Université Gustave Eiffel

Institut français des sciences et technologies des transports, de l'aménagement et des réseaux

ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

École Nationale des Sciences Géographiques

PROJET INFORMATIQUE
Rapport d'analyse

Cycle Ingénieur de l'ENSG 3ème année

Intégration de la constellation Glonass dans une application Android de positionnement GNSS : GeolocPVT

Lenhof Edgar

Octobre-Décembre 2019

☒ Non confidentiel ☐ Confidentiel IGN ☐ Confidentiel Industrie ☐ Jusqu'au ...

Ecole Nationale des Sciences Géographiques
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

Table des matières

Glossaire et sigles utiles

API Interface de programmation

Beidou Constellation satellite chinoise

ECEF système de coordonnées Earth-Centered Earth-Fixed

ECI Earth Centered Inertial

ENSG École Nationale des Sciences Géographiques

ESA European Space Agency

Galileo Constellation satellite européenne

GAST Greenwich Apparent Sideral Time

Glonass Système Global de Navigation Satellitaire (Russe)

GMST Greenwich Mean Sideral Time

GNSS Global Navigation Satellite System

GPS Global Positioning System

GSA European GNSS Agency

ICD Interface Control Document

IFSTTAR Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux, devenu Université Gustave Eiffel au 1er janvier 2020

IGS International GNSS Service

package regroupement(s) distinct(s) de classes

PFE Projet de Fin d'Etudes

PPMD Photogrammètrie Positionnement et Mesure de Déformation

PRN Indice du satellite au sein de sa constellation

RK4 Runge-Kutta 4

RTCM Radio Technical Commission for Maritime Services

SPP Single Point Positionning

TAI Temps Atomique International

UML Langage de Modélisation Unifié

UTC Universal Time Coordinated

Introduction

Un système de positionnement par satellites, ou Global Navigation Satellite System (GNSS), est un ensemble de composants reposant sur une constellation de satellites artificiels permettant de fournir à un utilisateur par l'intermédiaire d'un récepteur sa position, sa vitesse et l'heure par multilatération.

Il existe différentes constellations de satellite mondiales ou régionales qui peuvent être utilisées simultanément, comme la constellation américaine Global Positioning System (GPS) ou le Système Global de Navigation Satellitaire (Russe) (Glonass).

Dans le cadre de la troisième année du cycle ingénieur de l'École Nationale des Sciences Géographiques (ENSG) dans la filière Photogrammètrie Positionnement et Mesure de Déformation (PPMD), nous sommes tenus de réaliser un projet informatique. Celui-ci se réalise en deux temps, une étape d'analyse suivie d'une étape de réalisation.

Mon projet a pour objectif l'ajout et l'intégration des observables des satellites du système Glonass au processus de calcul de positionnement GNSS de l'application Android GeolocPVT, développée par le laboratoire GEOLOC de l'Université Gustave Eiffel/Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux, devenu Université Gustave Eiffel au 1er janvier 2020 (IFSTTAR).

Contexte du projet

Présentation du commanditaire

L'IFSTTAR est un établissement public à caractère scientifique et technologique, placé sous la tutelle de l'université Gustave Eiffel.

Au premier janvier 2020, l'IFSTTAR a fusionné avec l'Université Paris-Est Marne-la-Vallée pour créer l'Université Gustave Eiffel. L'ENSG est une école-composante de l'Université Gustave Eiffel.

En particulier, le laboratoire GEOLOC qui a commandé le projet, mène des recherches et expertises sur les méthodes et systèmes de géolocalisation afin d'accompagner l'évolution des pratiques de mobilité et des moyens de transport. Il fait notamment parti du *Raw GNSS Measurement Task Force*, un groupe de recherche crée par l'European GNSS Agency (GSA) dont l'objectif est de tester l'exploitation et l'utilisation des mesures GNSS brutes sous Android [14].

Problématique

En mai 2016, Google a annoncé la disponibilité des données brutes du *chipset* GNSS embarqué dans les nouveaux smartphones à partir d'Android 7. Il devient ainsi possible d'accéder aux mesures de phase et de code, décoder les messages de navigation des satellites à partir des dispositifs Android et développer une application de positionnement transparente.

Qui plus est, avec la mise sur le marché de smartphones intégrant des puces GNSS bi-fréquences (L_1 et L_5), apparait la possibilité d'un positionnement de précision par smartphone.

L'application Android GeolocPVT développée par l'Université Gustave Eiffel/IFSTTAR dans le cadre de précédents stages étudiants [17, 16, 13] permettait jusqu'à présent de :

- acquérir les observables des constellation GPS, Constellation satellite chinoise (Beidou) et Constellation satellite européenne (Galileo) en fréquence L_1 et L_5
- récupérer les *streams* Radio Technical Commission for Maritime Services (RTCM) depuis un serveur de l'International GNSS Service (IGS)
- utiliser les observables et les *streams* de correction afin de faire du positionnement Single Point Positionning (SPP)
- enregistrer les observables, corrections et positions calculées dans la mémoire du smartphone

Enjeux

Le positionnement de précision à partir d'un objet aussi répandu qu'un smartphone permet d'envisager de nombreuses applications. Toutefois les performances restent limitées à trois fac-

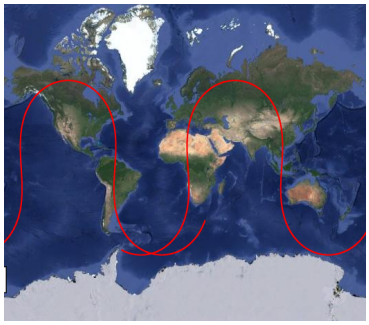
teurs importants : la qualité de l'électronique, les algorithmes de calcul et le nombre de satellites visibles.

Jusqu'à présent, il était possible de déchiffrer les messages de navigation des observables GPS uniquement.

En décryptant les messages de navigation des observables Glonass, on ajoute une redondance d'information pour le calcul des moindres carrés.

De plus, la constellation Glonass permet de se positionner aux hautes latitudes du fait de l'orbite des satellites.

(a) Trace d'un satellite GPS (BIIRM-8) [15]



(b) Trace d'un satellite Glonass (COSMOS-2492) [15]

FIGURE 1 – Traces GPS et Glonass

Objectif et analyse

Objectifs de l'étude

Les principaux objectifs du projet à atteindre consistent à :

— Acquérir les observables Glonass sur l'application
— Acquérir et décoder les messages de navigations Glonass
— Obtenir les orbites et horloges précises Glonass des *streams* RTCM depuis le serveur IGS
— Intégrer la constellation Glonass au calcul SPP
— Rédiger la documentation de l'application (READ ME, documentation développeur et utilisateur)

Analyse de l'étude

Durant mon analyse, j'ai cerné les différentes étapes et méthodes d'implémentation à réaliser pour atteindre les objectifs fixés. Les diagrammes de classes de l'application initiale sont disponibles en annexe A.

Refonte des classes d'éphémérides

Lorsqu'au début du projet, je découvrais la structure de l'application, les classes d'éphémérides étaient séparées en deux regroupement(s) distinct(s) de classes (package)s Java distincts. Celles récupérées par *streams* RTCM dans le package gogps et celle permettant de récupérer les éphémérides GPS par message de navigation dans le package geoloclib.

Les éphémérides des constellations préalablement implémentées sont toutes de type képlérienne. Il n'était pas possible d'intégrer proprement les éphémérides Glonass, non képlérienne, en l'état.

C'est pourquoi j'ai proposé de refondre l'architecture autour des classes d'éphémérides en exploitant les principes d'héritage et de polymorphisme dont dispose le langage Java.

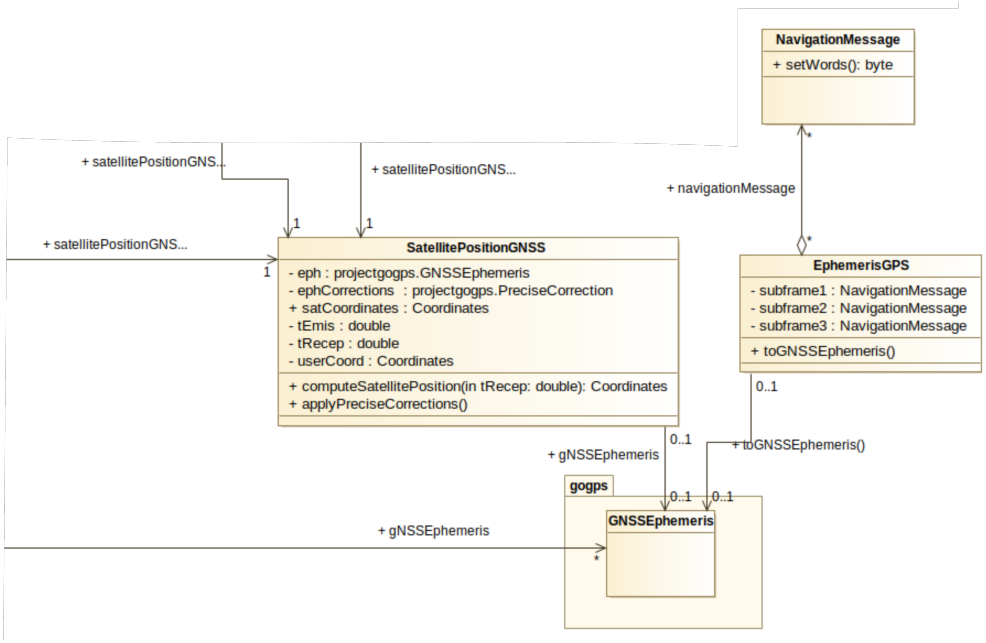


FIGURE 2 – Diagramme de classe simplifié des objets d'éphémérides

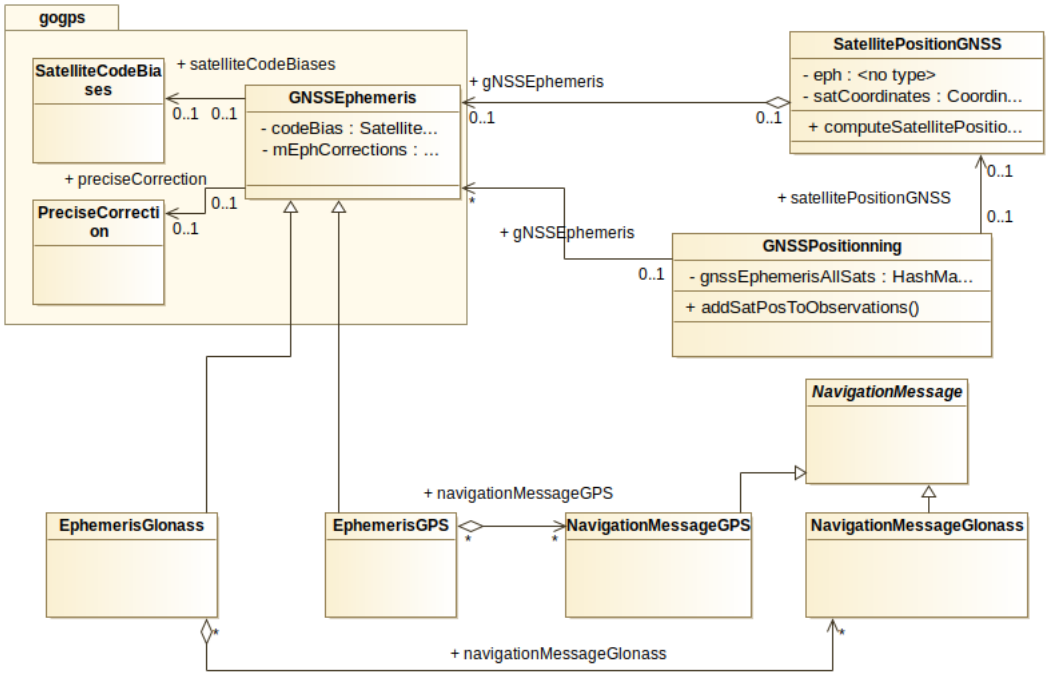


FIGURE 3 – Diagramme de classe simplifié et restructuré, prévu pour le calcul des coordonnées des satellites

Récupération des données d'éphémérides par message de navigation

La solution la plus simple sembla être de suivre le même procédé que celui déjà en place avec le GPS pour récupérer le message de navigation Glonass. C'est-à-dire définir une classe GlonassNavigationMessage chargée de décoder les messages et une classe GlonassEphemeris dans le package gogps.ephemeris, en suivant la structure de la refonte, chargée de contenir les données d'éphémérides obtenues.

Le décodage du message binaire se fera à l'aide de l'Interface Control Document (ICD) Glonass [11, 18].

Récupération des données d'éphémérides par streams RTCM

Pour les besoins de l'application nous devons décoder le message 1020 contenant les données d'éphémérides et d'horloges, le message 1065 contenant les biais de code et le message 1066 contenant les corrections d'éphémérides[2, 10].

Ici aussi, la solution la plus simple sembla être de suivre le modèle de récupération des éphémérides déjà existant avec une classe DecodeMsgXXXX par message (dans le package gogps.ephemeris), chargée de de décoder les *streams* RTCM. Il suffit ensuite de relier cette classe à la classe RTCM3Client recevant tous les *streams* et aux classes d'éphémérides et de corrections correspondantes.

Le décodage des messages RTCM pouvait être réalisé à l'aide du code en C open-source de RTKLIB qui réalise déjà ce procédé [22].

Calcul de position des satellites

La méthode d'intégration de la position d'un satellite Glonass par Runge-Kutta 4 (RK4) étant différente de celle d'un satellite képlérien comme GPS ou Galileo, il faut la coder *ex-nihilo*.

Pour conserver la méthode d'intégration des autres constellations, qui est déjà fonctionnelle, il est possible de définir un nouveau constructeur de la classe SatellitePositionGNSS du package geoloclib.satellites prenant en argument un objet de type GlonassEphemeris. Ce nouveau constructeur ferait appel à la méthode d'intégration RK4 afin de calculer la position du satellite.

Intégration au calcul de moindres carrés pour le positionnement récepteur

Une fois les pseudo-distances et éphémérides obtenues et le calcul de position des satellites mis en place, il faut à intégrer la constellation Glonass dans le calcul de position récépteur par multilatération.

Pour cela, il faut modifier la classe GNSSPositionning du package geoloclib.computations pour prendre en compte la constellation et si besoin ajouter un paramètre de biais entre le temps GPS et le temps Glonass dans le calcul de moindres carrés.

Planning prévisionnel

Semaine	Description
16/12	Modifications de l'interface et des fonctions du package app, vérifier si les satellites Glonass apparaissent comme visibles
06/01	Modifier la structure autour de la classe GNSSEphemeris et vérifier que l'application conserve toutes les fonctionnalités existantes
13/01	
20/01	Implémenter la récupération et le décodage des messages de navigation, le calcul de position satellite, vérifier le résultat des positions obtenues
27/01	
03/02	Implémenter la récupération et le décodage des stream RTCM, vérifier les éphémérides et corrections
10/02	
17/02	Vérification globale du fonctionnement de l'application, comparaison des résultats obtenus avec et sans Glonass, rédaction du rapport
24/02	Rédaction du rapport et préparation de la soutenance

TABLE 1 – Planning prévisionnel des développements

Etude technique

Structure des messages de navigation Glonass

Les messages de navigation Glonass sont continuellement transmis sur la fréquence L_1 par un *superframe* de 2,5 minutes. Il est composé de cinq *frames* de 30 secondes, eux même constitués de 15 *strings* de 2 secondes avec un équivalent de 100 bits d'information par *string*.[11, 18, 19, 21]

Frame Number	String Number	2s	1.7 s	0.3s			
I	1	0	Inmediate data	Kx	MB	30s	30s x 5 =2.5 min
	2	0	for	Kx	MB		
	3	0	transmitting	Kx	MB		
	4	0	satellite	Kx	MB		
	5	0	Non inmmediate data	0	MB		
	:	:	(almanac) for	:	MB		
	15	0	five satellites	Kx	MB		
II	1	0	Inmediate data	Kx	MB		
	2	0	for	Kx	MB		
	3	0	transmitting	Kx	MB		
	4	0	satellite	Kx	MB		
	5	0	Non inmmediate data	0	MB		
	:	:	(almanac) for	:	MB		
	15	0	five satellites	Kx	MB		
III	1	0	Inmediate data	Kx	MB		
	2	0	for	Kx	MB		
	3	0	transmitting	Kx	MB		
	4	0	satellite	Kx	MB		
	5	0	Non inmmediate data	0	MB		
	:	:	(almanac) for	:	MB		
	15	0	five satellites	Kx	MB		
IV	1	0	Inmediate data	Kx	MB		
	2	0	for	Kx	MB		
	3	0	transmitting	Kx	MB		
	4	0	satellite	Kx	MB		
	5	0	Non inmmediate data	0	MB		
	:	:	(almanac) for	:	MB		
	15	0	five satellites	Kx	MB		
V	1	0	Inmediate data	Kx	MB		
	2	0	for	Kx	MB		
	3	0	transmitting	Kx	MB		
	4	0	satellite	Kx	MB		
	5	0	Non inmmediate data	0	MB		
	:		(almanac) for four sat	:			
	14	0	Reserved Data	Kx	MB		
	15	0	Reserved Data	Kx	MB		

FIGURE 4 – Structure des messages de navigation Glonass [11, 18]

Chaque *string* se compose d'un message à proprement parler de 85 bits par index décroissant (85 84 83 ... 3 2 1) puis d'une séquence pseudo-aléatoire de 0,3 secondes appelée *Time mark* qui ne présente pas d'intérêt pour notre besoin. Les huit premiers bits désignent le code de Hamming permettant de vérifier la validité du message grâce à un algorithme spécifié dans la documentation Glonass [11, 18].

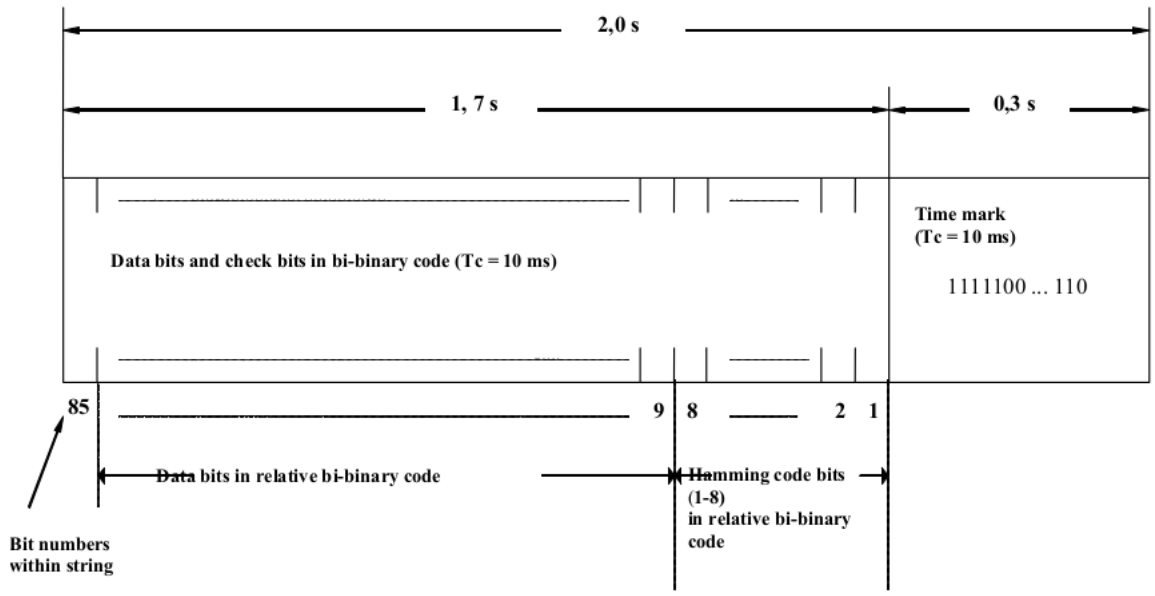


FIGURE 5 – Structure des "strings" des messages de navigation Glonass [11, 18]

Les 4 premières *strings* de chaque *frames* sont appelées *immediate data* et contiennent les données d'éphéméride, d'horloge, d'état du satellite, ect... sous forme de *words* de longueur variable dont le bit de poids fort est placé à l'avant (MSB first). Pour certains d'entre eux, il est nécessaire d'appliquer un facteur d'échelle à la valeur brute transmise par le message binaire.

Il faut donc récupérer certains de ces *words* pour reconstruire les éphémérides des satellites.

Récupération des observables Glonass

La puce GNSS d'un téléphone est un capteur passif qui transmet tous les signaux qu'elle recoit à l'Interface de programmation (API) Android. Il est alors relativement aisé d'obtenir les informations nécessaires sous la forme d'objet de type *GnssNavigationMessage* [7] et *GnssMeasurement* [6] grâce aux *Events* et *CallBacks* de l'API.

Android GnssMeasurement

Chaque objet de type *GnssMeasurement* implémente une mesure d'un seul satellite. Il permet d'accéder à la valeur de pseudo-distance, au temps de réception de la mesure par la puce ainsi que d'autres informations complémentaires.

Pour Glonass, seule la fréquence L_1 est disponible.

Android GnssNavigationMessage

Chaque objet de type *GnssNavigationMessage* implémente un sous-message d'un message de navigation d'un satellite. Il permet d'accéder aux informations du sous-message sous la forme d'une liste de bytes de longueur variable selon la constellation au format Gros Boutiste (poids des octets par ordre décroissant) et précise l'identifiant du message et du sous-message en question.

Dans le cas de Glonass, il est d'une taille de 11 octets soit 88 bits. Cela correspond à une *string* sans le *Time mark* auquel a été ajouté 3 bits à la fin de la chaine binaire.

Calcul des coordonnées des satellites Glonass

Contrairement aux autres constellations, les paramètres d'éphémérides des satellites Glonass, exprimés dans le repère système de coordonnées Earth-Centered Earth-Fixed (ECEF) PZ-90, ne sont pas définis selon le modèle képlerien. [21, 20, 12]

Parameter	Explanation
t_e	Ephemerides reference epoch
$x(t_e)$	Coordinate at t_e in PZ-90
$y(t_e)$	Coordinate at t_e in PZ-90
$z(t_e)$	Coordinate at t_e in PZ-90
$v_x(t_e)$	Velocity component at t_e in PZ-90
$v_y(t_e)$	Velocity component at t_e in PZ-90
$v_z(t_e)$	Velocity component at t_e in PZ-90
$X''(t_e)$	Moon and sun acceleration at t_e
$Y''(t_e)$	Moon and sun acceleration at t_e
$Z''(t_e)$	Moon and sun acceleration at t_e
$\tau_n(t_e)$	SV clock offset
$\gamma_n(t_e)$	SV relative frequency offset

FIGURE 6 – Paramètres nécessaires au calcul de positionnement d'un satellite Glonass [20]

Ces paramètres sont valables pour un instant d'émission t_e et sont mises à jour toutes les 2 à 3 heures. Quelques étapes d'intégration sont nécessaires afin de déterminer la position des satellites à un instant $t = t_e + dt$:

1. **Transformation des coordonnées vers un repère inertiel**

   - Temps Universal Time Coordinated (UTC)0 $t_0 = t_e - 3\,heures$

- Position :

$$\begin{aligned} x_a(t_0) &= x(t_0)\cos(\theta_{Ge}) - y(t_0)\sin(\theta_{Ge}) \\ y_a(t_0) &= x(t_0)\sin(\theta_{Ge}) + y(t_0)\cos(\theta_{Ge}) \\ z_a(t_0) &= z(t_0) \end{aligned}$$

- Vitesse :

$$\begin{aligned} \dot{x_a}(t_0) &= \dot{x}(t_0)\cos(\theta_{Ge}) - \dot{y}(t_0)\sin(\theta_{Ge}) - \omega_E * y_a(t_0) \\ \dot{y_a}(t_0) &= \dot{x}(t_0)\sin(\theta_{Ge}) + \dot{y}(t_0)\cos(\theta_{Ge}) + \omega_E * x_a(t_0) \\ \dot{z_a}(t_0) &= \dot{z}(t_0) \end{aligned}$$

- Accélération luni-solaire :

$$\begin{aligned} (Jx_am + Jx_as) &= \ddot{x}(t_0)\cos(\theta_{Ge}) - \ddot{y}(t_0)\sin(\theta_{Ge}) \\ (Jy_am + Jy_as) &= \ddot{x}(t_0)\sin(\theta_{Ge}) + \ddot{y}(t_0)\cos(\theta_{Ge}) \\ (Jz_am + Jz_as) &= \ddot{z}(t_0) \end{aligned}$$

Avec
$\theta_{Ge} = \theta_{G0} + \omega_E * (t_0 - 3heures)$ le Greenwich Apparent Sideral Time (GAST)
$\omega_E = 0.7292115 * 10^{-4} rad/s$ la vitesse de rotation de la terre
θ_{G0} le temps sidéral à l'époque t_0

2. **Système d'équations différentielles décrivant le mouvement des satellites**

$$\begin{cases} \frac{dx_a}{dt} = \dot{x_a}(t) \\ \frac{dy_a}{dt} = \dot{y_a}(t) \\ \frac{dz_a}{dt} = \dot{z_a}(t) \\ \frac{d\dot{x_a}}{dt} = -\bar{\mu}\bar{x_a} + \frac{3}{2}C_{20}\bar{\mu}\bar{x_a}\rho^2(1 - 5\bar{z_a}^2) + Jx_am + Jx_as \\ \frac{d\dot{y_a}}{dt} = -\bar{\mu}\bar{y_a} + \frac{3}{2}C_{20}\bar{\mu}\bar{y_a}\rho^2(1 - 5\bar{z_a}^2) + Jy_am + Jy_as \\ \frac{d\dot{z_a}}{dt} = -\bar{\mu}\bar{z_a} + \frac{3}{2}C_{20}\bar{\mu}\bar{z_a}\rho^2(3 - 5\bar{z_a}^2) + Jz_am + Jz_as \end{cases}$$

Avec
$\bar{\mu} = \frac{\mu}{r^2}$, $\bar{x_a} = \frac{x_a}{r}$, $\bar{y_a} = \frac{y_a}{r}$, $\bar{z_a} = \frac{z_a}{r}$
$\bar{\rho} = \frac{a_E}{r}$, $r = \sqrt{x_a^2 + y_a^2 + z_a^2}$
$a_E = 6378.136km$ le rayon de la terre à l'Equateur dans le système PZ-90
$\mu = 398600.44km^3/s^2$ constante gravitationnelle dans le système PZ-90
$C_{20} = -1082.63 * 10^{-6}$ second coefficient zonal de l'expression en harmonique sphérique

3. **Intégration de l'algorithme de Runge-Kutta**

Soit un système de la forme :

$$\begin{cases} \frac{dy_1}{dt} &= f_1(t, y_1, \dots, y_n) \\ & . \\ & . \\ & . \\ \frac{dy_n}{dt} &= f_n(t, y_1, \dots, y_n) \end{cases} \iff Y'(t) = F(t, Y(t))$$

L'algorithme de RK4 permet de résoudre ce système d'équation en intégrant des intervalles de pas à une solution initiale tel que :

$$\begin{aligned} K_1 &= F(t_n, Y_n) \\ K_2 &= F(t_n + h/2, Y_n + hK_1/2) \\ K_3 &= F(t_n + h/2, Y_n + hK_2/2) \\ K_4 &= F(t_n + h, Y_n + hK_3) \\ Y_{n+1} &= Y_n + h/6(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned}$$

Avec $h = t_{k+1} - t_k$
Et en prenant $Y_0 = Y(t_0)$ et $Y'_0 = Y'(t_0)$ comme solutions initiales.

4. **Transformation des coordonnées vers le repère PZ-90**

$$\begin{aligned} x(t) &= x_a(t)\cos(\theta_{Ge}) + y_a(t)\sin(\theta_{Ge}) \\ y(t) &= -x_a(t)\sin(\theta_{Ge}) + y_a(t)\cos(\theta_{Ge}) \\ z(t) &= z_a(t) \end{aligned}$$

5. **Transformation des coordonnées en WGS84 (=ITRF2000)**

$$\begin{aligned} x(t)_{ITRF2000} &= x(t)_{PZ90.02} - 0.36 \\ y(t)_{ITRF2000} &= y(t)_{PZ90.02} + 0.08 \\ z(t)_{ITRF2000} &= z(t)_{PZ90.02} + 0.18 \end{aligned}$$

Les coordonnées obtenues peuvent être directement utilisées pour du positionnement conjointement avec les autres constellations.

Travail réalisé

Refonte des classes d'éphémérides

Comme prévu lors de mon analyse, j'ai réalisé une refonte des classes d'éphémérides de l'application qui jusqu'à présent considérait une classe dédiée au remplissage par message de navigation et une autre par *streams* RTCM.

Cette étape s'est déroulée bien plus facilement que ce dont à quoi je m'attendais. Ayant déjà eu à refondre du code lors d'un précédent projet je savais que cela pouvait rapidement devenir problématique.

La nouvelle structure privilégie d'avantage le polymorphisme et je n'ai noté auncun dysfonctionnement du système existant. Toutes les classes d'éphémérides sont regroupées dans le package gogps.ephemeris.

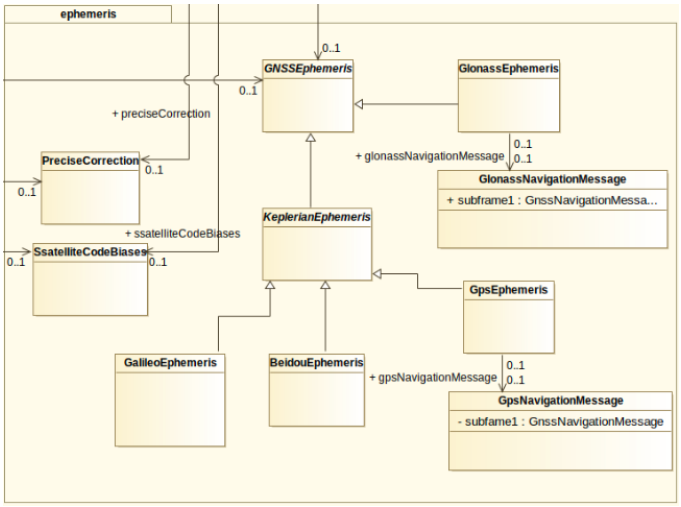


FIGURE 7 – Nouvelle structure du package gogps.ephemeris

Récupération et décodage des messages de navigation

Le décodage des messages de navigation s'avéra cependant beaucoup plus complexe que prévu durant l'analyse et j'ai largement dépassé le délai de 2 semaines que je m'étais fixé.

Le message Glonass comprime au maximum l'information et sa restitution est loin d'être immédiate comme avec le message GPS. Il nous aura fallu un certain temps pour comprendre, avec les membres de l'équipe GEOLOC, comment procéder.

Pour récupérer les données d'éphémérides qui nous intéressent il faut extraire certains morceaux du message binaire (appelés *words*) et reconstruire la valeur correspondante selon un certain formalisme décrit dans l'ICD.

Les différents points qu'il a fallu cerner et gérer afin de réaliser cet objectif sont listés ci-dessous : :

— l'API Android transmet le message de navigation sous la forme de 11 octets à convertir en chaîne de caractères binaires de taille 88 ;
— les trois dernières valeurs de cette chaine sont des bits ajoutés par l'API qu'il ne faut pas prendre compte, la chaîne binaire du message est donc de taille 85 ;
— les bits du message de navigation sont numérotés de façon décroissante dans l'ICD ;
— les sous-chaînes binaires constituant les *words* suivent le format MSB first (le bit de poids fort est en avant) ;
— certains *words* associent leur bit de poids fort au signe de la valeur des bits suivants (0 positif et 1 négatif) tandis que d'autres non ;
— certains *words* nécessitent d'appliquer un facteur d'échelle à la valeur restituée par la chaîne binaire pour obtenir la valeur non compressée.

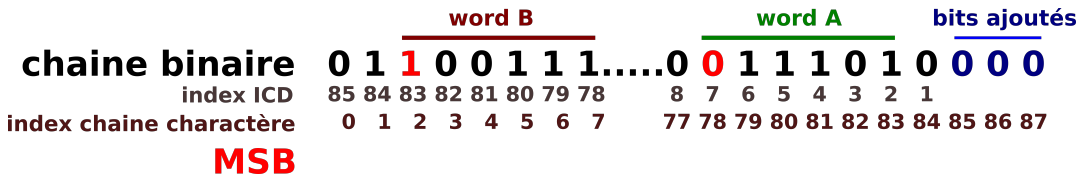


FIGURE 8 – Représentation de la chaîne binaire du message de navigation

Word	A	B
startbit (index ICD)	2	78
endbit (index ICD)	7	83
signe dans MSB	non	oui
facteur d'échelle	2^5	non
Valeur obtenue	$2^0 + 2^2 + 2^3 + 2^4 + 2^5$ $*2^5$ $= 1952$	$2^0 + 2^1 + 2^2$ $*(-1)$ $= -7$

TABLE 2 – Exemples de décodage de *words* du message de navigation

Seules les données des quatre premières *strings* sont nécessaires au calcul de position du satellite. Toutefois, pour reconstruire la date de validité de l'éphéméride, il est nécessaire d'accéder au *word* N_4 de la *string* 5 et d'implémenter un algorithme détaillé en annexe D. L'application permet d'obtenir cette date au format grégorien et de calculer la semaine GPS correspondante.

Positionnement des satellite à partir des données d'éphémérides

Une fois les données d'éphémérides acquises, l'étape de positionnement des satellites par méthode RK4 a pu être abordée.

La première étape consiste à calculer la valeur du GAST afin d'effectuer la transformation du repère ECEF au repère Earth Centered Inertial (ECI). Pour cela, il a fallu implémenter un outil de conversion de date grégorienne en jours juliens [1].

Le calcul du GAST a été réalisé comme expliqué dans l'annexe C mais une version simplifiée des équations d'équinoxe issue de la GNSSTOOLBOX a été mise en place.

D'une part, le vrai calcul de ces équations nécessite une implémentation très fastidieuse en introduisant l'ensemble des constantes du modèle de nutation **IAU1980** [9]. D'autre part, le recalcul toutes les secondes pour chaque satellite observé peut engendrer un coût de calcul considérable pour le téléphone.

La méthode RK4 a été implémentée comme décrite dans l'étude technique.

Récupération et décodage des messages RTCM

Les messages désignés dans l'étude technique ont été décodés. Une nouvelle classe a été attribuée par nouveau message et intégrée à l'application en suivant l'exemple des messages déjà décodés et le code-source de RTKLib.

Cependant, il est à noter que certains messages gèrent les mêmes types de données pour les différentes constellations (1059 et 1065 pour les biais de code GPS et Glonass par exemple) et sont structurés de la même façon, à la distinction des premières données que l'on pourrait associer à un *header*.

Rédaction de la documentation

Deux semaines avant la date de rendu et ayant accumulé un retard considérable durant le développement, nous dressâmes avec les commanditaires de l'équipe GEOLOC la liste des points à prioriser.

La rédaction du présent rapport ainsi que du guide développeur furent privilégiés au détriment de fonctionnalités annexes et de la rédaction d'un manuel utilisateur afin de pouvoir reprendre et terminer le travail réalisé plus facilement.

La rédaction du guide développeur est par ailleurs un travail important puisqu'il n'en existait aucun jusqu'à présent. Les diagrammes Langage de Modélisation Unifié (UML) fournis sont tous réalisés avec la solution libre Modelio 4.0 et mis à disposition sur le GitLab afin de pouvoir être repris par les contributeurs suivants.

Essais et résultats

Test des récupération d'éphémérides

Récupération par message de navigation

N'ayant pu trouvé aucune solution libre implémentant le décodage des messages de navigation sur lequel me baser et vérifier mon travail, il aura fallu intuiter une procédure de contrôle.

Tout d'abord, on peut simplement vérifier que l'ordre de grandeur de la valeur d'un *word* correspond bien à celui attendu. Cela me permit de me rendre compte qu'un facteur d'échelle devait être appliqué mais cela n'est pas suffisant.

Chaque *string* du message explicite son numéro aux bits 81 à 84. Il est donc possible de comparer la valeur décodée avec le numéro fournit par l'API (attribut *SubmessageId*) et ainsi s'assurer du bon décodage du message.

Récupération par streams RTCM

Afin de vérifier les données récupérées par *streams* il est possible d'utiliser un logiciel nommé Snip qui propose un décodeur de message RTCM. Cela avait d'ailleurs été le cas lors des précedents travaux sur l'application. Cependant, parmi les messages qui nous intéresse, seul le 1020 est disponible dans la version gratuite.

Finalement je n'aurais pas utiliser ce logiciel d'autant plus que les éphémérides obtenues correspondaient à celle décodées par message de navigation, me confortant dans l'idée que les deux différentes solutions étaient corrects.

Toutefois, il pourrait s'avérer intéressant de vérifier la véracité des corrections obtenues par les messages 1065 et 1066. Elles n'ont pas été vérifiées par manque de temps mais elles semblent cohérentes à celles obtenues pour les autres constellations.

Exemple de données d'éphéméride récupérées

Voici un exemple d'éphéméride Glonass. Il s'agit du Indice du satellite au sein de sa constellation (PRN) 18 enregistré le 10/02/2020 en fin d'après midi.

Attribut	Valeur	Unité
Constellation	3	
prn	18	
Bn (health flag)	0	
Tb (time of day of validity)	1245	min
Tk (time of day of message)	73800	s
X (PZ90)	24635.6748046875	km
Y (PZ90)	-4060.5107421875	km
Z (PZ90)	5093.74951171875	km
Vx	-0.7302684783935547	km.s-1
Vy	-0.040630340576171875	km.s-1
Vz	3.5195655822753906	km.s-1
AccX	-9.313225746154785E-10	km.s-2
AccY	-2.7939677238464355E-9	km.s-2
AccZ	-1.862645149230957E-9	km.s-2
tauN	-2.4645589292049408E-5	s
gammaN	9.09494701772928E-13	
Nt	41	jours
N4	7	années bissextiles
Wn	2092	semaine GPS

Table 3 – Ephéméride Glonass enregistrée par GeolocPVT

Test des résultats de positionnement des satellites

Pour m'assurer de la qualité du code de positionnement des satellites à partir des données d'éphémérides, j'ai comparé mes résultats avec ceux de la PYGNSSTOOLBOX développée par J. Beilin, professeur à l'ENSG.

Comparaison du résultat de position

La comparaison des résultats de la position du satellite après intégration RK4 sur notre éphéméride test avec $t_{emission} = t_b + 5min$ est visible en table 4 (page 24).

Ephémérides 05/02/20 PRN 26		
Solution	Gnsstoolbox	GeolocPVT
X (m)	5208588.818	5208588.834
Y (m)	-23949519.263	-23949519.260
Z (m)	7040669.980	7040669.956
dX (m)	0.016	
dY (m)	0.003	
dZ (m)	0.023	
Ephémérides 10/02/20 PRN 18		
X (m)	24395704.237	24395704.108
Y (m)	-4064169.372	-4064169.460
Z (m)	6143715.769	6143715.794
dX (m)	0.129	
dY (m)	0.088	
dZ (m)	0.025	

TABLE 4 – Comparaison des résultats de position (ITRF2000) sur une intégration RK4 de 5 minutes

Différence d'implémentation du calcul du GMST

En parcourant les deux différents code, il est à noter que le calcul du Greenwich Mean Sideral Time (GMST) que j'ai implémenté suit les recommandations du Navipédia de l'European Space Agency (ESA) [3, 4] et est différent de l'implémentation simplifiée de la PYGNSSTOOLBOX présentée ci-dessous :

```
1 GMST = 18.697374558 + 24.06570982441908 * (te.jd - 2451545.0)
  #in sideral hours
```

Après conversion de la valeur (obtenue en secondes sidérales) en radians et comparaison avec celle de la *toolbox*, je trouve un écart angulaire pouvant atteindre l'ordre de $10e-7$ entre nos deux solutions.

Bien que cet écart puisse sembler faible, multiplié par des valeurs de coordonnées pouvant atteindre $2,7e7$ m, lors du passage au repère inertiel, l' écart sur les coordonnées ECI peut être de l'ordre de 0,1-1 m.

J'ai cherché à comparer ces résultats sur différents calculateurs en ligne mais aucun ne semble s'accorder sur une meilleure exactitude :

— $GMST_{geoloc} = 2.4355671034810484rad$
— $GMST_{pygnsstoolbox} = 2.4355668296107793rad$

— $GMST_{gavo} = 2,435566901 rad$ [5]
— $GMST_{neoprogrammics} = 2,435567119 rad$ [8]

Au moment de l'analyse je n'avais pas approfondi la manière de déterminer le GAST. Or ce point s'avéra non trivial par la suite. Des informations complémentaires sont disponibles en annexe C.

Comparaison du résultat de position en prenant en compte la différence de GMST

J'ai donc repris le test précédent en explicitant en dur les valeurs de GAST calculées par GeolocPVT lors des transformation repère fixe/repère inertiel à la *toolbox* (voir table 5, page 25).

Ephémérides 05/02/20 PRN 26		
Solution	Gnsstoolbox	GeolocPVT
X (m)	5208588.847	5208588.834
Y (m)	-23949519.257	-23949519.260
Z (m)	7040669.980	7040669.956
dX (m)	0.013	
dY (m)	0.017	
dZ (m)	0.016	
Ephémérides 10/02/20 PRN 18		
X (m)	24395704.241	24395704.108
Y (m)	-4064169.345	-4064169.460
Z (m)	6143715.769	6143715.794
dX (m)	0.133	
dY (m)	0.115	
dZ (m)	0.025	

Table 5 – Comparaison des résultats de position (ITRF2000) sur une intégration RK4 de 5 minutes en explicitant les valeurs de GAST

Les écarts entre les deux solutions sont toujours très importants et plutôt comparables entre le cas où la valeur du GAST a été explicitée et le cas où cela n'a pas été fait. Il existe donc une autre distinction entre les deux solutions qui n'a pas été identifiée.

Cet écart déjà relativement important sur une durée d'intégration courte empire pour des durées plus longues jusqu'à atteindre des valeurs vraiment problématiques tel que le montre la figure 9 (page 26).

Ainsi sur une intégration longue on peut atteindre la centaine de mètres. Il y a définitivement un biais au sein de la méthode de RK4 de l'application. J'ai vérifié le processus de calcul et les constantes en jeu mais je ne suis pas parvenu à localiser la source d'erreur.

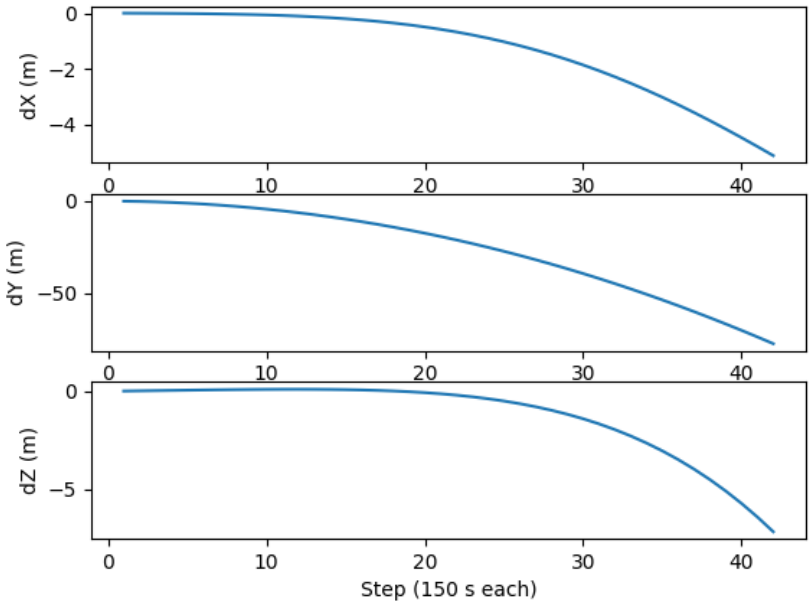


FIGURE 9 – Ecarts cumulés entre Geoloclib et PyGnssToolbox au cours d'une intégration RK4 de 105 minutes

Test global du positionnement utilisateur avec Glonass

Une fois toutes les différentes parties implémentées et testées séparément, il a été possible de les relier entre elles et de les greffer à l'algorithme de moindres carrés en charge du positionnement récepteur. Malheureusement, les résultats produits se révélèrent aberrants.

Le problème pourrait tout d'abord venir d'une erreur de calcul des positions des satellites puisque comme vu précédemment, les résultats de l'application ne concordent pas avec ceux de la GNSSTOOLBOX et sont donc possiblement erronés.

Il est aussi possible que la conversion d'une unité à une autre spécifique pour un certain attribut soit nécessaire lors de l'intégration de la constellation Glonass au calcul de multilatération.

La source d'erreur n'a cependant pas pu être trouvée à cause du retard accumulé lors des précedentes étapes.

Bilan

Gestion de projet

Date	Description
lundi 23/12/19	ajout et connexion des items UI pour glonass
mercredi 25/12/19	modification des fonctions de réceptions des messages de navigation
dimanche 29/12/19	début refactoring des classes éphémérides
lundi 30/12/19	refactoring fini et testé
jeudi 16/01/20	début implémentation classe ephemeride et navmsg de glonass
vendredi 17/01/20	poursuite implémentation classes éphémérides et messgae de navigation glonass
lundi 20/01/20	calcul du thetaG0 et outils conversion en jours juliens, début implémentation RK4
mercredi 22/01/20	Entretien du dépôt Git et mise en place de bonnes pratiques
vendredi 24/01/20	calcul date d'émission et GPS week number
mardi 28/01/20	facteur d'échelle des données du message de navigation
mercredi 29/01/20	découverte message de navigation à l'envers et code de Huffman
samedi 01/02/20	cherche ereur de conversion du message de navigation
lundi 03/02/20	décodage message RTCM 1020
mercredi 05/02/20	début décodage message 1065 et 1066, cherche ereur de conversion du message de navigation
vendredi 07/02/20	éphémérides par message de navigation réussi
samedi 08/02/20	implémentation position satellite presque finie, réalise différence de GAST
lundi 10/02/20	message de navigation et RTCM en accord sur les éphémérides
mardi 10/02/20	tests et modifications calcul GAST
mercredi 12/02/20	test global fonctionnement application, cherche erreur calcul position récepteur
jeudi 13/02/20	cherche erreur positionnement satellite, rédaction du rapport
vendredi 14/02/20	cherche erreur positionnement satellite, rédaction du rapport
dimanche 16/02/20	rédaction du rapport
lundi 17/02/20	rédaction du rapport et de la documentation développeur
mardi 18/02/20	rédaction du rapport et de la documentation développeur
mercredi 19/02/20	rédaction de la documentation développeur

Table 6 – Journal de bord du projet

Les étapes primordiales à réaliser et attendues par le commanditaire étaient les suivantes :
- Obtenir les épéhémérides par message de navigation
- Calculer les positions des satellites

— Ajouter la nouvelle constellation au calcul de postion récepteur
— Obtenir les éphémérides par message RTCM
— Rédiger la documentation développeur
— Rédiger la documentation utilisateur

Comme le travail sur les messages de navigation restait infructueux pendant un long moment, j'ai souhaité basculer sur le travail des messages RTCM afin de m'assurer d'obtenir des données d'éphémérides réelles sur lesquelles tester ma solution de position des satellites. C'est pour cette raison que cette fonctionnalité secondaire a été implémentée tandis que d'autres plus importantes n'ont pas été achevées.

Difficultés rencontrées

L'un des principaux problèmes rencontrés durant le projet fut pour moi la difficulté de trouver les bonnes informations dans la documentation Glonass. La logique du message de navigation est complexe puisque, contrairement à GPS, l'information est compressée au maximum. Elle n'est reconstructible qu'en utilisant des techniques et algorithmes peu intuitifs. Même si toutes les informations nécessaires sont disponibles dans l'ICD (seule source d'information facilement disponible en langue anglaise), il s'agit d'un document très long et très technique. Je pense que mon analyse sur cette partie du projet n'a pas été suffisament poussée et j'ai perdu un temps important durant le développement à comprendre comment la réalise.

Le second problème rencontré est la complexité de certaines notions, notament lors du changement de repère ECEF vers ECI qui me prirent aussi beaucoup de temps à assimiler. Ici aussi, une analyse plus poussée aurait grandement facilité le travail de développement.

Enfin, jusqu'à présent l'application malgré son envergure ne disposait d'aucune documentation sur laquelle s'appuyer ce qui nécessita un temps non négligeable durant l'analyse et la rédaction du guide développeur.

Perspectives d'amélioration

Validation du positionnement satellite

Il est impératif de trouver la source d'erreur du positionnement satellite de l'application. La divergence d'arrondi numérique entre les deux solutions comparées ne peut être uniquement justifié par des erreurs d'arrondis. Une légère erreur de quelques centimètres est probablement présente en amont du processus d'intégration numérique et se retrouve peu à peu accentuée au cours des itérations.

Intégrer Glonass au reste du calcul de postion récepteur

Une fois le positionnement satellite corrigé, il faudrait retester le positionnement récepteur avec la constellation Glonass. Il se peut que ce problème soit alors réglé. Si toutefois les résultats obtenus restent éronnés, cela signifie qu'une étape a été oubliée dans l'intégration au calcul de multilatération. Il est en effet possible qu'il faille convertir l'une des valeurs calculées d'une unité à une autre et que cela m'aura échappé.

Validation du message de navigation avec le code de Hamming

Chaque message de navigation Glonass comporte 8 bits dénommé "code de Hamming" permettant de vérifié l'intégrité du message et de corriger une potentiel erreur de transmission.

C'est un algorithme assez complexe présenté dans l'ICD [11] qui pourrait être implémenté pour assurer un contrôle.

Il est à noter cependant que la classe GnssNavigationMessage de l'API Android le réalise peut-être déjà puisqu'elle présente les attributs STATUS_ PARITY_ PASSED et STATUS_ PARITY_ REBUILT qui semble correspondre à ce que pourrait fournir ce code de Hamming [7].

Refonte des classes de décodage des messages RTCM

Les messages RTCM relayant le même type d'information pour des constellations différentes suivent le même schéma de transmission à la différence des premières données du message que l'on pourrait considérer comme *header* du message.

Ainsi, une seule classe par type de message (code de biais, correction d'éphémérides) suffirait avec une distinction pour le décodage du *header* plutôt que la structure actuelle qui définit une classe par message RTCM dédoublant certaines parties du code inutilement.

Je conseillerai donc de suivre l'exemple de la structure de RTKLib qui présente une telle optimisation (voir le code de rtcm3.c) [22].

Décoder le message de navigation Galileo

L'intérêt d'utiliser les messages de navigation et de ne pas avoir besoin de passer par un serveur de *streams* RTCM qui nécessite une connexion internet. De plus, ces serveurs ne sont le plus souvent accessibles qu'avec un identifiant et un mot de passe Décoder le messsage de navigation d'une constellation supplémentaire permettrait d'augmenter la redondance du positionnement par éphémérides directement transmises.

Tri des fonctionnalités gogps

Le *package* gogps en charge de la gestion des flux RTCM a été importé dans sa totalité dans le code-source de l'application. Il s'agit d'un projet assez important comportant un grand nombre de classe.

Ce code source n'est absolument pas documenté et il possible que certaines fonctionnalités tournent inutilement en fond de tâche consommant une partie de la mémoire. Je pense qu'il pourrait être important, tant pour la visibilité du code que pour les performances de l'application, de faire un tri des fonctionnalités de gogps pour ne conserver que ce dont il est réellement nécessaire.

Cela serait bien sûre un travail assez complexe nécessitant une analyse rigoureuse et il s'avererai peut-être plus simple de repartir de zéro.

Conclusion

Bien que l'ensemble des fonctionnalités nécessaires au postionnement GNSS par Glonass aient été implémentées, il est frustrant de voir que le travail rendu ne soit pas encore opérationel et nécessite d'être repris.

Il n'est donc pas possible de commencer dès à présent une analyse de l'impact sur la qualité du positionnement qu'apporte la constellation supplémentaire à l'application alors qu'il s'agit d'un sujet important et intéressant.

Toutefois, la documentation réalisée permettra aux futurs développeurs de se plonger plus aisément et plus rapidement dans le code et ainsi mieux se concentrer sur l'analyse des enjeux techniques.

De mon coté, ce projet aura été très intéressant puisque j'aurais développé mes compétences Android, jusqu'à présent très limitées, et pousser mes connaissances sur les systèmes GNSS et en particulier sur le système russe très en avant.

Table des figures

Liste des tableaux

Bibliographie

[1] Algorithme de conversion en jours juliens. URL https://fr.wikipedia.org/wiki/Jour_julien#Jour_julien_astronomique_(AJD)_ou_jour_julien_des_%C3%A9ph%C3%A9m%C3%A9rides_(JDE).

[2] BKG GNSS Data Center. URL https://igs.bkg.bund.de/ntrip/rtcmmessagetypes.

[3] Sidereal Time - Navipedia, . URL https://gssc.esa.int/navipedia/index.php/Sidereal_Time.

[4] ICRF to CEP - Navipedia, . URL https://gssc.esa.int/navipedia/index.php/ICRF_to_CEP.

[5] German Astrophysical Virtual Observatory. URL https://dc.zah.uni-heidelberg.de/apfs/times/q/form.

[6] GnssMeasurement | Android Developers. URL https://developer.android.com/reference/android/location/GnssMeasurement.

[7] GnssNavigationMessage. URL https://developer.android.com/reference/android/location/GnssNavigationMessage.

[8] Sidereal Time Calculator For Any Date, Time and Longitude. URL http://neoprogrammics.com/sidereal_time_calculator/index.php.

[9] ICRF to CEP - Navipedia. URL https://gssc.esa.int/navipedia/index.php/ICRF_to_CEP.

[10] An rtcm3 message cheat sheet. URL https://www.use-snip.com/kb/knowledge-base/an-rtcm-message-cheat-sheet/.

[11] *Glonass Interface Control Document*. Coordination Scientific Information Center, 4.0 edition, 1998. URL https://www.unavco.org/help/glossary/docs/ICD_GLONASS_4.0_(1998)_en.pdf.

[12] C. Fontaine and J. Beilin. Course : Calcul de la position d'un récepteur gnss "à la main". URL http://cours-fad-public.ensg.eu/course/view.php?id=86.

[13] A. Grenier. Developpment of a gnss positionning application under android os using galileo signals, 2019.

[14] *White paper on Using Raw GNSS Measurement on Android Devices*. GSA Raw Gnss Task Force, 2017. URL https://www.gsa.europa.eu/system/files/reports/gnss_raw_measurement_web_0.pdf.

[15] I. LMD/CNRS. Ixion en ligne. URL https://climserv.ipsl.polytechnique.fr/ixion/.

[16] M. Ramandaniaina and S. Gao. Récupération de données de navigation sur android 8, 2018.

[17] J. G. Resendiz Fonseca. Implémentation des algorithmes de géolocalisation sur android, 2018.

[18] *Glonass Interface Control Document*. Russian Institute of Space Device Engineering, 5.1 edition, 2008. URL http://russianspacesystems.ru/wp-content/uploads/2016/08/ICD_GLONASS_eng_v5.1.pdf.

[19] J. Sanz Subirana, J. Juan Zornoza, and M. Hernandez-Pajares. Glonass navigation message - navipedia, . URL https://gssc.esa.int/navipedia/index.php/GLONASS_Navigation_Message.

[20] J. Sanz Subirana, J. Juan Zornoza, and M. Hernandez-Pajares. Glonass satellite coordinates computation - navipedia, . URL https://gssc.esa.int/navipedia//index.php/GLONASS_Satellite_Coordinates_Computation.

[21] J. Sanz Subirana, J. Juan Zornoza, and M. Hernandez-Pajares. *GNSS Data Processing Volume I : Fundamentals and Algorithm*. May 2013. URL https://gssc.esa.int/navipedia/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_I.pdf.

[22] T. Tomoji. Code source rtklib. URL https://github.com/tomojitakasu/RTKLIB/.

Annexe A : Diagrammes de classe initiaux de l'application

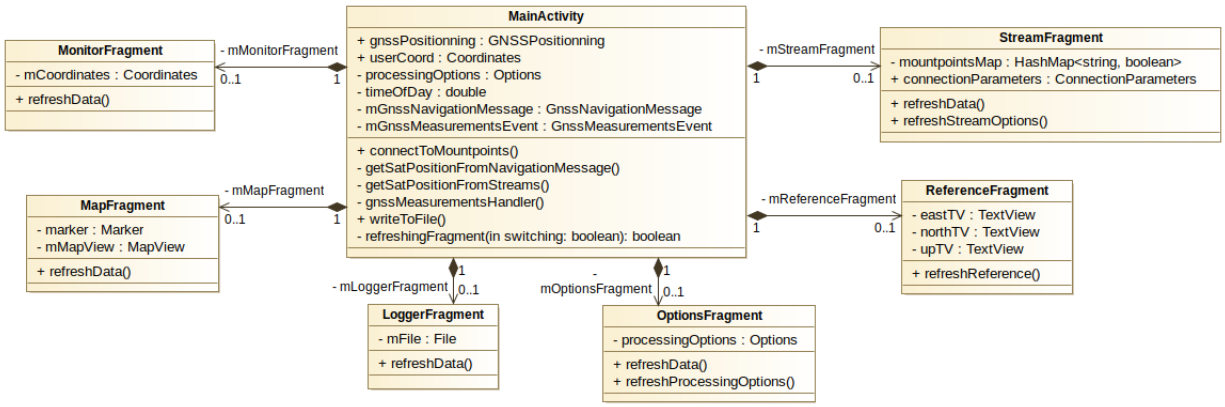


FIGURE 10 – Diagramme de classe simplifié du package "app"

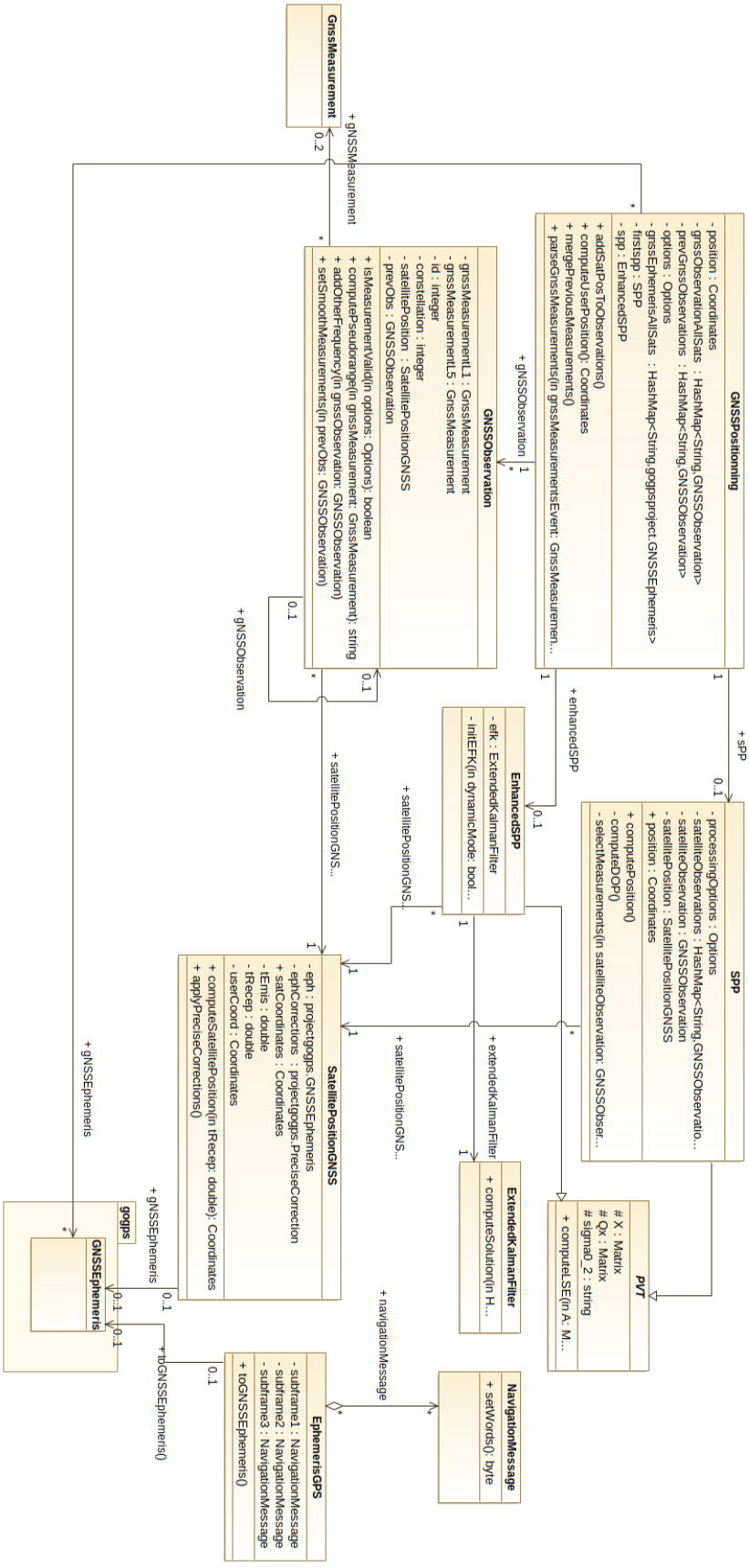


FIGURE 11 – Diagramme de classe simplifié du package "geoloclib"

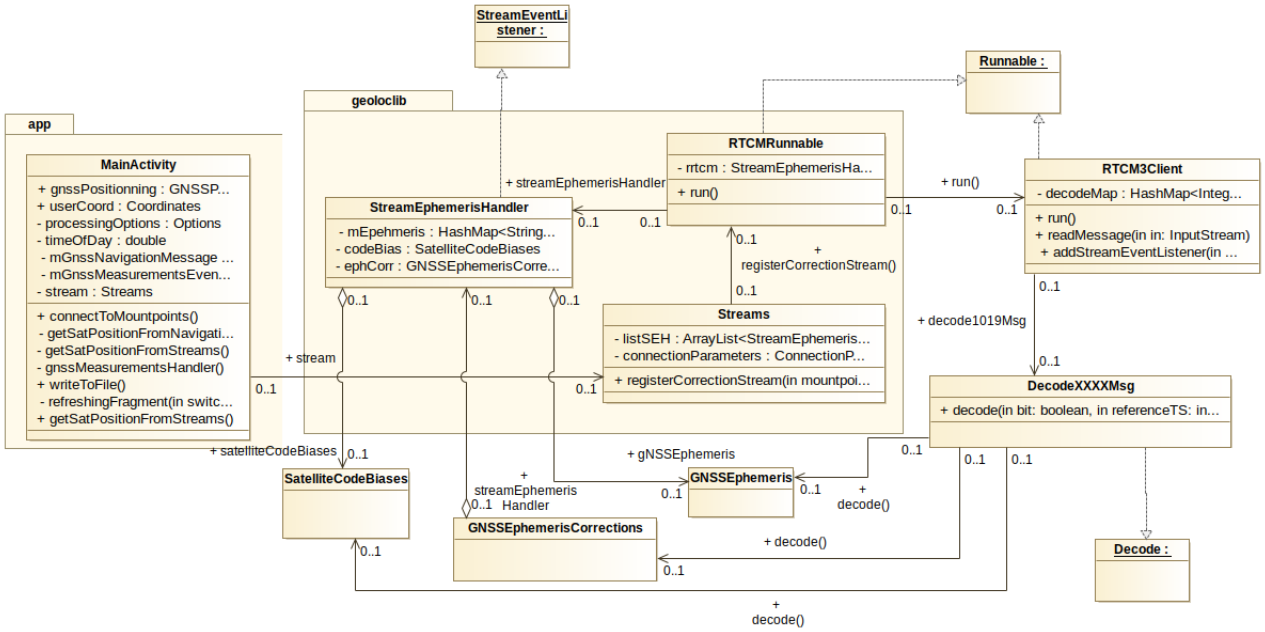


FIGURE 12 – Diagramme de classe simplifié des classes du package "gogps" utilisées par l'application et de leurs interractions

Annexe B : Nouveau diagramme de classe de l'application

FIGURE 13 – Diagramme de classe simplifié de l'application

Annexe C : Calcul du GAST

Dans le calcul de position des satellites il est nécessaire pour passer du repère ECEF au repère ECI en appliquant une rotation de $theta_{Ge} = \theta_{G0} + \omega_e - 3hours$ [20].

NB : Avant de poursuivre je tiens à préciser que la documentation présente pour moi une réelle ambiguïté sur la désignation de θ_{G0} : *"the sidereal time in Greenwich at midnight GMT of a date at which the epoch t_e is specified"* [20]. J'ai apparenté ce terme au GAST qui physiquement fait sens (et c'est aussi l'interprétation de la PYGNSSTOOLBOX de J. Beilin) mais une autre page Navipédia [4] utilise cette même nomination pour le GMST.

Le GAST correspond à l'angle entre le *"true Greenwich meridian"* et le *"true Vernal equinoxe"* [3].
On le calcul comme $\Theta_G = \theta_G + \alpha_e$.
Avec $theta_G$ le GMST et α_e les equations d'équinoxes [4].

1. **Calcul du GMST**

   $\theta_G = 1.002737909350795 * t_{UT1} + \theta_{G0}$
   t_{UT1} le temps d'observation au format UT1

   $\theta_{G0} = 24110.54841 + 8640184.812866 * T + 0.093104 * T^2 - 0.0000062 * T^3$ Avec $T = \frac{Julian-UT1-Date-2451545.0}{36525}$ la date en siècles juliens.

2. **Calcul des équations d'équinoxes**

   $\alpha_e = \arctan \frac{N_{12}}{N_{11}}$
   Les N_{ij} sont les facteurs de la matrice de nutation tels que :
   $N_{11} = -\cos(\epsilon + \Delta\epsilon) * \sin(\Delta\psi)$
   $N_{11} = \sin(\epsilon + \Delta\epsilon) * \sin(\Delta\psi)$ Les paramètres ϵ, $\Delta\epsilon$ et $\Delta\psi$ sont calculé à partir des tables

   du modèle de nutation **IAU1980** [9].

Annexe D : Algorithme de reconstruction de la date de validité

La date de validité du message de navigation n'est pas obtenue de manière immédiate mais doit être reconstruite à partir des *words* N_t *"calendar number of day within four-year interval starting from a leap year"* et N_4 *"the four-year interval number starting from 1996"* grâce à l'algorithme suivant [18] :

Si $1 < N_t < 366$ alors $J = 1$ et $DOY = N_t$
Si $367 < N_t < 731$ alors $J = 1$ et $DOY = N_t - 366$
Si $732 < N_t < 1096$ alors $J = 1$ et $DOY = N_t - 731$
Si $1097 < N_t < 1461$ alors $J = 1$ et $DOY = N_t - 1096$

On trouve l'année d'émission tel que $Y = 1996 + 4 * (N_4 - 1) + (J - 1)$ et le jour dans l'année comme le DOY.