera lieu à un point de pénalité et la page concernée pourra être soustraite de la correction.

" Les applications numériques sont attendues avec un (voire deux) chiffres significatifs. Le jury prendra pleinement en compte le fait que l'épreuve est sans calculatrice et de ce fait sera tout à fait tolérant sur la précision des résultats numériques, l'essentiel étant l'ordre de grandeur ".

Tournez la page S.V.P

Présentation de la problématique

AUTOUR DE LA GÉOLOCALISATION PAR SATELLITES

La *géolocalisation par satellites* ou GNSS (pour *Géolocalisation et Navigation par un Système de Satellites*) est une technologie permettant à tout utilisateur de déterminer sa position n'importe où sur Terre à l'aide d'un récepteur adapté, qui capte et traite des signaux radio émis par des satellites en orbite autour de la Terre. Il existe quatre grands services de GNSS, dont le *Global Positioning System* (GPS) américain mis en service en 1995, et le système *Galileo* européen, partiellement opérationnel depuis 2016.

1. Fonctionnement

Un système de GNSS repose sur trois constituants principaux (figure 1) :

1. une *constellation de satellites*, suffisamment nombreux pour que plusieurs d'entre eux soient en permanence visibles en tout point de la surface de la Terre et qui émettent en permanence des signaux radio contenant des repères temporels et des informations sur leurs localisations ;
2. des *stations de contrôle* au sol, qui suivent les satellites, identifient les erreurs dans les messages qu'ils envoient (sur les positions, vitesses et temps), les corrigent et leur envoient des rectificatifs ;
3. un *récepteur* par utilisateur, qui capte les signaux des satellites et les utilise pour se localiser.

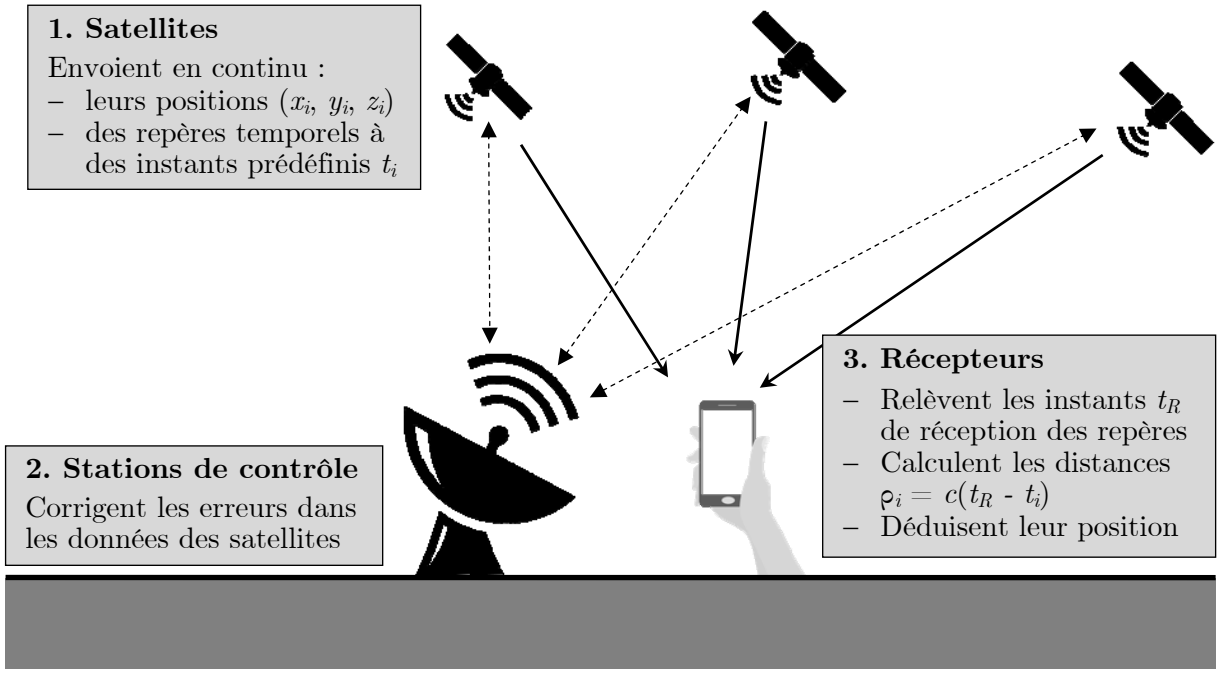

FIGURE 1 – Constituants d'un GNSS et informations transmises

Pour se localiser, le récepteur mesure la distance qui le sépare de plusieurs satellites. Il est pour cela muni d'une horloge qui lui permet, lorsqu'il détecte les repères temporels contenus dans les signaux, de relever leur instant de réception t_R. Ces repères étant émis à des instants précis t_i, il en déduit les distances $\rho_i = c(t_R - t_i)$, où c est la vitesse de propagation des ondes électromagnétiques (figure 1). En théorie, le récepteur a besoin d'au moins trois distances pour se localiser. En pratique, compte tenu des imprécisions qui affectent les mesures du temps, il en utilise au moins quatre. Le calcul géométrique de sa position à partir des distances (et des positions) des satellites s'appelle *trilatération* et est illustré figure 2.

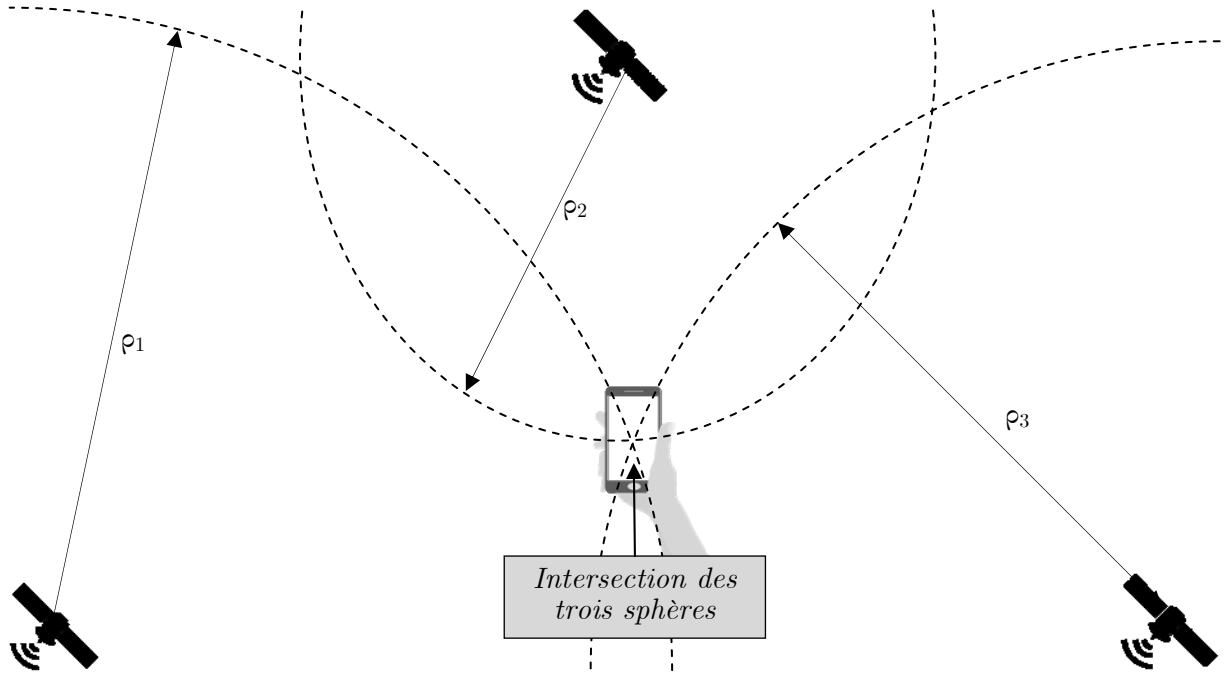

FIGURE 2 – Exemple de trilatération dans le plan

Les avantages de cette technologie sont nombreux : le positionnement est précis (quelques mètres pour la version basique du GPS américain), le service est disponible partout sur Terre sans interruption, l'information est mise à jour en moins d'une seconde une fois que le récepteur est initialisé, et le nombre d'utilisateurs est illimité puisque le récepteur est purement passif (il n'émet rien). La GNSS a ainsi connu un succès considérable : de nos jours, il existe plusieurs milliards de récepteurs GPS.

2. Travail demandé

Ce sujet comporte trois parties indépendantes :

1. la *modélisation des satellites* : il s'agit de prévoir leurs mouvements et de modéliser la propagation des signaux en direction de la Terre ;
2. la simulation du *décodage des signaux*, qui présente une partie du traitement du signal effectué par le récepteur pour identifier les satellites émetteurs ainsi que la distance le séparant de chacun d'eux ;
3. la simulation d'une *recherche d'itinéraires* qui est une application courante des GNSS, avec la recherche d'un plus court chemin et la mise en œuvre de statistiques sur des temps de trajet.

Tournez la page S.V.P.

Première partie

MODÉLISATION DES SATELLITES

A. Mouvements et trajectoires

On s'intéresse au mouvement d'un satellite GPS autour de la Terre, dont la trajectoire est circulaire à une altitude $h = 20000$ km. On considère la Terre et le satellite comme des points matériels de masses respectives $M_T = 6.10^{24}$ kg et $m = 700$ kg. Le mouvement est plan et circulaire, et on repère la position M du satellite en coordonnées polaires, le centre de la Terre étant à l'origine O (figure 3 ci-dessous). On donne la valeur de la constante de gravitation : $G = 6,7.10^{-11}$ SI ainsi que le rayon de la Terre : $R_T = 6400$ km.

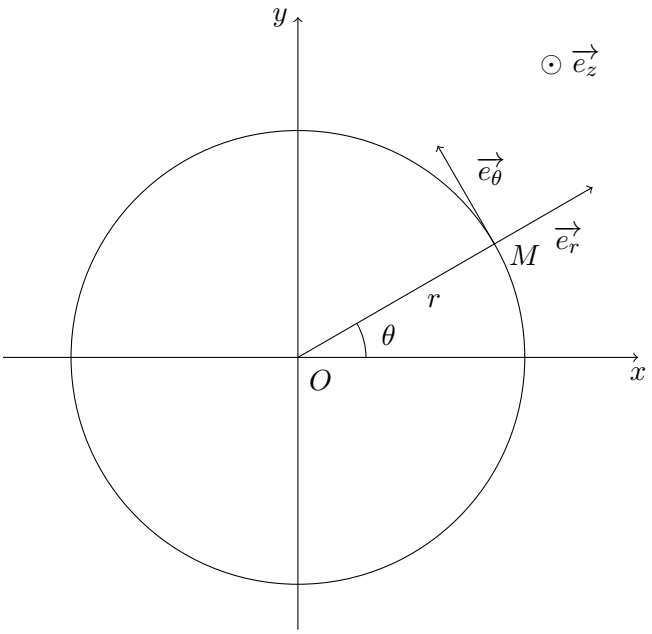

FIGURE 3 – Repérage en coordonnées polaires dans le plan du mouvement

1. Sans démonstration, écrire l'expression de la force vectorielle subie par le satellite et calculer la valeur numérique de sa norme. Toujours sans démonstration, donner l'expression du champ de gravitation $\vec{g}(M)$ créé par la Terre au point M en fonction de m, M_T, h, R_T et G. On pourra utiliser l'aide numérique suivante : $2,6^2 \simeq 6,7$.

2. En considérant la Terre comme sphérique et de masse volumique uniforme, justifier le fait que son champ de gravitation en un point extérieur à la Terre est le même que celui d'un point matériel situé en son centre affecté de toute la masse de la Terre.

3. En utilisant la deuxième loi de Newton, justifier :
- la relation $v^2 = \frac{GM_T}{r}$, où v est la vitesse du satellite ;
- que le mouvement circulaire est nécessairement uniforme.

4. En utilisant la question précédente, retrouver la troisième loi de Kepler dans le cas circulaire.

5. Calculer la période de révolution T_{GPS} des satellites GPS en secondes (rappel : $2,6^2 \simeq 6,7$). Comparer cette valeur avec la période d'un satellite géostationnaire.

On donne ci-dessous la projection sur la surface terrestre de la trajectoire d'un satellite GPS ainsi que quelques valeurs de latitudes et longitudes.

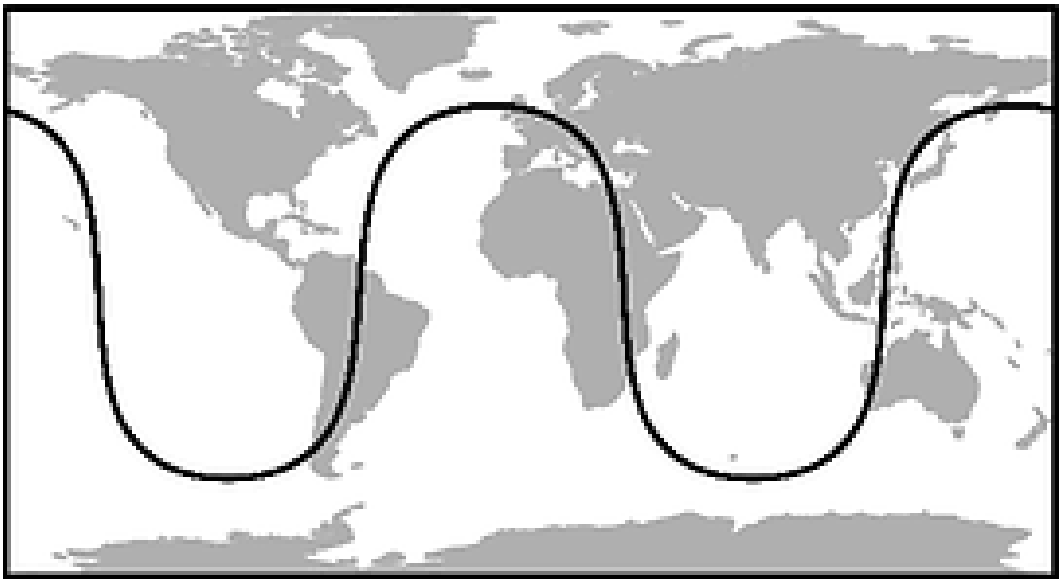

FIGURE 4 – Projection de la trajectoire d'un satellite gps

Lieu	Paris	Londres	Edimbourg	New York	Tokyo	Cap Horn
Latitude	48,5° Nord	51,3° Nord	55,6° Nord	40,4° Nord	35,4° Nord	55,6° Sud
Longitude	2,2° Est	0,1° Ouest	3,1° Ouest	74° Ouest	139,4° Est	67,2° Ouest

FIGURE 5 – Latitudes et longitudes de quelques lieux

6. Evaluer approximativement, à $\pm 5°$ près, l'inclinaison α du plan de l'orbite de ce satellite par rapport au plan équatorial.

7. On admet que le sens de la révolution du satellite est le même que celui de la rotation propre de la Terre. Sur la figure 4, ce sens est-il vers la droite ou vers la gauche (une justification, même succinte, est attendue) ? Quelle est la période du phénomène «le satellite se retrouve à la verticale d'un même point de la Terre» ?

B. Propagation du signal

On s'intéresse dans cette partie à la propagation du signal émis par un des satellites. On considérera qu'il s'agit d'une onde électromagnétique de fréquence environ $f = 1600$ MHz. On donne la vitesse de la lumière dans le vide ainsi que la permittivité diélectrique du vide : $c = 3.10^8$ m.s^{-1} et $\varepsilon_0 = 8,8.10^{-12}$ F.m^{-1}.

Tournez la page S.V.P.

8. En partant des équations de Maxwell, établir l'équation d'onde vérifiée par le champ électrique dans le vide, sans charges ni courants. Quelle est la relation entre μ_0, ε_0 et c ?

On considère que le satellite est à la verticale du récepteur. Localement, à l'échelle du récepteur, on peut modéliser l'onde qui arrive du satellite par une onde plane monochromatique de la forme :

$$\vec{E} = E_0 \cos(\omega t + kz)\vec{e_x}$$

9. Faire un schéma où figurent le satellite S, le récepteur R ainsi que la base directe $(\vec{e_x}, \vec{e_y}, \vec{e_z})$. Quels sont les direction, sens de propagation et la polarisation relativement à ce système d'axes ?

10. Vérifier que cette onde est bien solution de l'équation de propagation à une condition près à préciser reliant k, ω et c. Calculer numériquement k et la longueur d'onde λ associés.

11. Exprimer le champ $\vec{B}$ associé à la propagation de l'onde.

12. Exprimer le vecteur de Poynting $\vec{\Pi}$ et en déduire l'expression de la puissance moyenne I par unité de surface associée à la propagation de l'onde.

13. Au niveau du récepteur, l'amplitude du champ électrique est $E_0 = 2,5.10^{-6}$ V.m^{-1}. Calculer numériquement la puissance moyenne par unité de surface à cet endroit. On pourra utiliser l'aide numérique suivante : $2,5^2 \simeq 6,3$.

14. On considère que le satellite émet de manière isotrope. Calculer la puissance P émise par le satellite dans cette hypothèse (rappel : $2,6^2 \simeq 6,7$). Peut-on supposer que la puissance réelle est supérieure ou inférieure à celle-ci ?

C. Décalage dû à la traversée de la troposphère

Les informations reçues par le récepteur provenant d'un des différents satellites permettent de déterminer la distance d entre le satellite émetteur et le récepteur via le temps de propagation Δt. Si on considère que la vitesse de propagation du signal est c (vitesse de la lumière dans le vide), alors $d = c\Delta t$. Cependant, le signal traverse l'atmosphère et la vitesse de propagation dans l'air n'est pas exactement égale à $c = 3.10^8$ m.s^{-1}, il y a donc une correction à appliquer. On ne considère dans la suite que les 50 premiers kilomètres d'atmosphère (en partant du sol) que l'on appelle *troposphère*. On notera L cette longueur. Pour simplifier, on se place dans le cas où le satellite émetteur est à la verticale du récepteur.

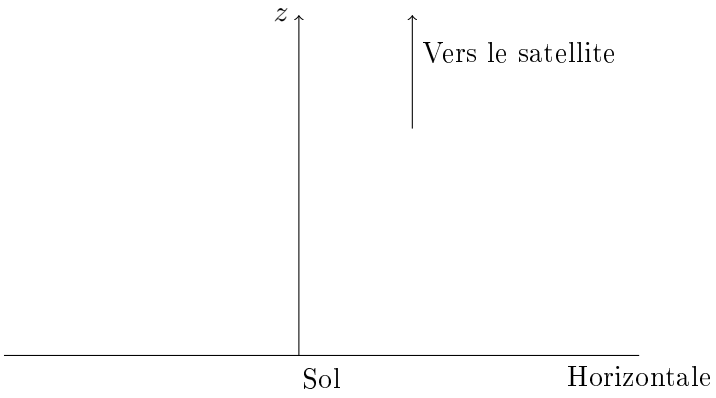

FIGURE 6 – Repérage de l'altitude

15. En considérant l'air comme un gaz parfait, donner l'expression de sa masse volumique ρ en fonction de M_{air} (masse molaire de l'air), P (pression), R (constante des gaz parfaits) et T (température). On donne $M_{\text{air}} = 29$ g.mol^{-1} et $R = 8,3$ SI. Calculer numériquement la masse volumique de l'air au niveau du sol où on prendra $P = P_0 = 1$ bar et $T = T_0 = 270$ K.

16. La loi de Gladstone donne une relation entre l'indice de réfraction n d'un gaz et sa masse volumique ρ, K étant une constante : $n = 1 + K\rho$
Montrer que, pour un gaz parfait, on peut écrire $n = 1 + K'\frac{P}{T}$. On prendra dans la suite $K' = 7,8.10^{-7}$ SI. Quelle est l'unité de K' ?

17. Montrer que l'expression de la pression P en fonction de l'altitude z dans le modèle de l'atmosphère isotherme s'écrit $P = P_0 \exp\left(-\frac{z}{H}\right)$ et préciser l'expression de H.

18. On prendra $P(z = 0) = P_0 = 1$bar, $T = 270$K et $g = 9,8$m.s^{-2}. Calculer numériquement H ainsi que la pression à l'altitude de H. Aide numérique : $e^{-1} \simeq 0,37$

19. On s'intéresse à la variation de l'indice de réfraction n de l'air en fonction de l'altitude z. Montrer que l'on peut écrire, à partir des résultats précédents :

$$n = 1 + \alpha P_0 \exp\left(-\frac{z}{H}\right)$$

Donner l'expression de α et sa valeur numérique.

20. Montrer que le temps dt pour traverser verticalement une petite épaisseur d'atmosphère dz s'écrit :

$$dt = \frac{1}{c}\left(1 + \alpha P_0 \exp\left(-\frac{z}{H}\right)\right) dz$$

21. En déduire l'expression littérale du temps Δt nécessaire pour traverser verticalement les $L = 50$km de la troposphère en fonction de c, L, α, P_0 et H.

22. En déduire le temps de propagation supplémentaire t_{sup} lors de la traversée de la troposphère par rapport à la même distance parcourue dans le vide. Faire l'application numérique. A quelle erreur sur l'estimation de la distance entre le satellite émetteur et le récepteur cela conduit-il ? Aide numérique : $e^{-6,25} \simeq 0,002$

Tournez la page S.V.P.

Deuxième partie

TRAITEMENT DU SIGNAL PAR LE RÉCEPTEUR

Dans cette partie, les fonctions et programmes attendus devront être écrits en langage Python. 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 dans la suite de celui-ci, même sans avoir été codée.

D. Identification du satellite et estimation de la distance

Le signal capté par l'antenne du récepteur, une fois amplifié, filtré et démodulé, donne accès à un *message binaire*, c'est-à-dire une succession de 0 et de 1 logiques. Le message présenté dans cette partie est une version simplifiée des fonctionnalités de base du GPS américain. Il est constitué de deux séquences :

1. le *message de navigation*, transmis à 50 bit/s, qui donne notamment la position des satellites ;
2. le *code C/A*, pour *Coarse/Acquisition code* (« code grossier et d'acquisition » en français), transmis à 1,023 Mbit.s^{-1}, qui permet d'identifier le satellite émetteur ainsi que la durée de propagation depuis celui-ci jusqu'au récepteur.

Ces deux séquences sont combinées à l'aide d'une opération « OU exclusif » (figure 7).

Message de navigation 50 bit.s^{-1}	0							1						
Code C/A 1 023 000 bit.s^{-1}	1	0	0	...		...	1	1	0	0	...		...	1
Message transmis 1 023 000 bit.s^{-1}	**1**	**0**	**0**	**...**		**...**	**1**	**0**	**1**	**1**	**...**		**...**	**0**

FIGURE 7 – Constitution du message émis par les satellites

Le message de navigation étant transmis beaucoup plus lentement que le code C/A, son effet se limite à inverser de temps à autre le code C/A, comme le montre la figure 7. L'objectif de cette partie est de montrer, à l'aide d'un programme informatique simple, comment le code C/A est utilisé pour identifier le satellite émetteur et estimer la durée de propagation.

D.1. Les codes C/A : définitions

Les codes C/A sont des codes binaires qu'il est commode d'écrire selon la convention NRZ (Non-Retour à Zéro), c'est-à-dire en notant les deux niveaux logiques par les nombres -1 et +1. Dans ce qui suit, les codes sont représentés par des suites $a = (a_i)_{i\in\mathbb{N}}$ d'entiers valant 1 ou -1, dont les indices vont de 0 à $n-1$ tous deux inclus, où n est la longueur du code.

Les codes C/A sont périodiques de longueur $n = 1023$ et sont émis à une vitesse de 1,023 Mbit.s^{-1} : ils se répètent donc exactement toutes les millisecondes. Chaque satellite émet un code C/A différent. Leur manipulation par le récepteur GPS repose sur les deux définitions suivantes.

Définition 1 : la *corrélation* de deux codes $a = (a_i)_{i\in\mathbb{N}}$ et $b = (b_i)_{i\in\mathbb{N}}$ de même longueur n est le nombre réel suivant :

$$\langle a, b\rangle = \frac{1}{n}\sum_{i=0}^{n-1} a_i b_i$$

Définition 2 : le code $a = (a_i)_{i\in\mathbb{N}}$ *retardé* d'un entier k est le code dont le i-ème terme est a_{i-k}, étant entendu que les codes sont n-périodiques : les indices sont donc définis modulo n et peuvent être des entiers relatifs quelconques. Par exemple, le code $(-1, 1, 1, 1)$ retardé de 1 est $(1, -1, 1, 1)$.

Nous nous proposons ici d'implanter ces opérations en Python ; les fonctions obtenues seront utilisées dans toute la suite du sujet. Les codes C/A sont représentés par des listes d'entiers valant 1 ou -1, de longueurs *a priori* quelconques (pas forcément 1023 ; nous verrons pourquoi par la suite). Si besoin, on utilisera l'opérateur modulo (`%`) pour gérer la périodicité des indices.

23. Écrire une fonction `corr` qui prend pour arguments deux listes `a` et `b` de *même longueur* (on ne demande pas de le vérifier) représentant des codes C/A, et qui renvoie un flottant contenant leur corrélation. Pour cette question, les fonctions intégrées de type `sum` sont interdites.

24. Écrire une fonction `retarde` qui prend pour arguments une liste `a` représentant un code C/A et un entier `k`, et qui renvoie la liste représentant le code retardé de `k`.

D.2. Trois propriétés des codes C/A

L'exploitation des codes C/A pour identifier les satellites et les durées de propagation repose sur les trois propriétés suivantes, qui font appel aux définitions précédentes :

1. l'*autocorrélation* d'un code, c'est-à-dire sa corrélation avec lui-même, vaut 1 ;
2. la corrélation d'un code avec le même code *retardé* d'un nombre k non nul est quasi-nulle ;
3. la corrélation entre deux codes émis par *deux satellites différents* est quasi-nulle, et cela vaut aussi si l'un des deux codes (ou les deux) est retardé.

Les trois questions suivantes visent à expliquer ces propriétés et ne demandent pas de longs développements mathématiques.

25. Montrer la première des trois propriétés ci-dessus (autocorrélation égale à 1).

26. Montrer plus généralement que la corrélation de deux codes est toujours comprise entre -1 et 1. À quelle condition vaut-elle -1 ? On pourra donner un exemple.

27. Combien y a-t-il de codes C/A possibles ? Donner un ordre de grandeur de ce nombre sous la forme 10^k en considérant que $2^{10} \approx 10^3$.

Tournez la page S.V.P.

En pratique, on n'utilise qu'une trentaine de codes, qui sont choisis de sorte à vérifier les propriétés 2 et 3 ci-dessus. Ces codes sont générés automatiquement à l'aide d'un algorithme simple mais calculatoire, qui n'est pas détaillé ici, à partir de deux entiers naturels notés p et q ; la documentation du GPS donne les valeurs de p et q correspondant à chaque satellite (par exemple, pour le satellite numéroté 1, p vaut 1 et q vaut 5).

Dans la suite du sujet, on suppose que l'on dispose d'une fonction Python `codeCA` qui prend pour arguments deux entiers `p` et `q` et renvoie une liste de longueur 1023, dont chaque élément vaut -1 ou 1, contenant le code C/A correspondant. On se propose d'utiliser cette fonction pour illustrer les trois propriétés ci-dessus.

28. En utilisant les fonctions `codeCA`, `retarde` et `corr` (les deux dernières étant définies dans la partie précédente), écrire un bloc d'instructions qui :

- construit une liste `code1` contenant le code C/A correspondant à `p=1` et `q=5` ;
- construit une liste `code2` contenant le code C/A correspondant à `p=2` et `q=6` ;
- construit une liste de listes nommée `codes1` telle que pour tout `i` allant de 0 à 1022, `codes1[i]` corresponde à `code1` retardé de `i` ;
- construit une liste nommée `correls11` telle que `correls11[i]` soit la corrélation de `codes1[i]` et de `code1` ;
- construit une liste nommée `correls12` telle que `correls11[i]` soit la corrélation de `codes1[i]` et de `code2`.

On pourra, sans obligation, utiliser des constructions de listes par compréhension.

Le tracé des listes `correls11` et `correls12` en fonction de l'indice est représenté figure 8.

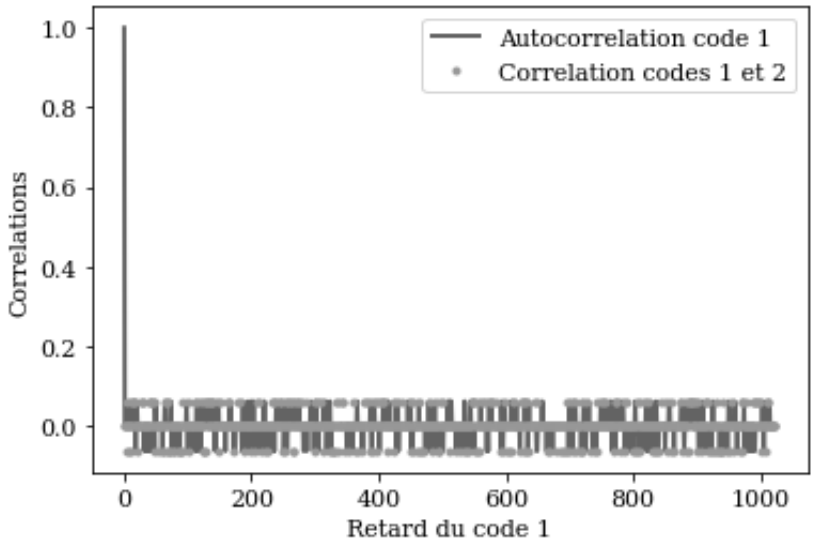

FIGURE 8 – Corrélation du code 1 retardé avec les codes 1 et 2

Ce graphe illustre bien les trois propriétés ci-dessus : seule l'*autocorrélation non retardée* de `code1` vaut 1. Si on corrèle `code1` avec une version retardée de lui-même ou avec `code2` (avec ou sans retard), la valeur obtenue est très faible.

D.3. L'acquisition

Les propriétés précédentes permettent, à partir du message binaire capté par le récepteur GPS, d'identifier le satellite qui l'a émis et d'estimer (grossièrement) la durée de la propagation. Cette opération s'appelle *acquisition*.

Pour la réaliser, le récepteur GPS est muni d'un générateur de codes C/A identique à celui embarqué dans les satellites. Il génère les codes correspondant à tous les satellites possibles (il y en a au plus 31) avec tous les retards possibles (il y en a 1023) et, pour chacun d'eux, calcule sa corrélation avec le message reçu. Les résultats précédents montrent que si les codes ne correspondent pas ou présentent un décalage, alors leur corrélation sera proche de zéro. C'est uniquement si les codes sont identiques *et* alignés que l'on obtiendra une valeur proche de 1 (ou de -1 si le code a été inversé par le message de navigation, comme le montre la figure 7). Le « bon » satellite et le « bon » retard sont donc ceux pour lesquels la corrélation est *maximale en valeur absolue*.

Pour la question suivante, on suppose que les résultats des corrélations sont stockés dans une liste de listes `T`, de sorte que `T[i][j]` est la corrélation du signal reçu avec le code C/A du `i`-ème satellite retardé de `j` bits. On précise que si `x` est un flottant ou un entier, `abs(x)` renvoie sa valeur absolue.

29. Écrire une fonction `indmax` prenant pour argument une liste de listes `T` et renvoyant le couple d'indices `(i,j)` tel que `abs(T[i][j])` soit maximal. On admettra que toutes les listes `T[i]` ont la même longueur et on contrôlera, à l'aide de l'instruction `assert`, que `T[0]` n'est pas vide.

On obtient ainsi directement le numéro du satellite ayant émis le message capté. De plus, le début de chaque code C/A est émis *exactement* un nombre entier de millisecondes après une heure « ronde » (les satellites possèdent à cet effet des horloges atomiques très précises). Si l'horloge du récepteur a été bien recalée lors des mesures précédentes, le nombre de valeurs dont il a fallu retarder le code donne donc une information sur la durée de la transmission.

Ce procédé rencontre néanmoins une limite physique importante. On rappelle que le code C/A est émis à 1,023 Mbit.s^{-1} et que la vitesse de la lumière dans le vide est d'environ 3.10^8 m.s^{-1}.

30. En supposant que le signal émis par les satellites se propage à la vitesse de la lumière dans le vide, donner une valeur approchée de la distance parcourue pendant la transmission d'un bit (qui est aussi la résolution de l'estimation de la distance par cette technique).

La résolution des récepteurs GPS est nettement inférieure à cette valeur. D'autres techniques sont donc nécessairement employées. L'une d'elles est de *suréchantillonner* les codes C/A, c'est-à-dire d'utiliser localement une fréquence d'échantillonnage multiple de 1,023 Mbit.s^{-1}. Chaque chiffre (-1 ou 1) de la séquence est donc écrit plusieurs fois au lieu d'une seule.

31. Écrire une fonction `surech` prenant pour arguments une liste `a` et un entier `k` et renvoyant la liste `a` suréchantillonnée d'un facteur `k` (par exemple, l'appel `surech([1,0],3)` doit renvoyer `[1,1,1,0,0,0]`).

En suréchantillonnant les codes, il est possible de tester des retards égaux à une fraction de la durée de transmission d'un bit, et donc d'atteindre une fraction de la résolution calculée précé-

Tournez la page S.V.P.

demment. Malheureusement, le procédé possède lui aussi ses limites, cette fois en ce qui concerne le temps de calcul. Si l'on appelle n le nombre d'échantillons utilisés pour représenter le code C/A, on rappelle que la corrélation de deux codes $a = (a_i)_{i\in\mathbb{N}}$ et $b = (b_i)_{i\in\mathbb{N}}$ est :

$$\langle a, b\rangle = \frac{1}{n}\sum_{i=0}^{n-1} a_i b_i$$

et que l'acquisition teste tous les retards possibles, ce qui implique de calculer n corrélations.

32. Donner et justifier la complexité asymptotique du calcul de ces n corrélations en fonction de n. D'après ce résultat, si l'on souhaite diviser la résolution par 100 et donc suréchantillonner d'un facteur 100, par combien cela multiplie-t-il le temps de calcul ?

La procédure d'acquisition ne peut donc donner qu'une estimation grossière de la distance entre le récepteur et le satellite. Pour affiner cette estimation, il faut utiliser une autre méthode.

E. Affinage de la mesure : la boucle à verrouillage de retard

Une fois la procédure d'acquisition terminée, le récepteur GPS dispose du code C/A correspondant au « bon » satellite et d'une mesure grossière du temps de propagation. Il doit alors affiner cette mesure, dans le double objectif :

— de « suivre » le satellite émetteur, c'est-à-dire de décoder son signal sans interruption malgré les variations de la distance satellite-récepteur et donc du retard de propagation ;
— de mesurer cette distance récepteur-satellite avec une précision permettant la localisation.

L'algorithme mis en œuvre pour cela s'appelle *boucle à verrouillage de retard*, ou DLL pour *Delay-Locked Loop*. Il consiste, à partir du code C/A local sélectionné lors de l'acquisition, à « caler » ce code sur le signal reçu en effectuant, périodiquement et en boucle, les trois opérations suivantes (figure 9) :

1. *estimer le décalage* du signal reçu vis-à-vis du code C/A local ;
2. *intégrer* par rapport au temps, ou plus simplement sommer, les décalages estimés ;
3. *retarder* le code C/A local d'un délai proportionnel à la sortie de l'intégrateur ; le code retardé est renvoyé à l'estimateur de décalage, formant un système bouclé.

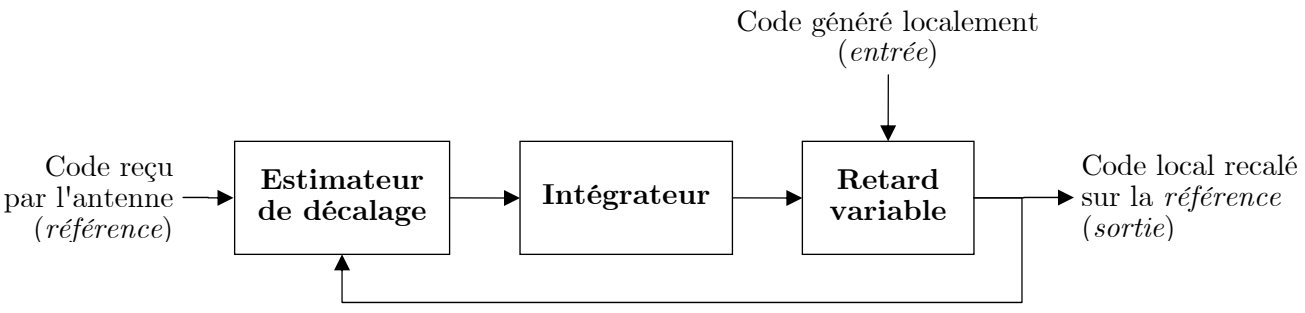

FIGURE 9 – Fonctionnement de la boucle à verrouillage de retard

E.1. Modélisation S.L.C.I. rudimentaire

Ce fonctionnement revient à *asservir le retard du code C/A local sur le retard de propagation*. Un modèle rudimentaire de cet asservissement est proposé figure 10 ; les variables de ce modèle ne sont pas les codes ou signaux eux-mêmes, mais leurs *retards* respectifs par rapport au code émis par le satellite. Plus précisément :

— $T_{\text{ref}}(p)$ représente le retard de propagation inconnu que l'on souhaite reproduire ;
— $T_{\text{e}}(p)$ représente le retard du code « d'entrée » obtenu lors de l'acquisition ;
— $T_{\text{s}}(p)$ représente le retard du code « de sortie » recalé par l'algorithme.

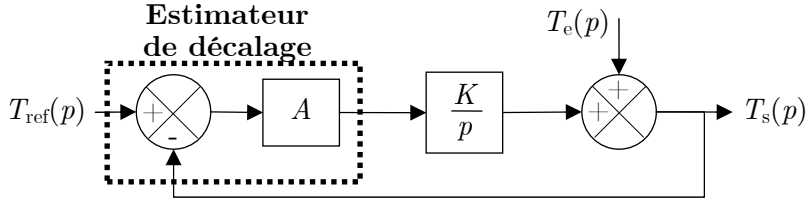

FIGURE 10 – Modèle rudimentaire de la boucle à verrouillage de retard

On souhaite vérifier qu'en régime permanent, le retard à la sortie $t_{\text{s}}(t)$ tend bien vers le temps de propagation du signal reçu (que l'on peut ainsi mesurer indirectement). Pour cela, on se propose d'exprimer $T_{\text{s}}(p)$ sous la forme $T_{\text{s}}(p) = H_1(p)T_{\text{ref}}(p) + H_2(p)T_{\text{e}}(p)$.

33. Exprimer les fonctions de transfert $H_1(p)$ et $H_2(p)$ sous forme canonique en fonction de A et de K. Justifier, d'après les formes obtenues, que ce modèle est stable.

On suppose ensuite que $T_{\text{ref}}(p)$ et $T_{\text{e}}(p)$ sont deux échelons d'amplitudes respectives T_{ref0} et T_{e0}, et on cherche la valeur finale de $t_{\text{s}}(t)$, dont la transformée de Laplace est $T_{\text{s}}(p)$. On rappelle que la transformée de Laplace d'un échelon d'amplitude B s'écrit B/p.

34. À l'aide du théorème de la valeur finale, montrer que lorsque le temps t tend vers l'infini, $t_{\text{s}}(t)$ tend vers T_{ref0} quelle que soit la valeur de T_{e0}. Conclure sur le fonctionnement de l'algorithme en régime établi.

35. Le résultat précédent aurait-il été encore vrai si l'algorithme n'avait pas comporté d'intégration, c'est-à-dire si le bloc K/p de la figure 5 avait été remplacé par un simple gain pur K ? On justifiera la réponse sans calcul.

Le modèle précédent ne permet pas de décrire le comportement de la boucle à verrouillage de retard en régime transitoire. En effet, il ne prend pas en compte la durée des différentes opérations (mis à part l'intégration, elles sont toutes modélisées par des gains purs) et il est formulé en temps continu alors que l'algorithme est par nature en temps discret. Pour observer le comportement transitoire de la boucle, il faut réaliser une simulation numérique.

E.2. Simulation du fonctionnement

Remarque : cette partie peut être traitée sans avoir préalablement traité la partie D. Néanmoins, elle fait appel à des notions définies dans cette partie (notamment « code C/A », « corrélation », « code retardé » et « suréchantillonnage ») et il est donc conseillé d'avoir lu attentivement l'énoncé.

Tournez la page S.V.P.

On se propose dans cette partie d'implanter l'algorithme présenté précédemment. On choisit pour cela de travailler sur des codes C/A suréchantillonnés d'un facteur 100, de sorte à obtenir une résolution temporelle égale à un centième de la durée de transmission d'un bit. On appelle :

— r le code « de référence » reçu par le récepteur ;

— s le code « de sortie » généré localement et retardé par l'algorithme.

L'*estimateur de décalage* fait appel à deux copies du code de sortie s, l'une avancée de 50 échantillons (soit la moitié de la durée de transmission d'un bit) et notée s^+, l'autre retardée de 50 échantillons et notée s^-. Son expression est $\epsilon = \langle s^+, r\rangle - \langle s^-, r\rangle$ où $\langle , \rangle$ désigne la corrélation.

On rappelle les spécifications des fonctions définies dans la partie D :

— `corr(a,b)` : prend deux listes `a` et `b` et renvoie un flottant égal à leur corrélation ;

— `retarde(a,k)` : prend une liste `a` et un entier relatif `k`, et renvoie une liste correspondant à `a` retardé de `k`. Si `k` est négatif, cela revient à avancer `a` ;

— `codeCA(p,q)` : renvoie la liste de longueur 1023 contenant le code C/A (sans suréchantillonnage) généré à partir des entiers `p` et `q` ;

— `surech(a,k)` : prend une liste `a` et un entier positif `k`, et renvoie la liste « suréchantillonnée », c'est-à-dire constituée des éléments de `a` présents `k` fois consécutives.

36. Proposer une fonction `decal` qui prend pour arguments deux listes `s` et `r` de même longueur (on ne demande pas de le vérifier) contenant respectivement le code de sortie et le code de référence, et renvoyant un flottant égal à l'estimateur de décalage défini ci-dessus. On appellera les fonctions `corr` et `retarde`.

La fonction `decal` étant définie, on exécute la suite d'instructions suivante, dont le résultat est donné figure 11 :

```
import matplotlib.pyplot as pp
c = codeCA(1,5) # correspond au satellite 1
cc = surech(c,100)
cr = list(retarde(cc,k) for k in range(-100,101))
d = list(-0.01*k for k in range(-100,101)) # signe moins pour avoir l'avance
e = list(decal(cr[i],cc) for i in range(len(cr)))
pp.plot(d,e)
pp.xlabel("Avance sortie sur ref (rapportée à la durée d'un bit)")
pp.ylabel("Sortie de l'estimateur de décalage")
```

37. Pour chacune des variables `cr`, `d` et `e`, indiquer s'il s'agit d'une liste « simple » ou d'une liste de listes et préciser à quoi correspond son `i`-ème élément.

Sur le schéma-bloc de la figure 10, on a modélisé l'estimateur de décalage par un comparateur suivi d'un gain pur A. Puisque les variables sont les retards respectifs des différents codes par rapport à une référence commune, ce modèle repose sur l'hypothèse que la sortie de l'estimateur est proportionnelle au décalage entre les deux codes.

38. Indiquer, à l'aide de la figure 11, dans quelle plage de décalages cette hypothèse est cohérente vis-à-vis du comportement de l'estimateur.

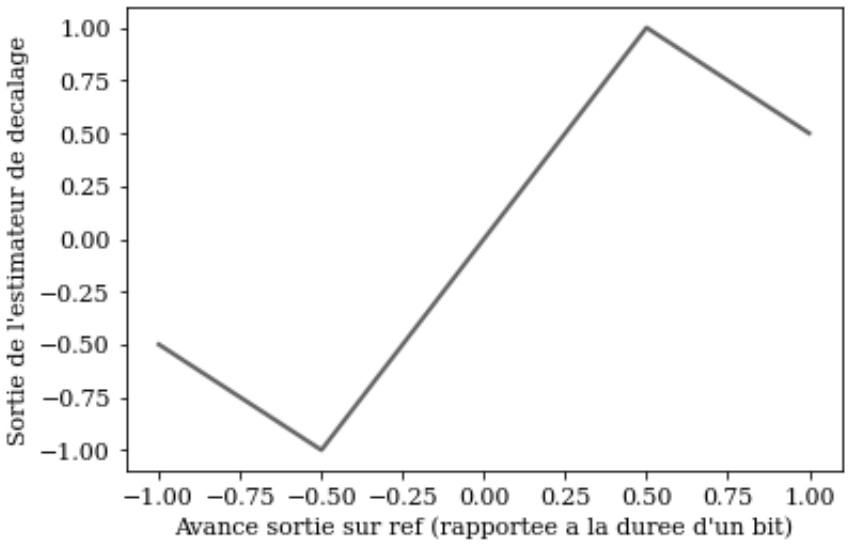

FIGURE 11 – Relation entrée/sortie de l'estimateur de décalage

On se limite à cette plage par la suite. On donne ci-dessous le code incomplet de la fonction `simu` qui simule la boucle à verrouillage de retard présentée sur la figure 9. Cette fonction prend quatre arguments en entrée :

- `ref` : liste contenant le code C/A de référence (choisi constant tout au long de la simulation) ;
- `entree` : liste contenant le code C/A d'entrée ;
- `imax` : durée de la simulation exprimée en nombre de périodes du code C/A (entier) ;
- `K` : « gain » de l'« intégrateur » (flottant).

```
def simu(ref, entree, imax, K):
    retard = 0.0
    sortie = entree[:]
    sorties, retards = [sortie],[retard]
    # boucle répétée à chaque période du code C/A
    for i in range(imax-1):
        # 1. on estime le décalage
        ecart = # LIGNE À COMPLÉTER NUMÉRO 1
        # 2. on "intègre" (somme) les écarts avec un facteur K
        retard = # LIGNE À COMPLÉTER NUMÉRO 2
        # 3. on retarde l'entrée pour obtenir la sortie
        sortie = # LIGNE À COMPLÉTER NUMÉRO 3
        # écriture des résultats de la simulation
        sorties.append(sortie)
        retards.append(retard)
    return sorties, retards
```

On précise que les trois opérations (estimation, sommation et retard du code d'entrée) sont effectuées une fois par période du code C/A, soit toutes les millisecondes. Chaque tour de la

Tournez la page S.V.P.

boucle `for` simule donc le fonctionnement de l'algorithme sur une période. On précise également que cette simulation ne tient pas compte des contraintes liées au fonctionnement en temps réel sur le récepteur GPS, mais vise uniquement à donner une première approximation du comportement de l'algorithme en régime transitoire.

Pour simplifier le code, l'« intégrateur » est réalisé par une somme discrète : la variable `sortie` est donc la somme cumulée des décalages estimés (variable `ecart`) multipliés par le « gain » `K`.

39. En appelant les fonctions nécessaires, compléter les trois lignes indiquées de la fonction `simu`. Ne pas oublier que la sortie de l'estimateur de décalage est un flottant, tandis que le nombre d'échantillons dont on retarde un code est nécessairement un entier.

On a réalisé une simulation sur 20 périodes (soit 20 millisecondes) à partir du code C/A du satellite 1 suréchantillonné d'un facteur 100. L'entrée (générée localement) est le code lui-même et la référence (reçue via l'antenne) est le code retardé de 20 échantillons, soit 20 centièmes de bit. Trois valeurs du gain K ont été testées. La figure 12 donne :

— à gauche, l'évolution du retard en sortie de l'intégrateur (qui doit tendre vers 20 pour aligner la sortie sur la référence),

— à droite, la corrélation de la sortie avec la référence (qui tend vers 1 si la sortie s'aligne parfaitement avec la référence).

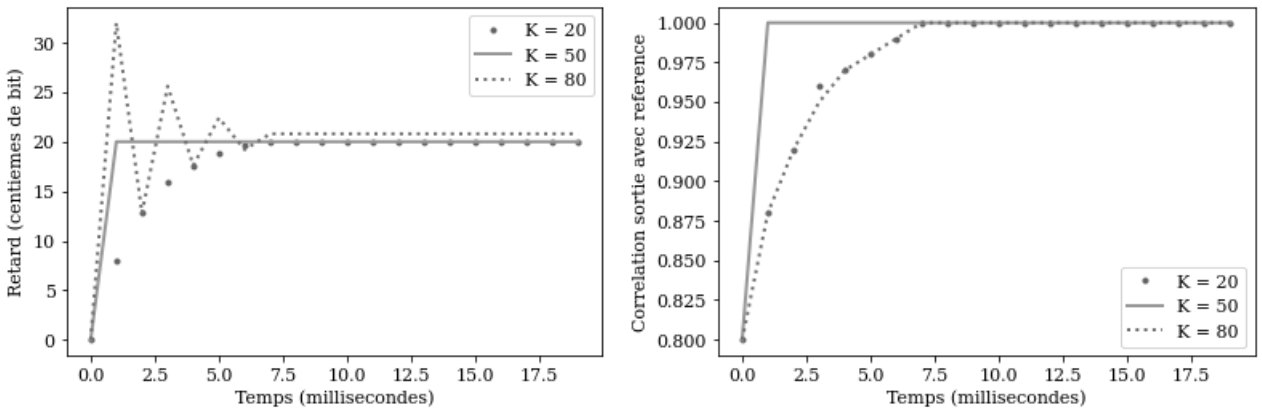

FIGURE 12 – Simulation de la boucle à verrouillage de retard pour trois valeurs de K

Les deux principales différences entre cette simulation et le modèle S.L.C.I. précédent sont la quantification du temps (qui est un nombre entier de périodes) et la quantification du retard appliqué à l'entrée (qui est un nombre entier de centièmes de bit).

40. Indiquer, en justifiant, si les oscillations (obtenues pour $K = 80$) et la convergence en un temps fini (obtenue sur les trois courbes) auraient pu être prédites par le modèle S.L.C.I.

Cette simulation montre que la sortie s'aligne bien sur la référence : il est donc théoriquement possible d'utiliser les codes C/A pour mesurer des distances satellite-récepteur avec une résolution d'un centième de bit. En pratique, cette valeur théorique est rarement atteinte car il est difficile de compenser les autres sources d'erreur, comme les décalages modélisés dans la partie C.

Troisième partie

EXPLOITATION DE LA LOCALISATION

Les opérations étudiées dans la partie précédente permettent au récepteur de mesurer la distance qui le séparent des satellites, tout en récupérant la position de ceux-ci contenue dans le message de navigation. Il peut alors se localiser à l'aide d'un calcul géométrique non abordé ici. Cette partie présente une exploitation de la localisation dans le cadre d'une recherche d'itinéraires.

F. Transmission de la localisation : les trames NMEA

Pour être exploitée par des applications tierces ou par d'autres matériels, la localisation déterminée par le récepteur GPS doit être transmise. Un format normalisé souvent utilisé à cet effet est le protocole NMEA 0183, qui consiste à structurer les informations sous la forme de *trames*.

Une trame NMEA est une chaîne de caractères contenant plusieurs données (les *champs* de la trame) séparées par des virgules. Il en existe différents types. Nous nous limitons dans ce sujet aux trames dites GGA. Un exemple de trame GGA est :

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E
```

La signification des différents champs est la suivante :

— `$GPGGA` : type de trame ;
— `064036.289` : heure d'envoi au format `'HHMMSS.SSS'` (ici 06 h 40 min 36,289 s) ;
— `4836.5375` : latitude en valeur absolue (ici 48 degrés 36,5375 minutes) ;
— `N` : sens de la latitude (`N` pour Nord, `S` pour Sud) ;
— `00740.9373` : longitude en valeur absolue (ici 7 degrés 40,9373 minutes) ;
— `E` : sens de la longitude (`E` pour Est, `W` pour Ouest) ;
— `1` : type de positionnement (non abordé ici) ;
— `04` : nombre de satellites utilisés pour la localisation ;
— `3.2` : indicateur de précision (non abordé ici) ;
— `200.2` : altitude (ici 200,2) ;
— `M` : unité de l'altitude (ici, des mètres) ;
— `,,,,0000` : champs non utilisés (d'autres informations peuvent y être inscrites) ;
— `*` : séparateur entre les champs et la somme de contrôle ;
— `0E` : somme de contrôle.

Pour extraire les champs de la trame, on propose d'utiliser la méthode `split` qui découpe une chaîne de caractères autour d'un caractère passé en argument (le « délimiteur »), et renvoie une liste de chaînes. Par exemple, si on définit la chaîne :

```
trame = '$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E'
```


Tournez la page S.V.P.

alors l'appel `trame.split(',')` renvoie la liste de chaînes suivante :

```
['$GPGGA','064036.289','4836.5375','N','00740.9373','E','1','04','3.2','200.2','M','','',
'','0000*0E']
```

On rappelle que si `s` est une chaîne et `i` et `j` deux entiers positifs ou nuls, `s[i:j]` renvoie la portion de `s` dont les indices vont de `i` inclus à `j` exclu.

41. Écrire une fonction `heure` qui prend pour argument la chaîne `trame` au format ci-dessus et renvoie un tuple `(h,m,s)` où `h` et `m` sont des entiers contenant respectivement les heures et les minutes, et `s` est un flottant contenant les secondes (avec les millisecondes).

Par convention, on représente les latitudes par des flottants signés contenant leur valeur en degrés, une latitude nord (`'N'`) étant positive et une latitude sud (`'S'`) étant négative. On rappelle qu'une minute d'angle (notée ') correspond à un soixantième de degré.

42. Écrire une fonction `latitude` qui prend pour argument la chaîne `trame` au format ci-dessus, et renvoie un flottant contenant la latitude signée exprimée en degrés.

Ces informations ne sont néanmoins fiables que si la trame a été transmise sans erreur. La somme de contrôle située à la fin de la trame permet de détecter certaines erreurs de transmission.

La somme de contrôle est un *OU exclusif bit à bit* entre les codes ASCII de tous les caractères situés entre le `$` (qui est toujours au début) et le `*`, tous deux exclus. Le OU exclusif bit à bit de deux entiers `a` et `b` est un entier, calculé par l'instruction `a^b` en Python, dont la définition formelle n'est pas utile ici. Les deux caractères situés après le `*` doivent correspondre à l'écriture hexadécimale de cet entier pour que la transmission soit validée.

Exemple : vérifions la somme de contrôle de la trame simplifiée `'$0,1*2D'`.
- Les trois caractères situés entre le `$` et le `*` sont `'0'`, `','` et `'1'` ;
- Leurs codes ASCII respectifs sont 48, 44 et 49 (en décimal) ;
- L'instruction `48^44^49` renvoie l'entier 45, dont l'écriture hexadécimale est `'2D'` ;
- Cela correspond bien aux deux caractères situés après le `*` : la trame est donc valide.

Pour programmer cette vérification, on donne les indications suivantes :
- si `s` est une chaîne, `s.find('a')` recherche la première apparition du caractère `'a'` dans `s` et renvoie son indice (ou -1 s'il n'est pas trouvé) ;
- si `c` est un caractère (c'est-à-dire une chaîne de longueur 1), `ord(c)` renvoie le code ASCII de `c`, qui est un entier compris entre 0 et 127 ;
- le OU exclusif bit à bit est commutatif (`a^b == b^a`) et associatif (`a^(b^c) == (a^b)^c`) ;
- le calcul du OU exclusif bit à bit peut être initialisé à zéro car quel que soit l'entier `i`, `0^i` est toujours égal à `i` ;
- si `s` est une chaîne contenant la représentation hexadécimale d'un entier (dont les chiffres vont de `0` à `F`), l'instruction `int(s,16)` permet d'obtenir l'entier correspondant.

43. Écrire une fonction `controle` qui prend pour argument la chaîne `trame` et renvoie `True` si la somme de contrôle correspond bien au résultat du calcul précédent, et `False` sinon.

G. Calcul du plus court chemin

Les récepteurs GPS utilisés pour les trajets automobiles disposent de bases de données géographiques et d'applications mettant en oeuvre des algorithmes de recherche de plus courts chemins. Un des plus classiques est l'algorithme de *Dijkstra*.

Pour pouvoir utiliser celui-ci pour déterminer les plus courts chemins, les différents lieux constituent les sommets d'un graphe et les distances entre deux lieux sont enregistrées comme les poids associés à chaque arête. Pour illustrer, les sommets (les lieux) sont codés par des nombres entiers et on utilisera le graphe pondéré représenté ci-dessous comme exemple.

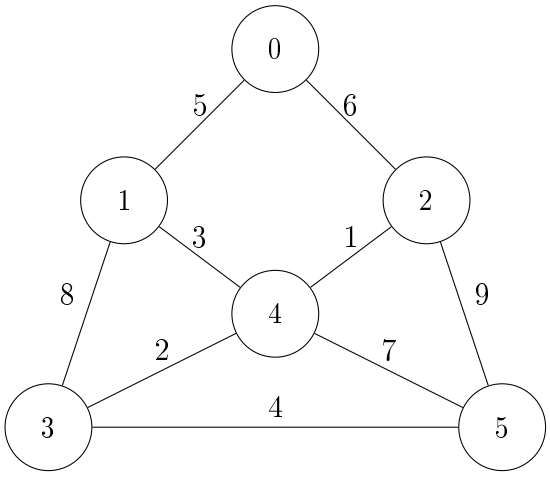

FIGURE 13 – Exemple de graphe pondéré

44. Le graphe (exemple ci-dessus) sera codé par un dictionnaire d'adjacence dont on donne ci-dessous les 3 premières lignes :

```
{0 : [(1,5), (2,6)],
 1 : [(0,5), (3,8), (4,3)],
 2 : [(0,6), (4,1), (5,9)],
 ...
 ...
 ...
 }
```

Recopier et compléter ce dictionnaire. Pourquoi un dictionnaire est-il, sur l'exemple proposé, préférable à une matrice d'adjacence ? Quelle méthode (la décrire brièvement en 3 à 4 phrases maximum) est à la base des performances des dictionnaires en termes de temps d'accès ?

Tournez la page S.V.P.

Description de l'algorithme :

Soit n le nombre de sommets du graphe. On cherche les distances minimales entre un sommet de départ donné et les autres sommets de la manière suivante :

- On utilise une liste de taille n appelée `distances` dont les éléments représentent les distances entre le sommet de départ et chacun des sommets. Cette distance est initialisée à 0 pour le sommet de départ et à -1 (qui représente l'infini) pour les autres ;
- On va visiter les différents sommets du graphe, et il faut savoir lesquels ont déjà été visités ou non. On utilise une liste `deja_visites` de taille n de booléens initialement tous à `False` ;
- On itère avec une boucle `while`, la condition d'arrêt étant le fait que la liste `deja_visites` ne contient que des `True`. Pour chaque itération :
  - On cherche parmi les sommets pas encore visités celui qui est à la plus petite distance du sommet de départ. On l'appelle `en_cours` pour la suite ;
  - On visite ce sommet `en_cours` :
    - On met ce sommet à `True` dans la liste `deja_visites` ;
    - On parcourt ses voisins et pour chacun d'entre eux qui n'a pas encore été visité (appelé `voisin`) on actualise dans la liste `distance` la valeur le concernant : on la remplace par le minimum entre l'ancienne distance et la distance entre le sommet de départ et `en_cours` à laquelle on ajoute la distance entre `en_cours` et `voisin`.

45. Donner les états successifs de la liste `distances` au cours de l'exécution de l'algorithme sur l'exemple de la figure 13, le sommet de départ étant le sommet 0.

46. Écrire une fonction `sommet_min(dist,deja_visites)` qui prend en argument une liste `dist` d'entiers positifs ou égaux à -1 et une liste `deja_visites` de booléens et qui renvoie l'indice du minimum de `dist` parmi ceux dont la valeur dans la liste `deja_visites` correspond à `False` et qui n'ont pas pour valeur -1 . Si aucun indice dans la liste ne convient, cela signifie que l'algorithme est terminé, on renverra `None`.

47. Écrire une fonction `actualiser(k,dist,deja_visites,g)` qui prend en entrée un graphe `g` sous forme d'un dictionnaire d'adjacence, deux listes `dist` et `deja_visites` de taille n et un entier `k` compris entre 0 et $n-1$ (où n est la taille du graphe), et qui met à jour les listes `dist` et `deja_visites`.

48. En utilisant les fonctions précédentes, écrire une fonction `dijkstra(g,depart)` prenant en entrée un graphe `g` sous forme de dictionnaire d'adjacence et un sommet `depart` et renvoyant la liste des distances minimales au sommet `depart`.

H. Statistiques sur les durées des trajets

L'algorithme vu dans la partie précédente permet de déterminer le trajet le plus court entre deux points donnés. Il peut également déterminer le trajet le plus rapide, c'est-à-dire qui prend le moins de temps. Pour cela, il faut disposer d'informations sur la durée des trajets. En pratique, ces informations peuvent provenir de données obtenues en temps réel sur la circulation, ou encore de statistiques préalablement collectées.

Dans cette partie, on suppose que l'on dispose de données sur des trajets individuels, stockées dans une base de données SQL constituée de deux tables. Ces tables sont :

1. `noeuds`, qui décrit chaque nœud (ou sommet) par les attributs suivants :
   — `id` (entier) : identifiant du nœud (clé primaire) ;
   — `num` (entier) : numéro de la voie de l'adresse ;
   — `voie` (chaîne) : type (rue, avenue...) et nom de la voie de l'adresse ;
   — `code` (entier) : code postal de l'adresse ;
   — `ville` (chaîne) : ville de l'adresse.
2. `trajets`, qui décrit chaque trajet enregistré par les attributs suivants :
   — `id_from` (entier) : identifiant du nœud de départ ;
   — `id_to` (entier) : identifiant du nœud d'arrivée ;
   — `date` (chaîne) : date de départ du trajet au format `"AAAA-MM-JJ"` ;
   — `heure` (chaîne) : heure de départ du trajet au format `"HH:MM:SS"` sur 24 heures ;
   — `durée` (entier) : durée du trajet en secondes.

49. Écrire une requête renvoyant les heures de départ et les durées des trajets ayant débuté le 1er avril 2025 et allant du nœud 123 au nœud 456 (les numéros sont les identifiants).

50. Écrire une requête renvoyant la date de départ, l'heure de départ et la durée des trois trajets les plus courts (c'est-à-dire dont les durées sont les plus petites) allant du nœud 123 au nœud 456, ordonnés par durées croissantes.

51. Écrire une requête renvoyant le nombre de trajets enregistrés qui partent du numéro 7 de la voie nommée « rue des Plantes » située dans la ville de Paris.

52. On considère les trajets *vers* la ville de Lyon ayant débuté le 1er avril 2025. Écrire une requête renvoyant, pour *chaque* ville de départ pour laquelle *au moins dix* trajets de ce type existent, le nom de la ville suivi des durées minimale, moyenne et maximale des trajets.

Fin de l'énoncé

IMPRIMERIE NATIONALE – 25 1010 – D'après documents fournis