

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

#### **GNSSToolBox**

Package python pour l'enseignement et la recherche en GNSS

jacques.beilin@ensg.eu

## Contexte

#### Objectifs

- Permettre l'enseignement et la recherche dans la thématique GNSS
- Accéder facilement aux données et produits
- Fournir des fonctions de calcul
- Servir de support à des TP

#### Contenu

- un package python
- données et produits GNSS
- un support de TP

# Plusieurs modules à disposition

#### Modules

- orbits : gestion des orbites radiodiffusée et précises
- rinex\_o : données GNSS
- gnss\_const : constantes liées aux GNSS
- gnsstools : fonctions utilitaires
- gnss\_corr : corrections
- gnss\_process : calculs

+freq num

+sv health

+age op inf

Sp3Pos

GNSS

+mjd: modified julian date +X: Coordonnée X en m (ECEF)

+Y: Coordonnée Y en m (ECEF)

+Z: Coordonnée Z en m (ECEF)

+dte: Biais d'horloge sat (us)

#### orbits +leap seconds: integer +ion alpha gps: vecteur[4] (numpy) +ion beta gps: vecteur[4] (numpy) +ion\_gal: vecteur[4] (numpy) +delta utc: vecteur[4] (numpy) +NAV dataG: liste de liste de nav element GPS +NAV dataR: liste de liste de nav element Glonass +NAV dataE: liste de liste de nav element Galileo +Sp3G: liste de liste de Sp3Pos GPS +Sp3R: liste de liste de Sp3Pos Glonass +Sp3E: liste de liste de Sp3Pos Galileo nav element GPS ou Galileo nav element +taps: apsdatetime +const = G ou E Glonass +PRN +tgps: gpsdatetime +TOC +const = R +mjd +PRN +IODE +mid +crs +SV clock offset +delta n +SV relat freg offset +M0 +Message frame time +alpha0 +alpha1 +X dot +alpha2 +MS X acc +cuc +6 +Y dot +cus +MS\_Y\_acc +sqrt a +TOE +Z dot +cic +MS Z acc +OMEGA0

+cis

+10

+crc +omega +OMEGA DOT +TDOT

+code L2

+gps wk

+sv acc

+TGD

+IODC

+sv health

 Chargement d'un fichier d'éphémérides radiodiffusées (\*.yyn ou \*.yyg)

```
1 Orb = orbits.orbit()
Orb.loadRinexN('brdm1500.13p')
```

• Chargement d'un fichier d'éphémérides précises (\*.sp3)

 Récupération des informations d'un fichier d'éphémérides radiodiffusées

```
Eph = Orb.getEphemeris(constellation, PRN, mjd)
try:
    print("TOC : ",Eph.tgps.st_iso_epoch())
except:
    print("Unable to find satellite")
```

Récupération des informations d'un fichier d'éphémérides précises

```
(orb,nl) = Orb.getSp3('G',5)
# Satellite GPS, PRN 5
```

• Calcul d'une position et d'une erreur d'horloge

```
X, Y, Z, dte = Orb.calcSatCoord(const, PRN, mjd,
   degree)
```

Pour accéder aux résultats intermédiaires

```
print(Orb.debug.__dict__)
```

#### Rinex\_o

#### rinex o

+type = o
+headers: liste d'objets "header"

#### header

Plusieurs headers possibles dans un rinex, par exemple lors d'observations stop&go

+VERSION: float +TYPE: string[1] = G,R,S,E,M +PGM: string[20] +RUN\_BY: string[20] +OBSERVER: string[20] +AGENCY: string[40] +MARKER NAME: string[60]

```
+MARKER NUMBER: string[20]
     +REC N: string[20]
     +REC TYPE: string[20]
-> +REC VERS: string[20]
     +ANT N: string[20]
     +ANT TYPE: string[20]
     +X: double
     +Y: double
     +Z: double
     +dH: double
     +dE: double
     +dN: double
     +WAVELENGTH FACTOR: double
     +TYPE OF OBSERV: dict
     +TIME OF FIRST OBS: qpsdatetime
     +TIME OF LAST OBS: gpsdatetime
     +EPOCH FLAG: integer
     +epochs: liste d'objets "epoch"
```

# epoch Contient une epoque avec tous les satellites présents +tgps: gpsdatetime +satellites: liste d'objets "sat" V sat

Contient tous les observables pour un satellite à une époque +type: string = o

+const: string[1] = [G,E,R] +PRN: integer

+obs: dict

## Rinex\_o

Chargement d'un fichier rinex d'observation (\*.yyo)

```
Rnx = rx.rinex_o()
Rnx.loadRinexO('smne1500.13o')
```

• Récupération d'un objet **epoch** pour une date donnée en MJD :

```
t=gpst.gpsdatetime()
t.rinex_t('13 5 30 1 0 30.0000000')
Ep = Rnx.getEpochByMjd(t.mjd)
```

Récupération d'un objet header pour une date donnée en MJD :

```
Hd = Rnx.getHeaderByMjd(t.mjd)
```

# Rinex o

Accès à un observable dans une epoch :

```
C1 = Ep.getObs("C1","G", 31)
```

ou

```
S = Ep.getSat("G", 31)
C1 = S.getObs("C1")
```

ou (valable uniquement pour le C/A code)

```
C1 = Ep.getSat("G", 31).C1
```

• Récupération des observations communes pour une même époque dans 2 rinex :

```
Ep_base,Ep_rover = rinex_base.getCommonEpochs(
    rinex_rover, 56442)
```

# gnsstools

• toolGeoCartGRS80 : conversion de coordonnées géographiques vers des coordonnées cartésiennes. Les angles en entrée sont en radians.

```
1 X,Y,Z = tools.toolGeoCartGRS80(lon,lat,h)
```

 toolCartGeoGRS80 : conversion de coordonnées cartésiennes vers des coordonnées géographiques. Les angles sont en radians.

```
| lon, lat, h = tools.toolCartGeoGRS80(X,Y,Z)
```

 toolCartLocGRS80 : conversion de coordonnées cartésiennes vers des coordonnées locales référencées au point de coordonnées X0.Y0.Z0.

```
|x, y, z| = tools.toolCartLocGRS80(X0,Y0,Z0,X,Y,Z)
```

# gnsstools

 toolAzEle: azimut et élévation (radians) d'un ou plusieurs satellites de coordonnées Xs,Ys,Zs (scalaire ou vecteur) vu(s) depuis un point de coordonnées X,Y,Z.

```
Az, Ele = tools.toolAzEle(X,Y,Z,Xs,Ys,Zs)
```

• toolRotX, toolRotY, toolRotZ : rotation d'angle  $\alpha$  radians autour de l'axe X, Y ou Z d'un point de coordonnées X,Y,Z.

```
1 X,Y,Z = tools.toolRotX(X,Y,Z,alpha)
```

#### gnss\_process

TrilatGps: trilatération à 4 paramètres (X,Y,Z,cdtr)

```
X,Y,Z,cdtr,sigma0_2,V,Qxx = trilatGps(PosSat,Dobs,
X0)
```

 TrilatGnss: trilatération à 4,5 ou 6 paramètres (X,Y,Z,cdtr,[cGGTO,cGPGL])

```
X,Y,Z,cdtr,cGGTO,cGPGL,sigma0_2,V,Qxx =
trilatGnss(PosSat,Dobs,X0,sat_index)
```

 TrilatGnssPonderationElev: trilatération à 4,5 ou 6 paramètres (X,Y,Z,cdtr,[cGGTO,cGPGL]) avec pondération sur l'élévation.