Ondrej Kafka
kafkaon1@fel.cvut.cz

Section 1 Introduction

The LGC report generator is an interactive extension to Survey Pad (SP), a well-established software within the CERN BE-GM group. SP primarily functions as a visualization tool utilized by surveyors for visualizations of formatted text results of surveying-related calculations. It enables features like inter-file hyperlinks, auto file formatting, and coloring or inspecting a file's structure. As SP serves for visualizations, the mentioned computations are performed by SP computing plugins such as LGC++ or Chaba. LGC is the basic building block for the LGC HTML report generator developed during this project.
The results from LGC used to be produced as a text file formatted by SP. Still, they didn't allow for much interactivity for users, such as sorting or interactive plots (histograms were displayed as text, too). The lack of interactivity served as the primary motivation behind this project and remained its central objective. Another objective was to produce a shareable dependency-free formatted results file, as the pure text without SP formatting may often be hard to orient in.
An interactive demo can be found here.

Section 2 Pipeline and Surveypad Integration

The LGC HTML report generator is built on top of a pipeline provided by SP and LGC. Visualization of this pipeline can be seen in Fig. 1. The data flow starts with an .lgc input file containing measured values. SP then uses LGC to produce a .res text file with results and a .json file. This .json file is then inserted into an HTML report template (previously built into a single file prepared for this step). Then, upon opening, the report HTML file processes the .json data and renders all its interactive features. Since SP also offers a browser-like environment so that the HTML interactive report file can be viewed directly there.

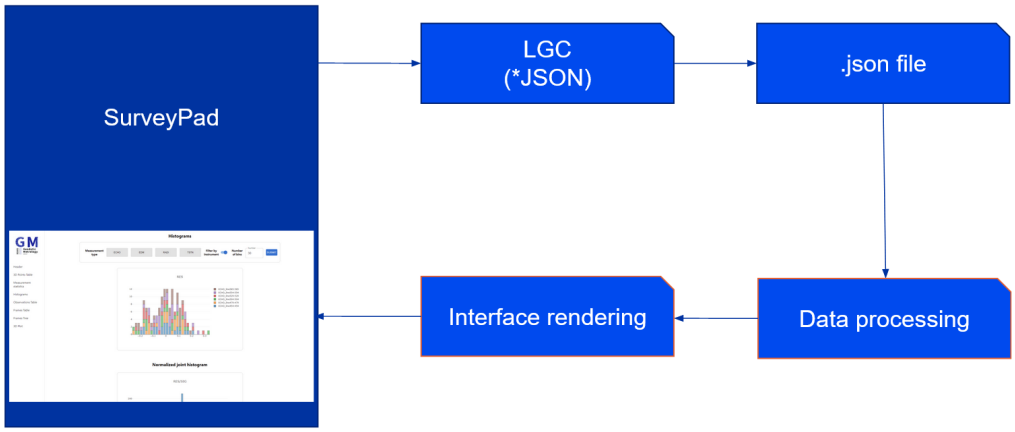

Fig. 1: Pipeline of HTML report generation.

Moreover, since the file can be viewed inside the Survey Pad, mimicking browser behavior, SP can then listen to hyperlink click events. This allows for implementing inter-file hyperlinks in HTML

reports, jumping from the report file to the input file. The links have to have a specific structure, and if SP recognizes a link click with such form, it uses them to redirect users to a particular line in the input file.
Lastly, the HTML file is produced as a single, dependency-free file in the output folder along with the text results file. This file can then be easily shared, creating a straightforward channel for transmitting the computation results in formatted and interactive ways.

Section 3 Implementation details

The report was implemented using JavaScript, namely React.js. As one of the objectives was to compile all source codes into one file to make it easily shareable, another used framework was Webpack with a custom configuration to produce a standalone offline-working .html file. The implementation comprised about 3,500 lines of code, which was directly caused by the complexity of the data processed before rendering the report. The structure of the .json file can be seen in Fig. 2. This structure is very nested and follows the inner data structure of LGC, not the format needed for front-end rendering, and thus needs to be heavily processed to obtain all the data required for rendering the front end from all around the file. This processing is done only once upon opening the file. To prevent recomputing and secure reasonable performance even for big files (cca. ten thousand observations), all of this processing and other smaller computations use *useMemo()* function to cache the results and prevent recomputing on each re-render.
For the sake of the maintainability of the code, all the paths to the data from the .json file used in the front end are specified in a single file so that future maintainers don't have to search for the way of obtaining data from .json if the file structure changes. It just suffices to change the path. The path syntax is similar to web paths, e.g.

```
measECHO/i/distancesResiduals/0/fValue,
```

returning distance residual of ECHO measurement type, where the subparts are keys in nested dictionaries. However, not all of the displayed values can be found directly in the .json file, and in keeping with the theme described above, I introduced a notation for calculating directly from paths using exclamation marks, as can be seen in this snippet (three dots standing for missing parts of code):

```
export const generateDVERObsCols = () => {
  return {
    ...
    TGTLINE: fieldGen( ...{path:"lkp:targetPos" }),
    CALC: ...path: "!distances/0/fValue!+!residuals/0/fValue!"...
  };
};
```

The CALC value was computed by adding values from two paths. The snippet also illustrates another feature: taking the link position from the precomputed lookup (lkp) table, where the key is the value in the path. This feature is introduced since the lookup information is at a very different location than the rest of the data when processing the current observation. It thus would create big textual and performance overhead to use a normal path. The exclamation computation approach also supports mathematical operations such as *sqrt()* or *abs()*. An example of full-column data definition can be found in Fig. 3.
The source code for the tool implementation can be found here. It follows the classical React.js file structure, as shown in Fig. 4.

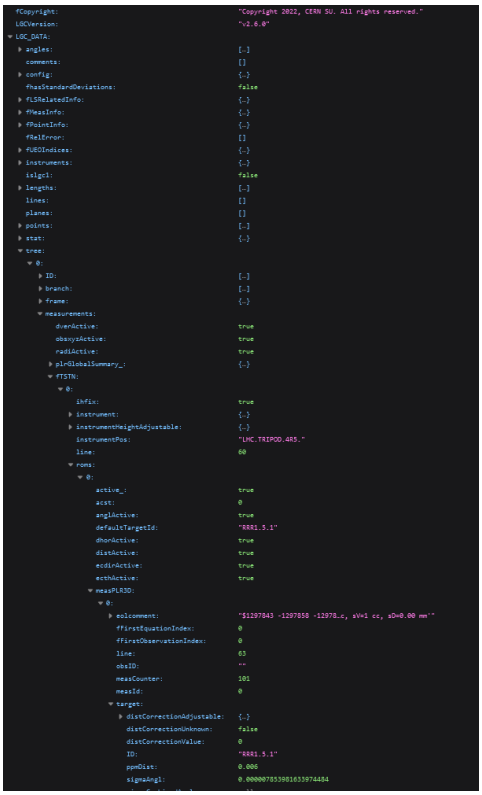

Fig. 2: Illustration of JSON complexity.

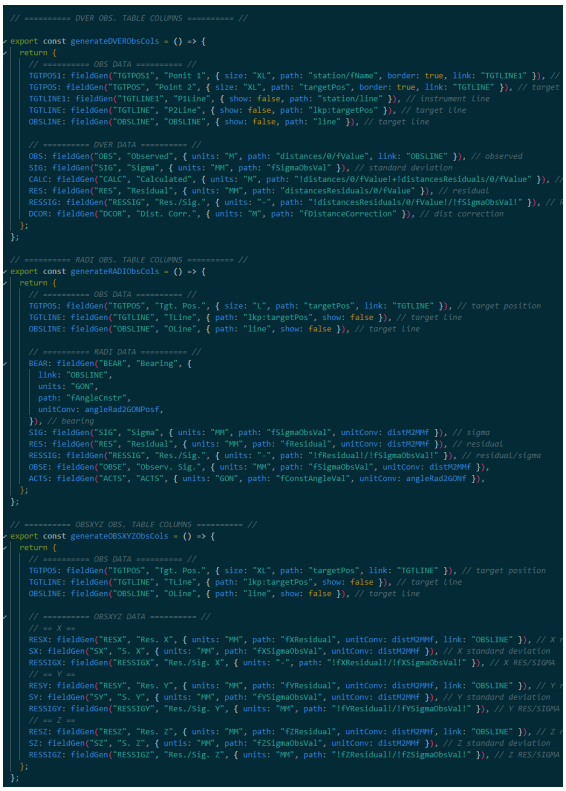

Fig. 3: Code strcture.

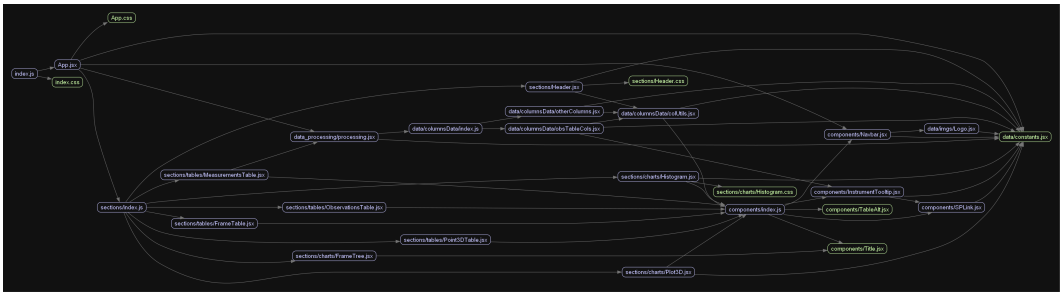

Fig. 4: Dependency structure of React.js codebase.

Section 4 Features Overview

This section briefly describes all the interactive features implemented in the LGC HTML report. All the parts can be explored in an interactive demo here.

4.1 Header

The header contains all the file and computation metadata, some of which are obtained directly from the .json file. Others have to be computed. Figure 5 shows an example of a header.

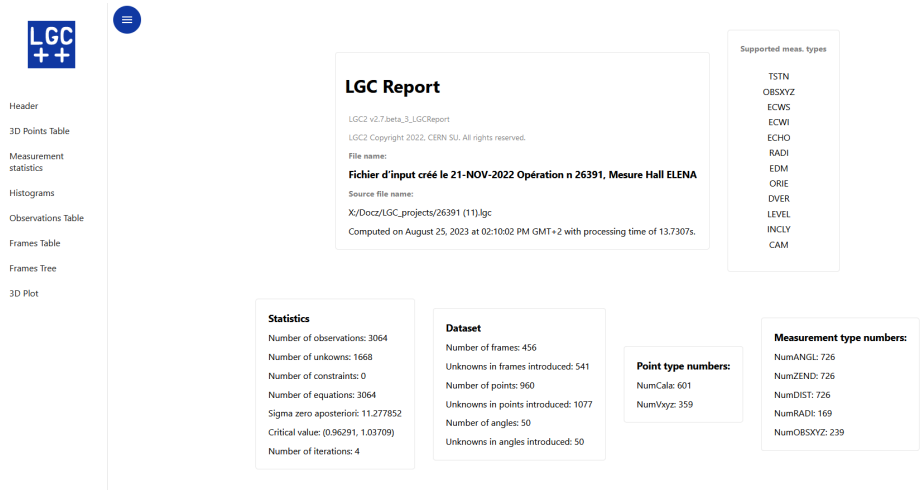

Fig. 5: Example of HTML report header.

4.2 3D points overview & Measurements overview

3D points overview contains all information about all the points introduced during the computation, whether target or station position. Again, some of them are obtained directly from the .json file. Others have to be computed using the notation shown in Section 2. An example of a table with 3D points can be seen in Figure 6. Note that the table supports many interactive features, such as multi-filtering, sorting, hiding columns, or just copying cells from the HTML table to Excel or .csv export. Also, the highlighted values in the Position row are links to the input file evocable using CTR+click. The measurements overview is very similar to the 3D points overview, capturing residual-related data after optimization in LGC, split into two tables for residuals in millimeters or milligons.

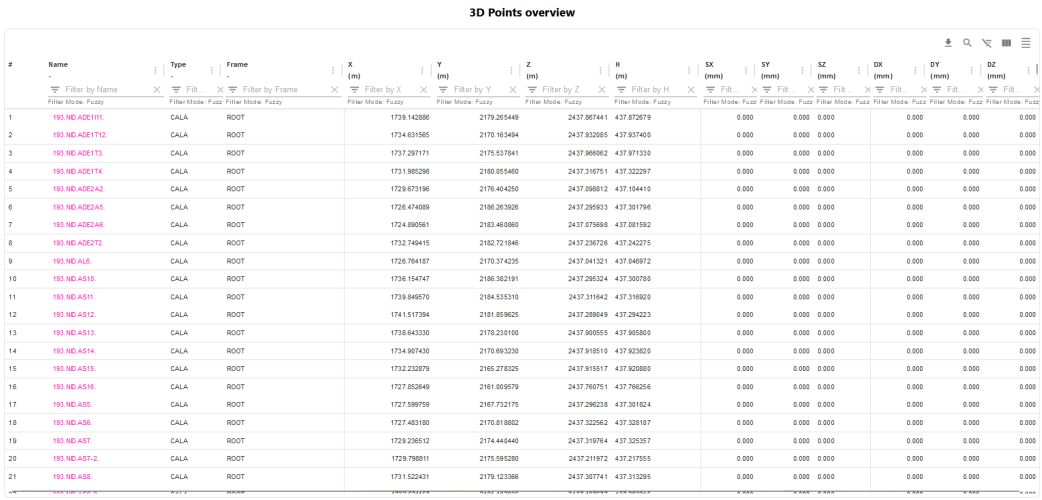

Fig. 6: Example of table of 3D points.

4.3 Histograms

The Histograms section includes histograms of residuals split by units for each measurement type used in the input. The number of bins is configurable by users. Users can also trigger stacking the data based on the source from which the observations were measured. And then turn some of them off, as seen in Fig. 7. The histogram also offers a list of observations in the given bin, and after CTRL+click, a user is taken to the line of this observation. The second part pictures normalized histograms of residuals merged from all measurement types normalized by their standard error, thus unit less.

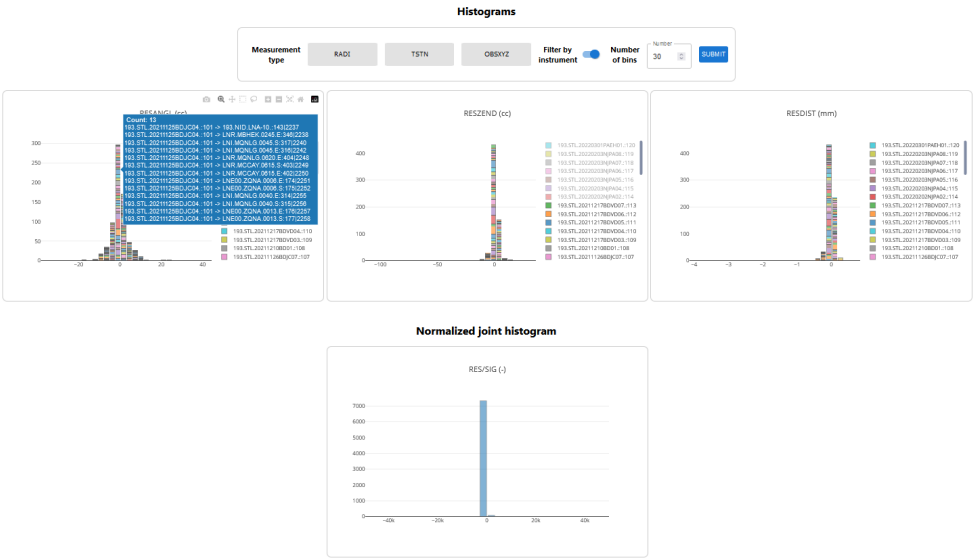

Fig. 7: Example of histograms section.

4.4 Observation overview

The observation overview section includes interactive tables of all observations for all measurement types used in the input table. Note that each measurement type has a unique set of columns and units, so the data must be separated into several tables. Besides regular highlighted links, this table also offers additional information about the instrument via a tooltip invoked by clicking on the instrument position in the table, as seen in Fig. 8.

4.5 Frames

The frame section has two parts, as there are two essential aspects of frames: transformations between frames and frame structure. The first aspect is exploited in the frames table, which has the same features as the previous tables. The structure is then shown in a dedicated interactive tree graph.

4.6 3D plot

The last part of the report is a 3D plot, which shows the users the 3D aspect of the introduced 3D points. This part was meant to capture more information, but as the 3D visualizations will be done in different software, it stayed only in a symbolic version.

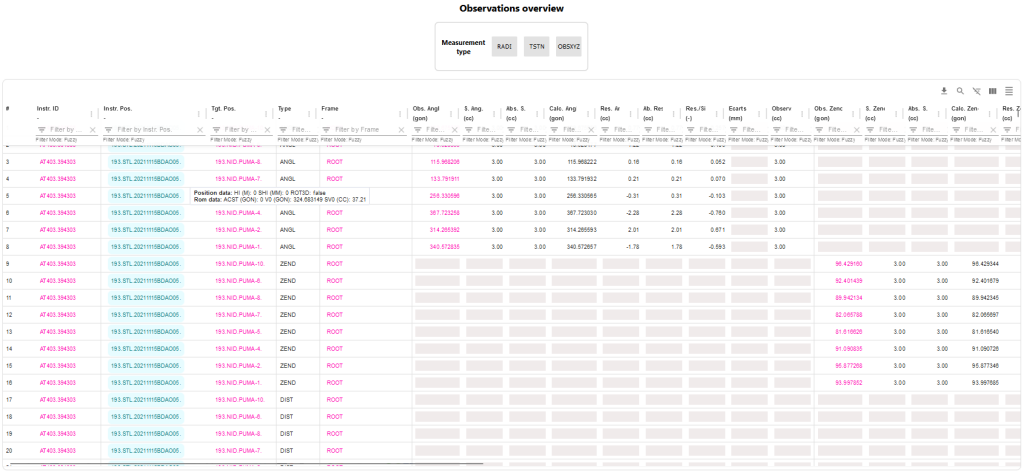

Fig. 8: Example of observation overview.

Section 5 Conclusion

There are still more features that could be added, but I had three user feedback session and the feedback has been positive. The result is a working version, which will be shipped with the next Survey Pad release.