ENSG Géomatique

École Nationale des Sciences Géographiques

Rapport de stage

Test de GNSS low-cost

Lucas DASSIN - Hugo TARDY -
Melvin PICHENOT - Ianis PONTIER
ING1

Professeur référent : M. Jacques BEILIN

8 Juillet 2019 — 24 Juillet 2019

Sommaire

1 Introduction

1.1 Contexte

Depuis quelques années, le marché des GNSS low-cost, puces GPS utilisées dans les smartphones et autres GPS de randonnée est en expansion. Avec l'amélioration de la précision de ces puces, les tarifs des puces de précision subdécimétriques deviennent raisonnables. Le but de ce projet est d'évaluer la possibilité de fabriquer et d'utiliser des GNSS centimétriques pour quelques centaines d'euros.

1.2 Sujet

En 2018, un groupe d'IT1 a construit une paire de récepteurs GNSS. Par ailleurs, nous disposons de deux instruments construits par Franklin Nguyen. Un logiciel a été développé lors du projet geodev2: EasyGNSS.

L'objectif du projet est d'utiliser les deux types de récpeteurs GNSS en testant le logiciel développé par les IT2. Leur code se base sur celui de Taro Suzuki. Ce dernier est lui-même une implémentation Python du code RTKBase écrit en C++ par des élèves de l'ENSG.

Une campagne de tests terrain sera menée à bien en utilisant les différents modes de positionnement disponibles: statique, cinématique, RTK, NTRIP sur des précisions centimétriques à décimétriques... Un accès au NTRIP IGN sera possible.

1.3 Quelques notions sur les GNSS

1.3.1 GNSS

On appelle GNSS (Global Navigation Satellite System, pour système global de positionnement par satellite) les systèmes de positionnement basé sur des signaux émis de satellites en orbite autour de la Terre et fournissant une couverture mondiale.

1.3.2 Fonctionnement

Le fonctionnement des GNSS repose sur la mesure du temps de propagation du signal émis par un satellite jusqu'à sa mesure par un récepteur. La mesure du temps de propagation du signal en provenance de plusieurs satellites permet par intersection de déterminer la position du récepteur.

Une autre technique que la mesure de code utilisable pour le positionnement par GNSS repose sur la mesure du déphasage entre les signaux reçus et générés par le récepteur. Bien sûr, les horloges récepteur et émetteur (satellite) n'étant pas synchronisées, le déphasage mesuré est entaché d'une erreur de synchronisation qu'il est toujours nécessaire d'estimer.

1.3.3 Précision

Ce positionnement est réalisé de manière instantanée, avec une précision d'une dizaine de mètres, n'importe quand, n'importe où sur la Terre, quelle que soit la météo et à un faible coût.

1.3.4 Erreurs sur les mesures GNSS

- Erreurs liées aux satellites; désynchronisation horloge satellites/récepteur, imprécision relative du point physique d'émission du signal satellite (centre de phase) et de la position du satellite
- Erreurs liées à la propagation; modification de la propagation du signal lors de la traversée de la troposphère (dues aux conditions météorologiques) et de la ionosphère (dues aux charges libres)
- Erreurs liées à la station; masque, trajets multiples, position physique du centre de réception

1.4 Méthodes de positionnement

On distingue deux modes de positionnement: absolu et relatif. A partir de ces deux modes de positionnement on peut réaliser différentes mesures, suivant la précision désirée pour les résultats, résumées ci-après synthétiquement:

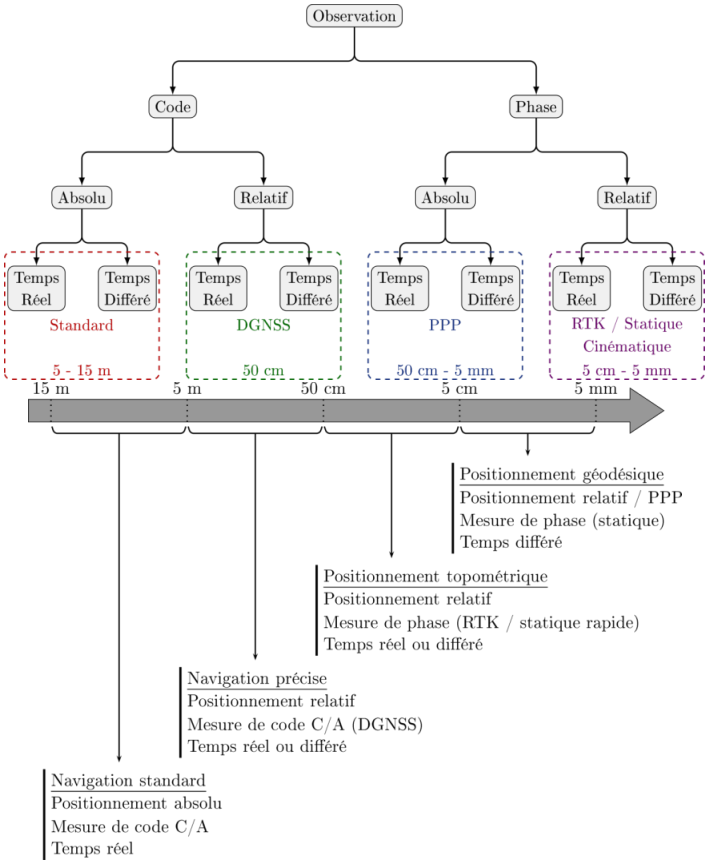

fig.1 Synthèse des méthodes de positionnment
GNSS, P.BOSSER

Nous nous intéresserons ici principalement au RTK et au NTRIP.

RTK : Real Time Kinematic, positionnement relatif sur la phase en temps réel. La précision est de l'ordre de 5 cm. Les données sont envoyées par radio.

Le NTRIP est un mode de transfert de données qui passe par Internet utilisable en RTK.Pour mettre en place le NTRIP, il ne faut qu'une connexion Internet et un récepteur GNSS.

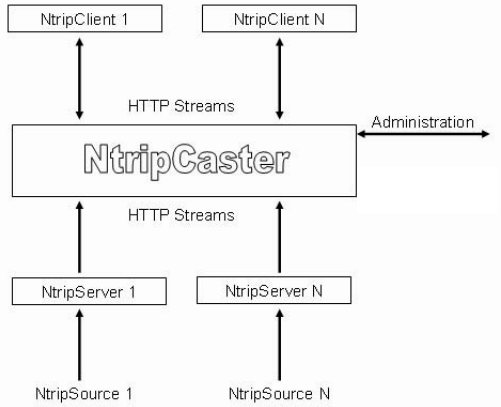

fig.2 Diagramme des relations entre les différents acteurs NTRIP (Trimble, 2004)

1.5 Le format RINEX

Les données acquises par un récepteur GNSS sont habituellement stockées et échangées dans le format RInEx (Receiver Independent Exchange Format) :

- Les fichiers d'observations RINEX contiennent les observations GNSS brutes : mesure de codes, de phases. Ces fichiers peuvent être disponibles sous forme compressée (compression Hatanaka).
- Les fichiers de navigation RINEX contiennent les éphémérides de chaque satellites, les paramètres de correction d'horloge, etc.

Ces fichiers RINEX sont une retranscription des données brutes stockées en binaires dans un language plus adapté et normalisé. C'est pourquoi on travaillera avec ce format.

2 Organisation

2.1 Composition de l'équipe

Notre groupe est constitué de 4 membres: TARDY Hugo, DASSIN Lucas, PICHENOT Melvin, PONTIER Ianis.

2.2 GANTT

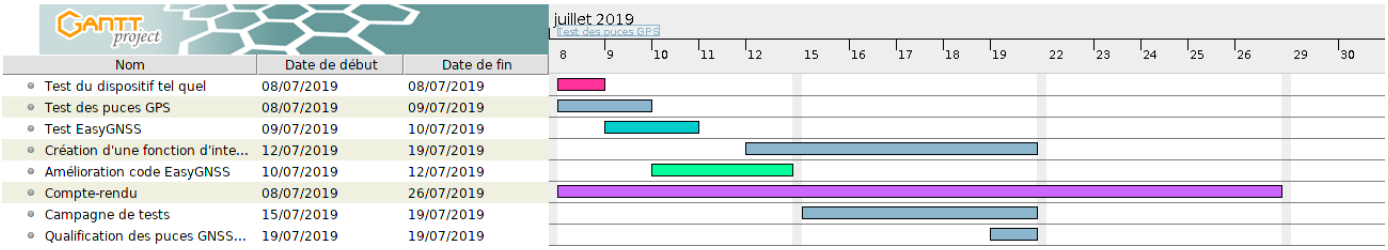

fig.3 Diagramme de GANTT du projet

2.3 Liste du matériel

- 2 récepteurs GNSS low-cost blancs fabriqués à Forcalquier par les étudiants ingénieurs de 1ère année en 2018. Ils ont été nommés GNSS1 et GNSS3. Fonctionne avec le protocole de transfert NTRIP car les GNSS1 et GNSS3 sont équipés de cartes wifi, mais pas de radio. Le software utilisé est une adaptation de RTKLib codé en Python appelé EasyGNSS développé par des ingénieurs 2ème année de l'ENSG en 2019.

fig.4 GNSS1 et GNSS3

- 2 récepteurs GNSS low-cost noirs construits et prêtés par Franklin NGUYEN, nommés GNSS2 et GNSS4. Fonctionne avec le protocole de transfert RTK car ils possèdent une radio mais ne possèdent pas de cartes wifi. Le software utilisé est une adaptation de RTKLib codé en C++ appelé RTKBase.

fig.5 GNSS2 et GNSS4

- Un récepteur GNSS NETR5 et une antenne Zéphyr, de précisions centimétriques
- 1 ordinateur portable de terrain
- Câbles en tout genre
- Trépied classique
- Pied de cercle pour l'utilisation du récepteur Trimble NetR5
- Embase pour centrer et buller nos récepteurs
- Batteries en tout genre

3 Déroulement

3.1 Contexte

Dans un premier temps, nous devions qualifier la précision des résultats fournis par nos GNSS low cost par l'application EasyGNSS dans différents modes d'acquisition : RTK, cinématique.

3.2 Méthodologie

Tout d'abord, nous avons dû vérifier que nos puces GNSS seules fonctionnaient correctement. Pour cela, nous avons relié la puce GNSS à un ordinateur sur lequel nous avons lancé le logiciel propriétaire du constructeur de la puce: u-blox. Nous nous sommes placés dans un endroit où il n'y avait pas de masque et nous avons lancé l'acquisition depuis le logiciel. Une fois cela fait, il nous a fallu encore convertir le fichier .ubx obtenu en fichier d'observation RINEX à l'aide de RTKConv. Enfin, il ne reste plus qu'à l'envoyer au service de calcul en ligne de l'ENSG, développé par Jacques BEILIN, professeur à l'ENSG. Celui ci fonctionne en téléchargeant les données des stations permanentes proches puis calcule les lignes de bases et résout les ambiguïtés pour déterminer la position du récepteur lors de l'acquisition.

3.3 Tests de puces GNSS

Pour commencer, nous avons donc testé chacune des puces GNSS en-dehors de leur boîtier branchée sur l'ordinateur de terrain. La réception des données est lancée par le logiciel u-center du fabricant des puces: u-blox.

Il apparaît tout d'abord les décalages d'horloge suivants:

- GNSS1: 5ms
- GNSS2: 8ms
- GNSS3: 7ms
- GNSS4: 12ms

Cependant, on constate que les résultats des calculs de coordonnées entre 5 et 10ms donnent des résidus très élevés et au-delà de 10ms, les calculs divergent et ne permettent pas de trouver de solutions.

		écarts (m)			
	Zephyr	GNSS1	GNSS2	GNSS3	GNSS4
N	6322115,751	6,296	0,204	26,575	non abouti
E	922608,194	4,049	1,535	6,731	
h	601,547	0,306	2,921	49,619	

fig.6 écarts de coordonnées pour les puces GNSS

N'ayant pas remarqué immédiatement ces décalages d'horloges, nous avons continué avec les GNSS dont les calculs aboutissaient.

3.3.1 Mise en place

Dans l'hypothèse d'interférences entre les composants dont nos prédécesseurs nous avaient parlé lors de leur fabrication, nous avons réalisé une série de tests dans des configurations différentes:

- éclaté : antenne, puce GNSS et raspberry étaient séparés les uns des autres
- boitier : la puce GNSS et le raspberry étaient proches, tandis que l'antenne GNSS était mise à l'écart
- solidaire : antenne, puce GNSS et raspberry étaient dans la boite, côte à côte

Durant nos observations, nous nous sommes rendus compte que les époques de nos fichiers RINEX (.obs) obtenus à l'issue des acquisitions de notre récepteur étaient en décalage avec les époques des stations permanentes (de l'odre de quelques millisecondes à une quinzaine de millisecondes) et nous l'avons quantifié dans les différentes configurations. Nous avons obtenu les résultats suivants:

		Valeur en ms de décalage			
		min	max	moyenne	écart type
GNSS1	solidaire	2,9	4	3,6	0,5
	boîtier seul	7,99	9	8,4	0,5
	éclaté	9,9	11	10,4	0,5
GNSS3	solidaire	2,9	3	3	0
	boîtier seul	5,9	7	6,1	0,3
	éclaté	12,9	13	12,9	0

fig.7 Décalages d'horloges sur les récepteurs

On constate que les résultats ne sont pas meilleurs dans une configuration éclatée, voire sont pires que dans une configuration solidaire.

Dans tous les cas, la tolérance d'une milliseconde est largement dépassée et ne permet pas d'assurer des calculs corrects de positionnement. En effet, pour garantir des résultats, les horloges internes des récepteurs GNSS sont censées être asservies au temps GPS de l'USNO (United States Naval Observatory). un décalage de plusieurs millisecondes entraine un défaut de qualité sur la résolution des ambiguïtés.

fig.8 Le test avec l'ensemble puce/antenne séparé du boîtier contenant le Raspberry Pi

fig.9 Ce boîtier permet de fixer sur le trépied notre ensemble puce GNSS/antenne réceptrice

3.4 Créations d'une fonction d'interpolation et d'écriture de fichiers RINEX

C'est pourquoi, le calcul en ligne ne trouvait aucune solution à nos observations. Ainsi, nous avons dû créer un outil agissant sur les fichiers d'observations dans le but d'interpoler les valeurs des observables aux bonnes

époques.

Pour cela, nous nous sommes servis de la bibliothèque *gnsstoolbox* développé par Jacques BEILIN. Dans cette bibliothèque, chaque fichier RINEX est considéré comme un objet *rinex_o* qui contient des objets header qui chacun contiennent des objets *epoch* qui chacun contiennent des objets *sat*.

Tout d'abord, n'ayant pas idée de l'évolution des observables C1 et L1 (les deux suffisant pour le calcul en ligne, nous nous sommes contentés d'interpoler seulement ces deux observables), nous avons créé une fonction d'interpolation de Lagrange car celles-ci est utilisée pour le calcul des orbites. Nous avons commencé par des interpolations de degré 9. Mais, par la suite, nous avons également testé le degré 7,5,3 et 1. Tous les calculs s'avérant concluant par la suite, nous en avons conclu que C1 et L1 étaient des fonctions linéaires, sinon quasi-linéaires dans le cas de nos observations.

Pour obtenir les éphémérides à une époque, on utilise un polynôme de Lagrange.

$$y(t) = \sum_{j=0}^{m} L_j(t).y(t_j)$$

où

$$L_j(t) = \prod_{k=0}^{m} \frac{(t - t_k)}{(t_j - t_k)}, k \neq j$$

où m est l'ordre du polynôme, $y(t_j)$ est la valeur à l'instant t_j, $L_j(t)$ est appelée fonction de base d'ordre m et t est l'instant où la donnée est interpolée.

m est généralement un nombre impair.

Le polynôme de Lagrange est utilisé sur chacune des coordonnées du satellite ainsi que sur l'erreur d'horloge.

```
19  7 12  6 36  4.9930000  0  9G25G12G29G32G24G 2G14G31G 6
 18443867.253    96923176.7702        184.870          50.000
 18582554.150    97651972.8832      -2234.496          50.000
 19527017.275   102615167.0462       2245.092          51.000
 20004799.281   105125936.0942      -1153.269          30.000
 20324982.664   106808507.0682      -3647.105          48.000
 20880409.309   109727297.0522        109.673          47.000
 20677887.492   108663021.1972        526.048          46.000
 21630010.774   113666471.6532       2595.477          39.000
 21739891.009   114243897.8142      -1781.708          46.000
```

fig.10 exemple de fichier RINEX avant interpolation

```
19  7 12  6 36  5.0000000  0  9G25G12G29G24G 2G14G31G 6G32
 18443867.008    96923175.478
 18582557.128    97651988.527
 19527014.285   102615151.332
 20324987.523   106808532.600
 20880409.163   109727296.287
 20677886.792   108663017.516
 21630007.314   113666453.486
 21739893.382   114243910.287
 20004800.786   105125944.171
```

fig.11 exemple de fichier RINEX après interpolation

Après avoir interpolé une de nos acquisitions à différents degrés, on observe une amélioration importante des résidus, et l'impact des degrés est négligeable sur les coordonnées.

		Valeur en ms de décalage			
		min	max	moyenne	écart type
GNSS1	solidaire	2,9	4	3,6	0,5
	boîtier seul	7,99	9	8,4	0,5
	éclaté	9,9	11	10,4	0,5
GNSS3	solidaire	2,9	3	3	0
	boîtier seul	5,9	7	6,1	0,3
	éclaté	12,9	13	12,9	0

fig.12 exemple de fichier RINEX avant interpolation

3.5 Split avec Trimble NetR5

Afin de tester notre interpolation, nous avons réalisé une acquisition à l'aide d'un splitter qui permet de partager le signal d'une antenne entre deux récepteurs. On utilise une antenne Zéphyr de qualité géodésique, à laquelle on connecte un récepteur GNSS NetR5, mais aussi notre récepteur low-cost.

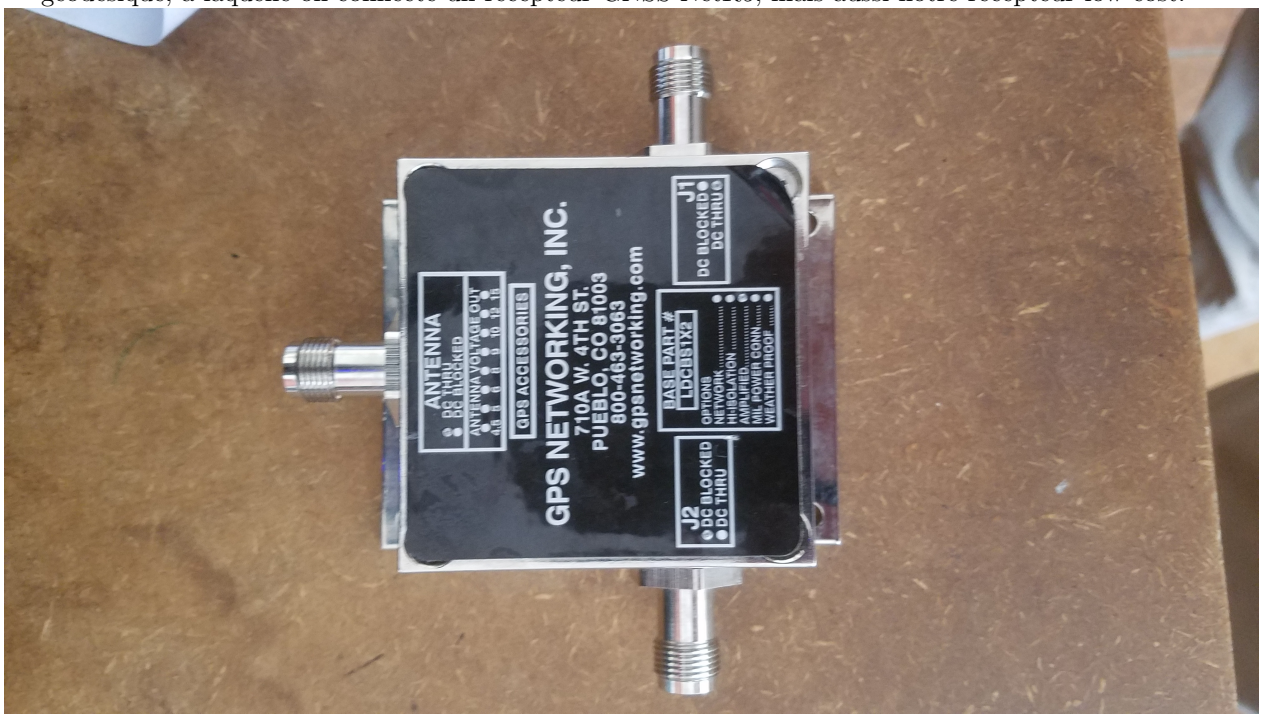

fig.13 Le récepteur low-cost, l'ordinateur de terrain et le boitier Trimble NetR5

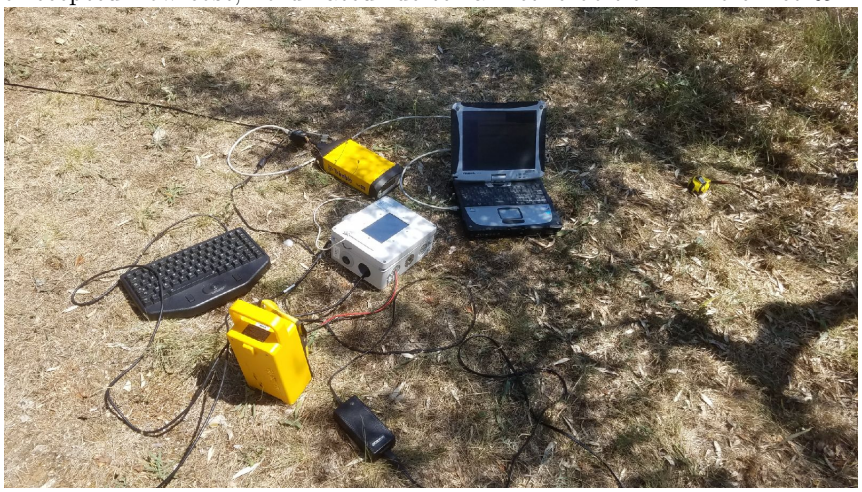

fig.14 Le récepteur low-cost, l'ordinateur de terrain et le boitier Trimble NetR5

fig.15 Le récepteur low-cost, l'ordinateur de terrain et le boitier Trimble NetR5

fig.16 Récepteur Trimble NetR5

fig.17 Installation globale

Après avoir interpolé nos acquisitions, les résultats du récepteur 1 sont satisfaisants en coordonnées Lambert93 planimétriques. Un écart de seulement 2cm est constaté tandis qu'en coordonnées Lambert93 altimétriques le décalage est lui de 40cm.

Les résultats du récepteur GNSS3 sont eux par contre très mauvais, ainsi on observe un décalage de plusieurs mètres, probablement dû à un problème d'acquisition.

	Résultats obtenus exprimés en Lambert 93 lors d'un split avec le Trimble NetR5					
	Trimble NetR5 (split GNSS1)	GNSS1	écart (m)	Trimble NetR5 (split GNSS3)	GNSS3	écart (m)
E	922608,016	922608,024	0,01	922607,995	922623,466	15,47
N	6322115,741	6322115,74	0,00	6322115,729	6322121,322	5,59
H	601,442	601,024	0,42	601,464	605,755	4,29

fig.18 Références obtenues avec le Trimble NetR5 pendant le split avec GNSS1

3.6 Décalages d'observables C1 et L1

Un second critère de vérification de la qualité d'interpolation a été de mesurer la différence entre les observables des satellites, d'une époque à l'autre. Pour s'assurer que l'interpolation n'est pas erronée, nous avons donc développé une méthode *rinex_o* qui permet de parcourir un fichier RINEX dans le but de comparer les valeurs des observables C1 et L1 entre deux époques pour s'assurer qu'il n'y ait pas d'écarts aberrants. Après calculs et affichage, on observe que le rapport entre le C/A code et la différence de phase est constant et donne près de 19 cm, ce qui est cohérent avec la longueur d'onde radio utilisé par les satellites de positionnement.

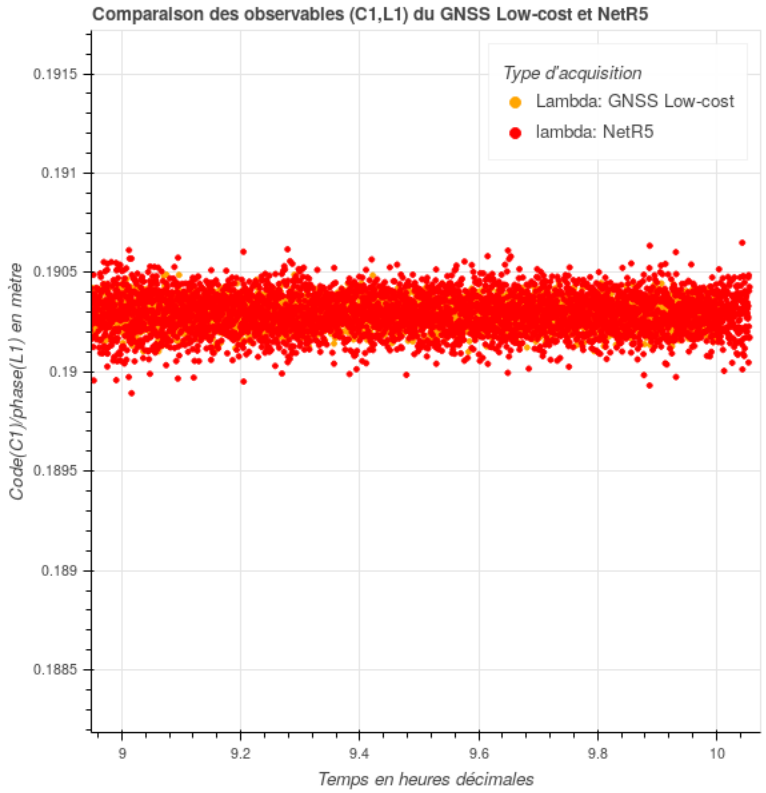

fig.19 rapport de C1 sur L1, donnant la longueur d'onde utilisée

Les graphiques représentant la différence d'observables ont aussi confirmé que les décalages de phase des satellites pour chaque observable étaient proches, comme indiqué sur la figure ci dessous dans le cas particulier d'un seul satellite. L'aspect global des courbes est semblable, cependant des aspects sont gommés par l'échelle, c'est pourquoi on se ramène à une figure plus adapté aux observations de détails.

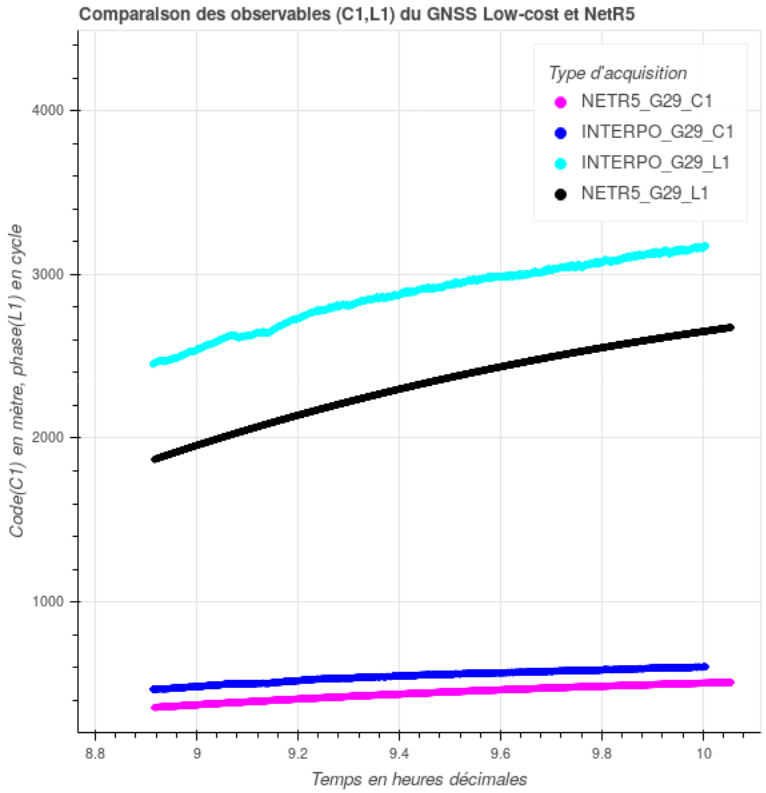

fig.20 Observables L1 et C1 d'un satellite sur NETR5 et sur GNSS-low-cost

On observe sur la figure suivante la différence de décalage d'observables entre le NetR5 et le GNSS low-cost pour une même acquisition sur un même satellite. Si la différence est d'une valeur acceptable au regard de la vitesse de la lumière, il est anormal que le bruit de L1 soit métrique et d'obtenir malgré tout des résultats centimétriques.

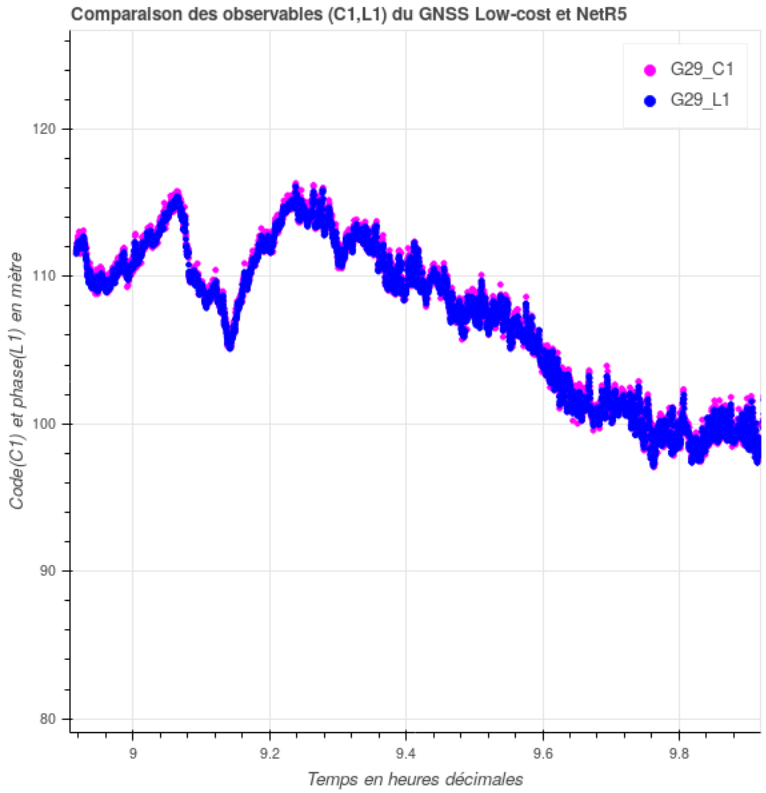

fig.21 Différence de décalage d'un satellite entre NETR5 et GNSS-low-cost

3.7 NTRIP

Nous avons essayé d'utiliser le mode NTRIP des GNSS en nous connectant au serveur de l'IGN à l'aide des identifiants fournis par nos encadrants. Cependant, nous n'avons pas eu le temps d'intégrer notre fonction d'interpolation dans l'application EasyGNSS, ce qui ne permet pas aux calculs en temps réel d'aboutir, et donc de donner une position.

A terme, pour intégrer l'interpolation, il faut récupérer le flux de données arrivant de la puce GNSS au format .ubx, ce qui implique d'apprendre à travailler en binaire, pour interpoler les observables, et ressortir un fichier ubx interpolé qui serait finalement envoyé à l'application EasyGNSS de manière classique (même si l'interpolation se ferait en réalité dans EasyGNSS).

4 Conclusion

En premier lieu, nous avons donc observé que les puces GNSS fournies par u-blox avaient un défaut de décalage d'horloges pouvant être amplifié par la configuration de la prise de mesures.

Puis, nous avons créé une fonction d'interpolation que nous avons intégrée au module **Rinex_o** de la librairie *gnsstoolbox* pour nous permettre de calculer la position de notre GNSS à l'aide du calcul en ligne.

Ensuite, nous avons qualifié la précision de nos observations en mode statique à l'aide du "split" i.e. la comparaison de nos données avec le récepteur Trimble NetR5. Dès lors, il est apparu que nos coordonnées calculées à l'aide de notre récepteur low-cost étaient proches de celle d'un récepteur haut de gamme.

En conclusion, la fabrication d'un GNSS low-cost est aujourd'hui possible pour une utilisation en mode statique. Cependant, l'objectif de notre projet était à la base de s'assurer d'avoir une précision du même ordre de grandeur en mode Cinématique et Statique à l'aide du NTRIP, ce que nous n'avons pu tester faute de temps... Les premiers résultats sont cependant encourageants et laissent présager une utilisation possible de GNSS low cost.

Il reste à intégrer l'interpolation à l'application EasyGNSS et à qualifier la précision des différents modes du GNSS que nous n'avons pas eu le temps de tester : cinématique en RTK ou en NTRIP.

5 Hommages

Nous tenons à remercier Jacques BEILIN pour les conseils et les explications patiemment prodigués durant notre projet. Christian LEPAGE pour sa bonne humeur et son enthousiasme quant à l'avancement du projet ainsi que son aide précieuse pour la fabrication des batteries des récepteurs GNSS 3 et 4.

6 Annexes

6.1 Batterie DIY

Dans le cadre de notre projet, nous avons été amené à modifier des chargeurs pour pouvoir alimenter les récepteurs GNSS de Franklin NGUYEN à partir de batteries. A l'aide de Christian LEPAGE, nous avons donc dégainé le câble d'alimentation et percé un trou dans un chargeur pour y souder les fils, permettant ainsi de charger la batterie et de l'emmener une fois chargée.

6.2 Notice d'utilisation des nouvelles fonctionnalités du module Rinex$_o$

Notice d'utilisation - Interpolation de fichiers RINEX avec gnsstoolbox - Lucas DASSIN - July 2019

July 24, 2019

6.2.1 Introduction

Lors de la récupération de données GNSS à l'aide de puces low-cost ou trop vieilles, l'obtention des observables se fait parfois à des époques en décalage avec les époques rondes utilisées par les stations pour les calculs de position. Cependant, il est possible d'interpoler ces valeurs aux époques désirées.

Le module *rinex_o* de la librairie Python *gnsstoolbox* permet cela à l'aide de fonctions permettant le calcul par interpolation de la valeur des observables aux époques désirées et la réécriture d'un fichier RINEX.

6.2.2 Création d'un objet rinex_o

Tout d'abord, il convient de créer un objet rinex_o à l'aide de la commande suivante:

Rnx = rinex_o()

Il faut ensuite charger un fichier RINEX pour remplir la structure de données de l'objet rinex_o à l'aide de la méthode *load_rinex_o*:

Rnx.load_rinex_o(nom du fichier en entrée)

6.2.3 Interpolation

Nous avons maintenant chargé le fichier dont on veut interpoler les valeurs des observables. Pour procéder à l'interpolation, il nous faut utiliser la méthode *interpolation* qui va recréer un objet rinex_o avec les époques désirées et les observables interpolés associés:

Rnx.interpolation(intervalle, degre)

6.2.4 Ré-écriture du fichier RINEX

Enfin, il ne reste plus qu'à réécrire un nouveau fichier RINEX à l'aide de notre nouvel objet rinex_o à l'aide de la méthode *write_RINEX*:

Rnx.write_RINEX(nom du fichier de sortie)