University of Peloponnese

National Observatory of Athens

MSc Space Science Technologies & Applications

Quality Control of daily GPS data from station KLOK

Authors

Ioannis Karavasilis

Spiridon Savvas

Vyron Vasileiadis

February 2, 2017

Abstract

NOA operates Klokotos GNSS station in Thessaly. In this report, we create an automated modular procedure of downloading Klokotos' data from the beginning of its operation until present day, performing quality control on the data and plotting the calculated values of the quality parameters (Multipath and SNR) over this period.

CONTENTS

INTRODUCTION

NOA operates a National GNSS network of thirteen (13) stations transmitting real-time 1-s data to the control centre in Athens. One such station is station THL (Thessaly) is located at 39 deg. 33′ 53.065″ North (latitude) and 22 deg. 00′ 51.7814″ East (longitude). It is also known as Klokotos station, because of the nearby Klokotos village. It is equiped with both seismological and geodetical equipment.

The GNSS equipment was offered by INGV[1] and installation was completed during May 2008. It comprises a Leica receiver (GRX 1200 Pro) with a choke-ring antenna (Leica AT 504). The sampling interval of observations is 1-s while a 5-Hz file is held on the ring-buffer with a life-time of 48 hrs. The 30-s data are available from the NOANET webpage[2] (code KLOK). An image of the installed equipment is shown in Figure 1.

Every station is prone to errors, and to improve station's quality and further reduce common GPS errors, quality control is required. The most important quality parameters are Multipath in frequencies L1 and L2 (MP1 & MP2) and SNR in both frequencies (SN1 & SN2). Both can be obtained utilizing TEQC[3], which is a simple yet powerful toolkit to handle various GNSS data.

In this report, we create an automated procedure of downloading Klokotos' data from the beginning of its operation until present day, performing quality control on the data and plotting the calculated values of the quality parameters over this period.

METHODOLOGY

The procedure is stored in five different files, which are executed linearly. This approach, instead of a united file, offers modularization of the procedure and on-demand execution of the individual steps involved. For example, one could not download Klokotos' data if already has offline access to them, or could easily swap them with another station's data, while keeping the rest of the procedure intact.

[1] http://www.ingv.it/
[2] http://194.177.194.238:8080/noanetgsac
[3] https://www.unavco.org/software/data-processing/teqc/teqc.html

Figure 1: GNSS equipment of Klokotos station in Thessaly.

Download

First step is to download the data from NOANET GSAC repository. Luckily, GSAC provides a search engine with many parameters. We are only concered about two; the file type and the site code. We check both *RINEX GPS navigation file* and *RINEX observation file* for downloading, while providing the code name *KLOK* in the site code text field. These settings are shown in Figure 2. GSAC provides a variety of different download options and formats, including a bash script containing `wget` commands. We choose this option as it can be easily included in our automated workflow. After clicking `'Wget Script for FTP download'` button we download `gsacwget.sh` and save it in our workspace directory.

Next we execute `0_download.sh` script. This script creates `zipped/` directory, if does not exist, and executes `gsacwget.sh` script in it. The result is the `zipped/` directory populated with lots of zipped `.Z` files, one for each day of observation.

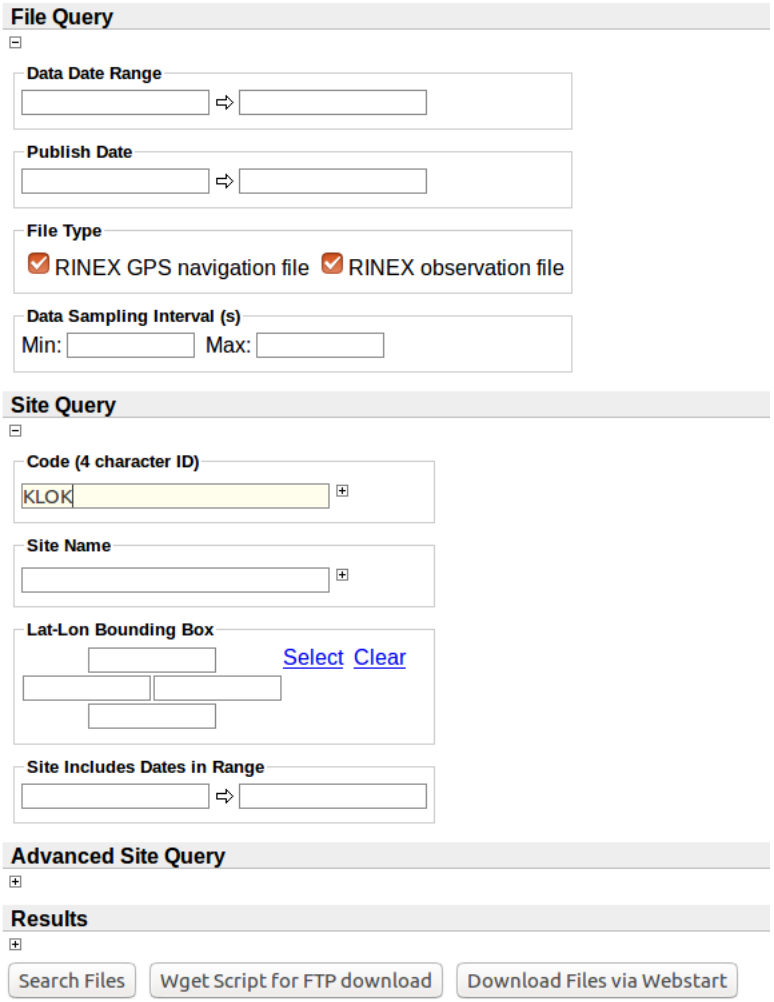


Figure 2: Settings for downloading required Klokotos' data through NOANET GSAC repository

```
#!/bin/bash
if [ ! -d zipped ]; then
        mkdir zipped
fi
cp gsacwget.sh zipped/
cd zipped/
/bin/bash gsacwget.sh
rm gsacwget.sh
cd ..
```

o_download.sh

Unzip

Next step is to unzip the files. For this task we execute `1_unzip.sh` script. `1_unzip.sh` creates the directory `unzipped/`, if it does not exists, extracts the contents of all the `.Z` archive files in `zipped/` directory and moves them to the `unzipped/` directory. This procedure's result is `unzipped/` directory populated with CompactRINEX observation files `*.*d` and RINEX GPS navigation files `*.*n`

```
#!/bin/bash
if [ ! -d unzipped ]; then
        mkdir unzipped
fi
for zipped in zipped/*.Z; do
        7z e $zipped -ounzipped/
done
```

1_unzip.sh

Convert

CompactRINEX observation files are not compatible with TEQC. We first need to convert them to regular RINEX observation files. For this task, we use `CRX2RNX` tool, which can be downloaded from its website[4]. To convert all files, we execute `2_convert.sh`, which converts all CompactRINEX `*.*d` to RINEX `*.*o` files.

```
#!/bin/bash
for crx in unzipped/*d; do
        CRX2RNX $crx
done
```

2_convert.sh

Quality Control

Next step is to perform the actual quality checking using TEQC. TEQC command for quality checking takes as input argument

[4] http://terras.gsi.go.jp/ja/crx2rnx.html

a single RINEX observation file, finds automatically the navigation file with the same name, and outputs a full report into a file with the same name and suffix `S`. For example, if we run `teqc +qc klok0010.10o` the report file would be `klok0010.10S`.

To perform quality checking to all files we execute script `3_qc.sh`, which also moves the report files to a seperate `reports/` directory to simplify the code of the last step.

```
for rnx in unzipped/*o; do
        teqc +qc $rnx
done
if [ ! -d reports];  then
        mkdir reports
fi
for report in unzipped/*S; do
        mv $report reports/
done
```

3_qc.sh

Analyse data & plot

The final step involves analyzing the data from the produced reports and plotting the results. It is written in Python v3.6 and the source code can be examined in `4_plot.py`.

```
#!/usr/bin/env python
import glob
import math
import matplotlib.pyplot as plt
import datetime as dt

def convert_snr(r):
    SNR = [(),(25, 26), (26, 28), (28, 32), (32, 36), (36, 39), (39, 42), (42, 45),
    b = math.floor(r)
    if b == 0:
        return 25
    elif b == 9:
        return 49
    else:
        return SNR[b][0] + ((SNR[b][1] - SNR[b][0]) * (r - b))
```

```
input_directory = "reports/"
output_direcory = "plots/"

years = ["08", "09", "10", "11", "12", "13", "14", "15", "16", "17"]
reports = []

for i in years:
    for r in sorted(glob.glob(input_directory + "*" + i + "S")):
        reports.append(r)

M1 = []
M2 = []
S1 = []
S2 = []

base = dt.date(2008, 6, 17)
dates = [base + dt.timedelta(days=x) for x in range(0, len(reports))]

for report in reports:
    with open(report, encoding='utf-8') as f:
        for line in f:
            if line.startswith('Moving average MP12'):
                splitted = line.split(':')
                M1.append(float(splitted[1].replace('m', '').strip()))
                continue
            if line.startswith('Moving average MP21'):
                splitted = line.split(':')
                M2.append(float(splitted[1].replace('m', '').strip()))
                continue
            if line.startswith('Mean S1'):
                splitted = line.split(':')
                S1.append(convert_snr(float(splitted[1].strip().split(' ')[0].strip(
                continue
            if line.startswith('Mean S2'):
                splitted = line.split(':')
                S2.append(convert_snr(float(splitted[1].strip().split(' ')[0].strip(
                continue

fig, ax = plt.subplots()
plt.xlabel('Time (year)')
plt.ylabel('Multipath (m)')
ax.plot(dates, M1, 'r+', label='MP1 KLOK')
ax.plot(dates, M2, 'gx', label='MP2 KLOK')
legend = ax.legend(loc='upper right', numpoints=1)
plt.show()
```

```
fig, ax = plt.subplots()
plt.xlabel('Time (year)')
plt.ylabel('Mean SNR (dBHz)')
ax.plot(dates, S1, 'r+', label='SN1 KLOK')
ax.plot(dates, S2, 'gx', label='SN2 KLOK')
legend = ax.legend(loc='upper right', numpoints=1)
plt.show()
```

4_plot.py

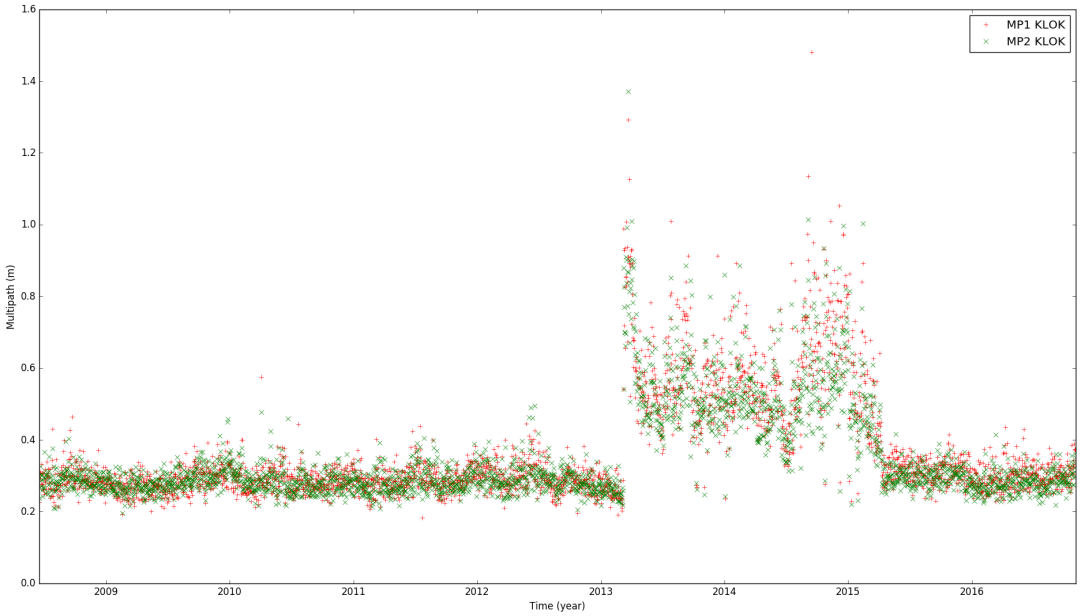
Despite being larger than the rest scripts, `5_plot.py` is essentialy simple. First, it reads all the filenames of the reports, stores them in a list and rearranges the list in chronological order. Then it reads the files one by one, retrieves the quality parameters' values, multipath averages MP1 & MP2 and mean SNRs SN1 & SN2, and stores them in four lists keeping the chronological order. Finally, it create one plot with the the moving average timeseries and one plot with the two mean SNRs timeseries.

RESULTS

Generated plots can be saved as image files and are shown in Figure 3. The first plot shows the distribution of multipath values over time, while the second shows the distribution of mean SNR values over time.

Studying the timeseries, it is clear that, while the range of values is minimal and predictable from the beginning of Klokotos operation in 2008, there has been a disturbance starting mid 2013 until mid 2015. This disturbance causes a large increase in multipath error for both frequencies L1 and L2. Also, there is a large decrease in SNR values for frequency L1 and a very minor decrease in SNR valuess for frequency L2.

We can assume that a large object was placed near the station during that period causing reflections of signals and thus multipath errors.



(a) Plot 1. Values of multipath averages over time.

(b) Plot 2. Values of mean SNRs over time.

Figure 3: Results of quality control procedure