GPS Solutions (2023) 27:197
https://doi.org/10.1007/s10291-023-01529-y

GHASP: a Galileo HAS parser

D. Borio[1] · M. Susi[1] · C. Gioia[2]

Received: 2 February 2023 / Accepted: 3 August 2023 / Published online: 5 September 2023

Abstract
Galileo High Accuracy Service (HAS) corrections are broadcast through the E6B signal using a high-parity vertical Reed-Solomon encoding scheme, which reduces message recovery time and improves transmission reliability. To recover HAS corrections, it is thus necessary to invert the encoding process and interpret the decoded bits. In order to foster HAS adoption and facilitate experimentation with HAS corrections, a Galileo HAS Parser (GHASP) has been developed. GHASP is available open-source and supports different input data types from different receiver manufacturers. Decoded corrections are provided in Comma-Separated Values files, which can be directly loaded using common data-science languages. In this way, corrections are readily available and can be used not only for Precise Point Positioning (PPP) applications but also for scientific analysis such as clock characterization using the Allan Deviation.

Keywords Galileo · Decoder · High accuracy service · HAS · Corrections

Introduction

The Galileo High Accuracy Service (HAS) is providing Precise Point Positioning (PPP) corrections enabling decimeter-level accuracy (EUSPA 2023). The corrections, which are mainly disseminated through the E6B Signal-in-Space (SIS), are also available for registered users through a dedicated Internet channel (EUSPA 2023b), making HAS accessible to a wider range of receivers without requiring the processing of the E6B signal.

The HAS corrections distributed through the E6B signal are encoded using a high-parity vertical Reed-Solomon (RS) code (Fernández-Hernández et al. 2020). The original HAS message pages are vertically stacked and multiplied by the Reed-Solomon encoding matrix. In this way, a new redundant set of pages is obtained. Each Galileo satellite broadcasts a subset of such pages. The properties of Reed-Solomon codes guarantee that any subset of K different pages can be used to reconstruct the original HAS message. Here, K is the size in pages of the original HAS message. The encoding and dissemination scheme adopted by the HAS reduces the Time-to-Retrieve Data (TTRD) (Borio et al. 2020) and improves the reception performance, owing to the redundancy introduced by the Reed-Solomon encoding scheme.

The recovery of the HAS corrections requires inverting the encoding scheme and interpreting the decoded bits. Details on the HAS encoding scheme and correction format can be found in the HAS Interface Control Document (ICD) (European Union, 2022) published by the European Space Programme Agency (EUSPA).

To facilitate the adoption of HAS corrections, HAS correction decoders have been developed (Gioia et al. 2022 and Horst et al. 2022). Both decoders provide a Python implementation of the processes needed to recover the HAS corrections from the raw E6B symbol stream obtained using, for instance, a high-end receiver such as the Septentrio PolaRx 5.

Horst et. al. (2022) developed the *HASlib*, which supports only the processing of Septentrio Binary Files (SBF). HASlib code is available open source in GitHub (https://github.com/nlsfi/HASlib) and can convert HAS corrections

The GPS Toolbox is a topical collection dedicated to highlighting algorithms and source code utilized by GNSS engineers and scientists. If you have a program or software package you would like to share with our readers, please submit a paper to the GPS Toolbox collection or email https://geodesy.noaa.gov/gps-toolbox/ for more information. To download source code from this or any GPS Toolbox paper visit our website at http://geodesy.noaa.gov/gps-toolbox.

✉ D. Borio
daniele.borio@ec.europa.eu

[1] European Commission, Joint Research Centre, 21027 Ispra, VA, Italy

[2] Piksel, Milano, Italy

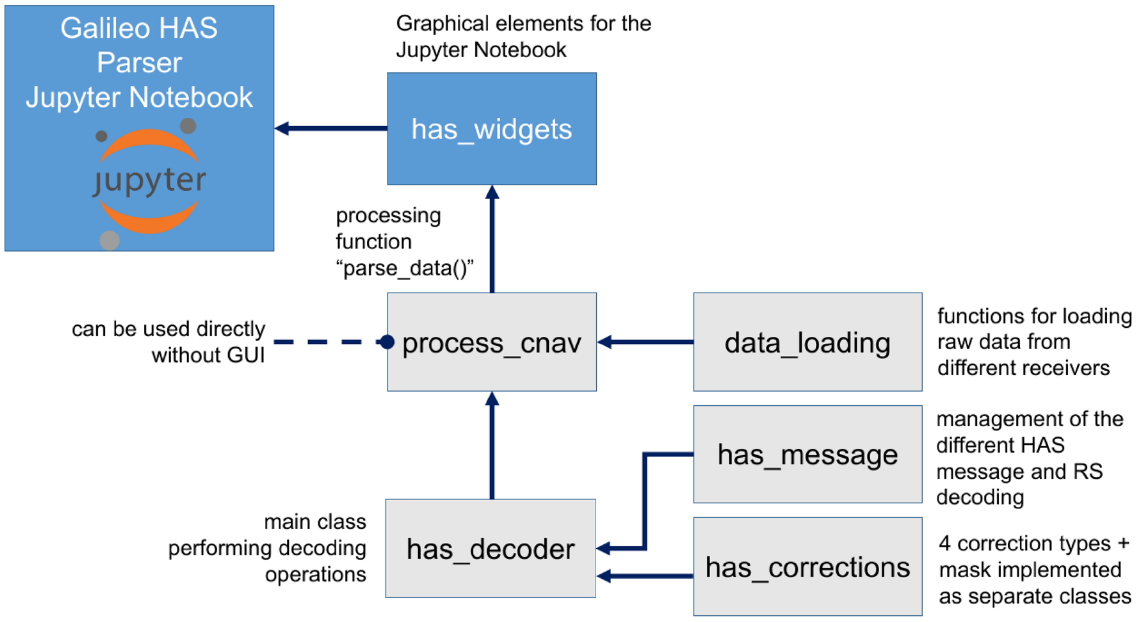

Galileo HAS Parser Jupyter Notebook
jupyter
Graphical elements for the Jupyter Notebook
has_widgets
processing function "parse_data()"
can be used directly without GUI
process_cnav
data_loading
functions for loading raw data from different receivers
has_message
management of the different HAS message and RS decoding
main class performing decoding operations
has_decoder
has_corrections
4 correction types + mask implemented as separate classes

Fig. 1 Overall architecture of GHASP. Grey boxes represent core elements, whereas blue boxes indicate GUI elements

into the RTCM 3 or the International GNSS Service (IGS) State-Space Representation (SSR) formats (Hirokawa et al. 2021, IGS 2020), which are directly supported by some open-source PPP software.

In parallel, the authors developed a Python HAS decoder that was used for several analyses (Gioia et al. 2022), including the assessment of the HAS demodulation performance at high-latitudes (Susi et al. 2021). While the basic functioning and architecture of the decoder were described in (Gioia et al. 2022), the related code was never published open source. In this paper, we describe the latest version of the HAS decoder introduced in (Gioia et al. 2022) that is now provided open source and available under the GPSToolbox repository and in GitHub (https://github.com/borioda/HAS-decoding). With respect to the version used in Gioia et al. (2022), the software has been enriched by several functionalities including the support to several receiver formats, a Graphical User Interface (GUI), plotting capabilities and the possibility to evaluate the Allan Deviation (ADEV) from the clock corrections.

The decoder developed in this context is denoted as GHASP, a Galileo HAS Parser. With respect to the version described in (Gioia et al. 2022) and to HASlib, additional input data formats are supported including Septentrio SBF, Javad and NovAtel raw data.

The decoder features a simplified GUI provided as a Python Jupyter notebook (https://jupyter.org/). Exploiting the GUI, the decoder can be operated in a easier and more intuitive way.

Differently from HASlib, corrections are saved into four Comma-Separated Values (CSV) files containing the four different correction types: orbit corrections, clock corrections, phase and code biases. This choice has been adopted in order to simplify the access to the corrections, which can be directly loaded, without any additional parsing operations, using any data-science language, such as Matlab, Python or R. In this way, HAS corrections can be directly plotted, analysed and included in customized navigation software. For instance, HAS clock corrections can be directly used for estimating the ADEV of the corresponding satellite. This possible application is briefly discussed in the following.

The remainder of this article is organized as follows: the architecture of the code is first described. The user interface, input and output data are then introduced. Some use cases are briefly discussed before the conclusions.

Architecture of the code

The overall architecture of GHASP is presented in Fig. 1: core elements are represented by grey boxes, whereas GUI modules are depicted as blue boxes.

The main processing loop is implemented in the "process_cnav.py" module, which contains the *parse_data*() function and a main routine that can be modified, for example, to implement batch processing, i.e. the processing of several input files. The *parse_data*() function is also called by the "has_widgets.py" module, which implements the different graphical elements used by the Jupyter notebook GUI, whose use is briefly described in the next section.

The *parse_data*() function loads the different input files using the functions available in the "data_loading" module and instantiates a "has_decoder" object, which is responsible for the different decoding operations. The decoder has a list of HAS messages that are progressively decoded. Reed-Solomon decoding is performed in the "has_message" module.

Messages are converted into corrections: four correction types are currently supported and correspond to the four HAS correction types currently described into the HAS ICD (European Union, 2022). Each correction type has been implemented as a separate class. In this respect, the code features a flexible object-oriented structure and can be easily extended to support additional correction types. A description of the base correction class and its four derived classes can be found in (Gioia et al. 2022).

In addition to the four correction classes, a "mask" class has also been implemented. It is used to process and store the information broadcast into the mask field of the HAS message.

Using the different objects described, GHASP can parse different input data files and extract the corresponding HAS corrections.

User interface, supported data and output format

As indicated above, a simple GUI, implemented as a Jupyter notebook, was developed to facilitate the use of GHASP. A screenshot of the GUI is shown in Fig. 2: parsing HAS corrections from receiver raw data can be performed in a few simple steps. First, the required Python libraries are loaded, then the user is asked to choose the input file containing the raw E6B bits. This is done through the Python FileChooser() widget that opens a pop-up window allowing the selection of the input file. Several data formats are supported including:

- Septentrio binary and parsed data
- Javad raw data
- NovAtel raw data.

Figure 2 shows the widgets allowing the user to specify the input format. A dropdown menu allows one to select the receiver type, in the example Septentrio. Following the receiver selection, additional options are prompted. For the Septentrio case, different options are available including binary SBF files or converted data. Septentrio receivers are shipped with conversion tools that allow one to convert SBF files into text files. Depending on the version of the Septentrio conversion tool, E6B data can be parsed into decimal or hexadecimal formats. GHASP supports both formats and the different options can be selected through the GUI. After setting the input file and format, data parsing and decoding can start. This action is triggered by the "Parse" button visible at the bottom of Fig. 2.

In addition to the Jupyter notebook, it is possible to parse data by directly modifying the "process_cnav.py" script. As discussed above, this Python script calls the main function of the GHASP library and performs the same operations available through the Jupyter notebook.

As already mentioned, the output of the parser is a set of four CSV files, one for each type of correction. The different columns of the CSV files are described in the related header and include the Time of Week (ToW), the Week Number (WN) and the Time of Hour (ToH). Each field corresponds to the different elements described in the HAS ICD (European Union, 2022).

Detailed information of the structure of the output CSV files can be found in the GHASP user manual (Borio et al. 2023).

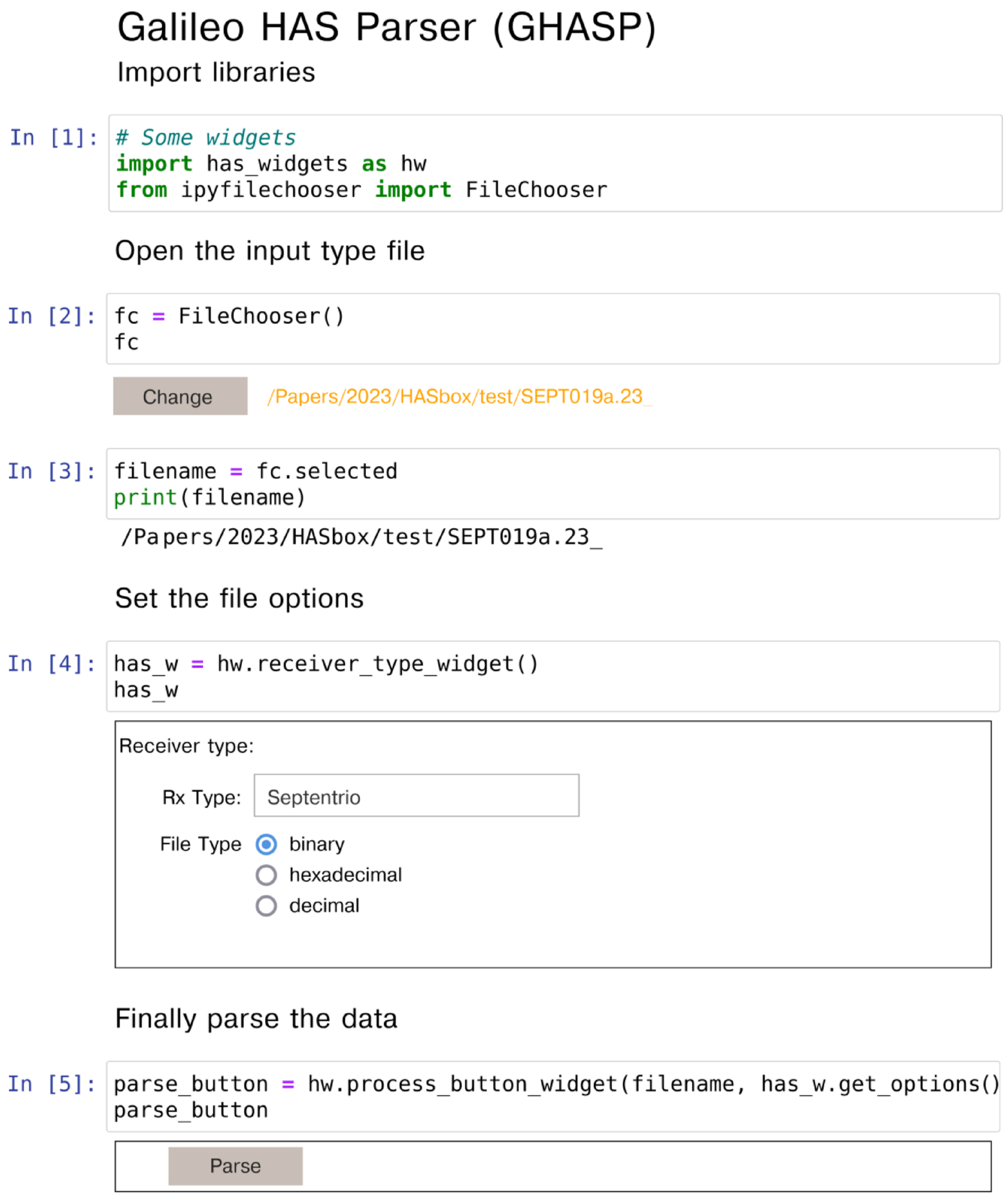
Data processing: examples

In addition to the core library, the GHASP library is provided with simple plotting routines that can be used to display and analyse the different corrections. In the following, data extracted from a Septentrio SBF file are displayed. The dataset is from October 20th, 2022. While only results for Galileo satellites are presented, the same kind of plot can be produced for the GPS case as well. Additional sample plots can be found in the GHASP user manual (Borio et al. 2023).

Figure 3 shows the corrections broadcast by the HAS service for the different Galileo satellites on October 20th, 2022. The HAS orbit corrections are expressed with respect to the radial, tangential (along track) and normal (cross-track) components of the Satellite Coordinate System (SCS) centred into the satellite's ionosphere-free antenna phase centres (European Union, 2022) that are the E1-E5b and L1- L2C phase centres for Galileo and GPS, respectively. The time series displayed in three different subplots are zero mean and assume amplitudes lower than one metre. As detailed in (European Union, 2022), the orbit corrections, after a conversion into the broadcast Earth Centred Earth Fixed (ECEF) frame, have to be added to the satellite position computed from the broadcast data.

The time series of the corresponding Galileo clock corrections are shown in Fig. 4. Also in this case, the time series have a nominal behaviour with a zero-mean value. The HAS clock corrections are provided in metres, and after conversion in seconds, must be added as delta offset to the broadcast ionosphere-free clock offset (European Union, 2022).

The clock corrections can be used to compute the ADEV, which quantifies the stability of the clock corrections.


Galileo HAS Parser (GHASP)

Import libraries

```
In [1]: # Some widgets
        import has_widgets as hw
        from ipyfilechooser import FileChooser
```

Open the input type file

```
In [2]: fc = FileChooser()
        fc
```

Change /Papers/2023/HASbox/test/SEPT019a.23_

```
In [3]: filename = fc.selected
        print(filename)
```

```
/Papers/2023/HASbox/test/SEPT019a.23_
```

Set the file options

```
In [4]: has_w = hw.receiver_type_widget()
        has_w
```

Receiver type:

Rx Type: Septentrio

File Type [✓ binary] hexadecimal decimal

Finally parse the data

```
In [5]: parse_button = hw.process_button_widget(filename, has_w.get_options()
        parse_button
```

Fig. 2 Screenshot of the Jupyter notebook developed to simplify the user interaction with the HAS decoder

A sample plot showing the overlapping ADEVs for the different Galileo satellites is provided in Fig. 5.

Routines for plotting code and carrier phase biases are also provided as part of GHASP. Sample plots for the code

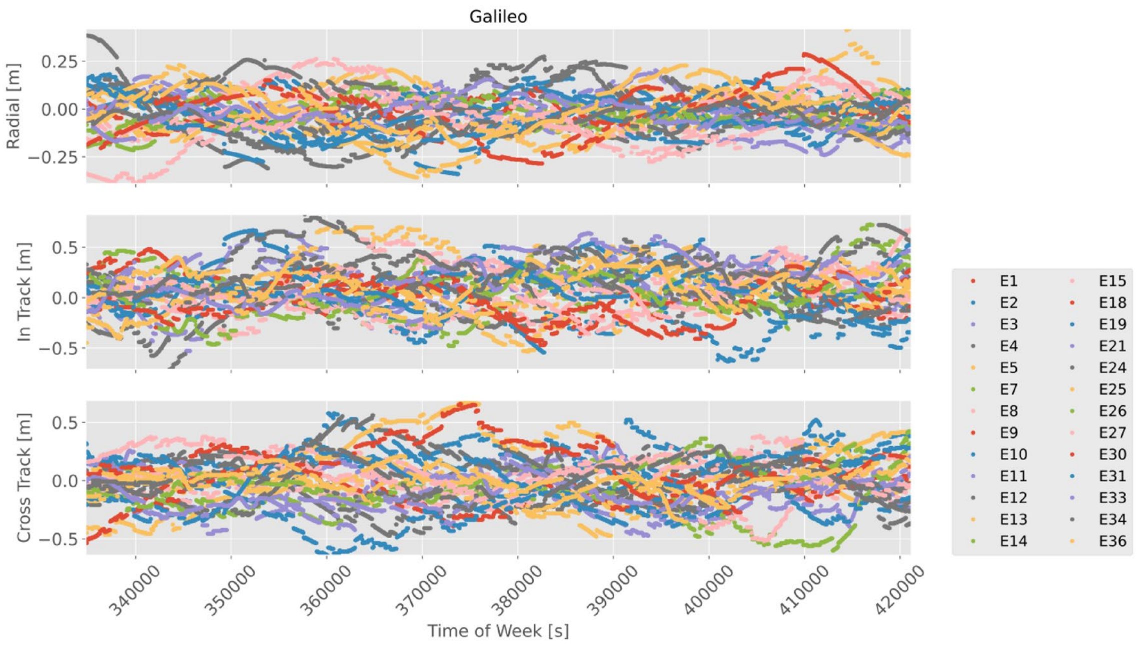


Fig. 3 The three components of the orbit corrections broadcast by the HAS service for the different Galileo satellites. Corrections for the whole day of October 20th, 2022

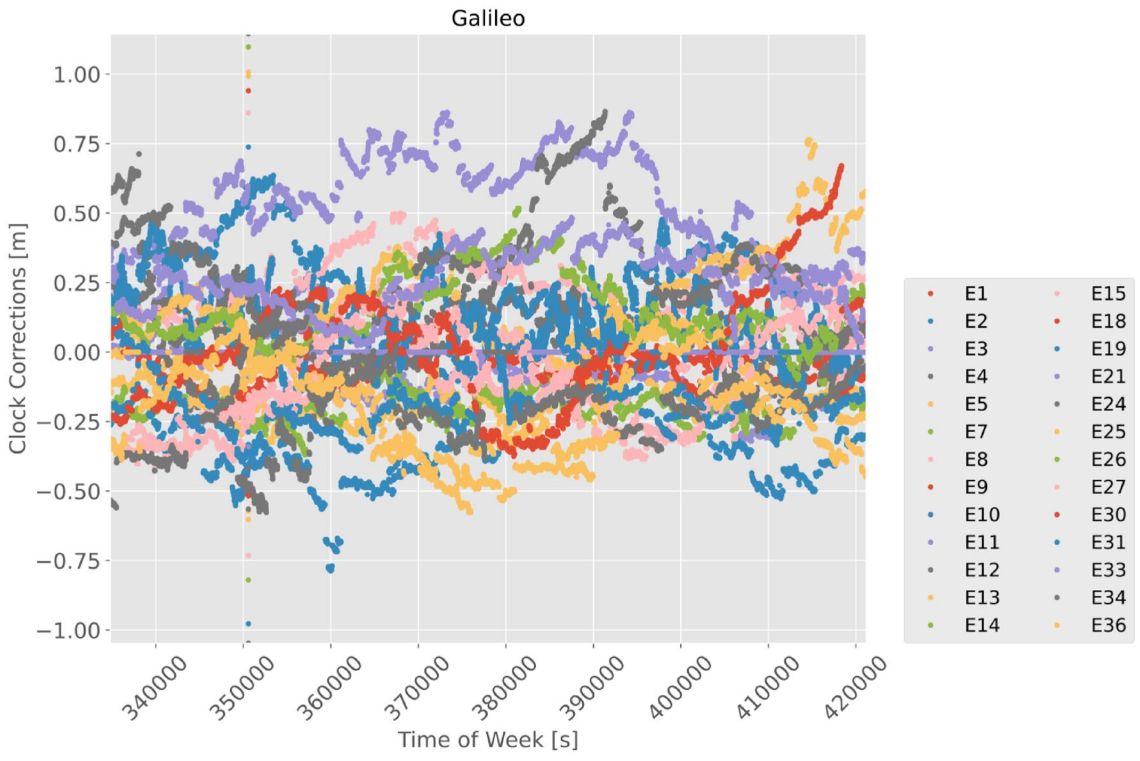


Fig. 4 Clock corrections broadcast by the HAS service for the different Galileo satellites. Corrections for the whole day of October 20th, 2022

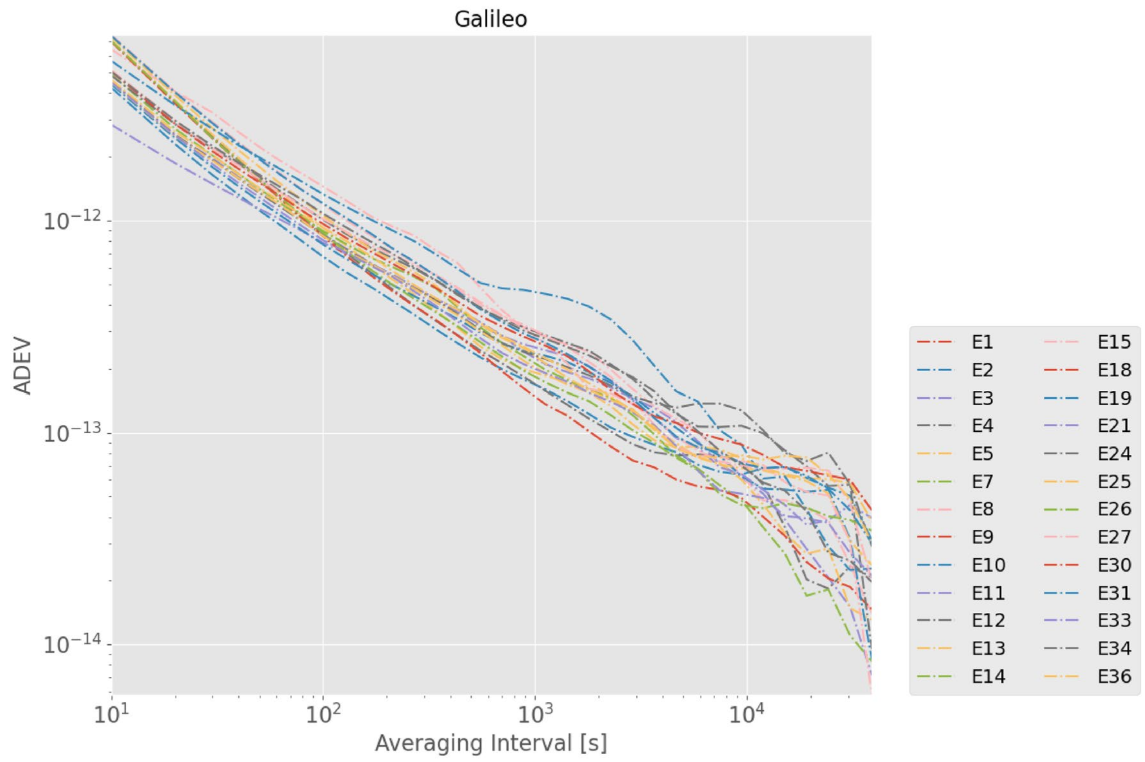


Fig. 5 Overlapping ADEV computed using the clock corrections broadcast by the HAS service for the different Galileo satellites. Corrections for the whole day of October 20th, 2022

and carrier phase biases can be found in the GHASP user manual (Borio et al. 2023).

Conclusions

A library for parsing Galileo HAS corrections from different receiver files containing the E6B navigation message in binary format has been developed. The parser allows one to experiment with the Galileo HAS corrections. A simple GUI facilitates the usage of the parser and the configuration of the different options. The parser is shipped with routines for plotting the parsed corrections and demonstrating operations such as the computation of the ADEV for the different corrected satellites.

Author contributions Daniele Borio developed the core elements of the decoder. Melania Susi and Ciro Gioia contributed to the data collection and analysis. All the authors contributed to the testing and debugging activities, including the development of the different plotting and analysis routines. Daniele Borio wrote the main manuscript, which was complemented and reviewed by the other authors.

Data availability Data can be found at https://github.com/borioda/HAS-decoding

Declarations

Competing interests The authors declare no competing interests.

References

Borio D, Senni T, Fernández-Hernández I (2020) Experimental analysis of a candidate Galileo E6-B data dissemination scheme. In: Proceedings of the international technical meeting (ITM) of the institute of navigation, San Diego, California, Jan. 2020, pp 509–520. https://doi.org/10.33012/2020.17159

Borio D, Gioia C, Susi M (2023) GHASP: a Galileo high accuracy service parser, user manual. https://github.com/borioda/HAS-decoding

European Union (2022) Galileo high accuracy service signal-in-space interface control document (HAS SIS ICD), Issue 1.0, May 2022. https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_HAS_SIS_ICD_v1.0.pdf

European Union Agency for Space Programme, EUSPA (2023b) Galileo HAS internet data distribution. https://www.gsc-europa.eu/galileo/services/galileo-high-accuracy-service-has/internet-data-distribution

European Union Agency for Space Programme, EUSPA (2023) Galileo high accuracy service (HAS). https://www.euspa.europa.eu/european-space/galileo/services/galileo-high-accuracy-service-has

Fernández-Hernández I, Senni T, Borio D, Vecchione G (2020) High-parity vertical reed-Solomon codes for long GNSS high-accuracy messages. Navigation 67(2):365–378

Fernandez-Hernandez I, Chamorro-Moreno A, Cancela-Diaz S, Calle-Calle JD, Zoccarato P, Blonski D, Mozo A (2022) Galileo high accuracy service: initial definition and performance. Gps Sol 26(3):65

Gioia C, Borio D, Susi M, Fernández-Hernández I (2022) The Galileo high accuracy service (HAS): Decoding and processing live corrections for code-based positioning. In: Proceeding of the 2022 international technical meeting of the institute of navigation, Long Beach, California, pp 1065–1074. https://doi.org/10.33012/2022.18204

Hirokawa R, Fernández-Hernández I, Reynolds S (2021) PPP/PPP-RTK open formats: overview, comparison, and proposal for an interoperable message. Navigation 68(4):759–778. https://doi.org/10.1002/navi.452

Horst O, Kirkko-Jaakkola M, Malkamäki T, Kaasalainen S, Fernández-Hernández I, Chamorro-Moreno A, Cancela-Díaz S (2022) HASlib: An open-source decoder for the galileo high accuracy service, Proceeding of the 35th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2022), Denver, Colorado, September 2022, pp 2625–2633. https://doi.org/10.33012/2022.18508

International GNSS Service (IGS) (2020) IGS state space representation (SSR) format version 1.00. https://files.igs.org/pub/data/format/igs_ssr_v1.pdf

Susi M, Borio D, Gioia C, Brunes MT, Dähnn M, Grinde G, Rost C (2021) assessment of galileo E6B data demodulation performance at high latitudes: a norwegian vessel case study. Remote Sens 13:4669. https://doi.org/10.3390/rs13224669

D. Borio received the M.S. degree in communications engineering from Politecnico di Torino, Italy, and the M.S. degree in electronics engineering from ENSERG/INPG de Grenoble, France, in 2004. He received the doctoral degree in electrical engineering from Politecnico di Torino in April 2008. From January 2008 to September 2010, he was a senior research associate in the PLAN group of the University of Calgary, Canada. He is currently a scientific policy officer at the European Commission (EC) Joint Research Centre in the fields of digital and wireless communications, location and navigation.

M. Susi received her B.E. in Mathematical Engineering from the Università TorVergata di Roma,Italy, and M.Sc.s in Communication Engineering and in Geomatic Engineering, respectively, from the Università di L'Aquila, Italy, and from the University of Calgary, Canada. She holds a Ph.D. from the University of Nottingham where she was a Marie Curie fellow. After being a scientific technical officer at the Joint Research Centre of the European Commission she is now a GNSS senior researcher at Topcon in Concordia, Italy.

C. Gioia received the MSc. in Nautical Sciences and the PhD from Parthenope University in 2009 and 2014, respectively. From May 2014 to June 2016, he worked as a consultant for the European Commission Joint Research Centre (JRC). From July 2016 to July 2022, he was a scientific project officer at the JRC focusing on location and navigation with special emphasis on geomatics aspects. Since September 2022, he is a consultant for the JRC.