Automated Method for Ruling out Spurious VGOS Cable Delay Data

L. V. Benkevitch

MIT Haystack Observatory, Westford MA

This document describes several programs intended for automated selection of the reliable cable delay calibration data in VGOS experiments.

1. select_bandpols.py
2. generate_pcmt.py
3. compare_pcmt.py
4. hist_pcmt.py

1. Introduction

The programs are Python scripts. They are usually run in the `IPython` environment. The command `%run` is used.

The VGOS stations have one-letter and two-letter abbreviations used in the described software and provided in the table below.

One-Letter	Two-Letter	Station Name
E	wf	Westford
F	w2	Westford2 (defunct)
G	gs	GGAO12M (Goddard)
H	k2	Kokee
V	ws	Wettzell
Y	yj	Yebes
I	is	Ishioka
S	oe	Onsala-Northeast
T	ow	Onsala-Southwest
M	mg	MGO (MacDonald)

There are four data transmission bands, `A, B, C` and `D`, each having two polarizations, `X` and `Y`, which yields total of eight channels called band-pols. They are denoted as `AX, AY, BX, BY, CX, CY, DX, DY`.

2. Script `select_bandpols.py`: Selection of Cable Delay Calibration data

The cable delays measured over a session as functions of time are intrinsically highly correlated processes. If for some reason a band-pol delay process shows poor correlation with other band-pols, it should be discarded as bad. This is a basis of the discriminator algorithm used in the main program, `select_bandpols.py`. For the eight band-pols, the `8x8`-dimensioned correlation matrix R is calculated, each element of which containing the Pearson correlation coefficient between i-th and j-th band-pol. The matrix is symmetric and has a unity diagonal. If all the band-pols are good, all the matrix has only values close to +1. However, if, say, the k-th band-pol data are spurious, then all the values in the k-th row (and in the k-th column) will be much lower than +1. Fig. 1 shows two examples of correlation matrices.

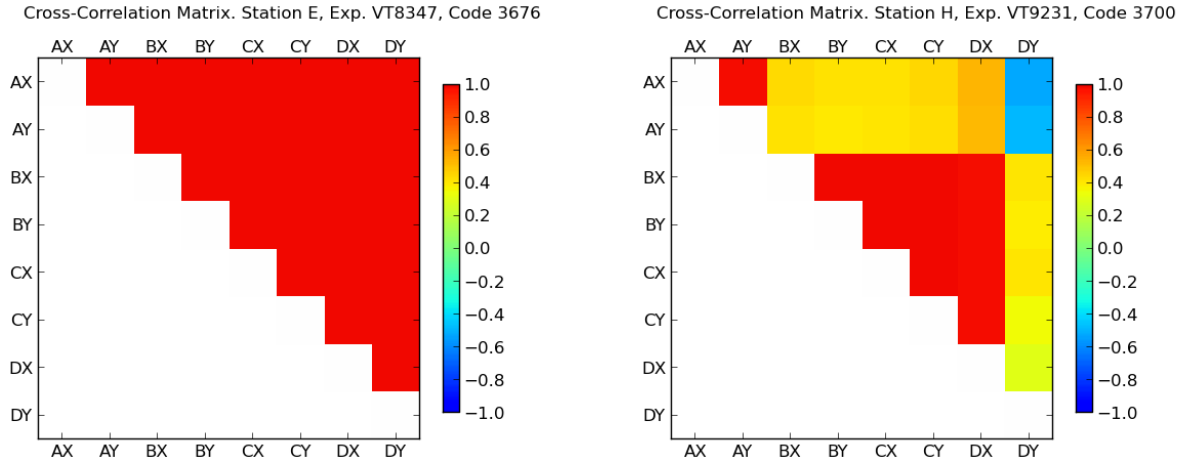

Fig. 1.— Visualization of the correlation matrices for two experiments, VT8347 on Station E (Westford) and VT9231 on Station H (Kokee). Since the correlation matrices are symmetric with the unities on the diagonal, only the upper triangle above the diagonal is shown. The matrix in the left panel contains the Pearson correlation coefficients very close to +1 because the band delay curves are highly correlated between each other. The right-panel correlation matrix demonstrates poor correlation of band-pols AX, AY, and DY with other band-pols.

Obviously, in Experiment VT9231 on Station H (Kokee) three band-pol channels, AX, AY, and DY have spurious band delay data and are to be rejected, because their columns (or rows, which is the same in a symmetric matrix) contain low correlations. A channel is discarded if the median of the column is below the specified "median threshold".

The script `select_bandpols.py` selects good band-pols whose rows in the correlation matrix have medians below the threshold (default value 0.5). The rows and columns of bad band-pols are iteratively removed from the matrix, improving the median values for other band-pols, until all medians are above the threshold. The correlation matrix, medians, and multiple correlation coefficients on each iteration (if any) are saved in text files.

Earlier, the coefficient of multiple correlation was considered as a candidate for the discriminator between good and bad band-pols. For each band-pol channel the multiple correlation with other seven was calculated. This parameter did not prove to be as reliable as the median of correlations of each with all others and is not used here. However, the coefficients of multiple correlation are calculated and printed for each channel.

The script can work on all the data (option -a) for one or several stations One can specify them in option `-s`, like `-s E, -s EGY`, or `-s IEHV`.

The script `select_bandpols.py` generates plots of the band-pols for one experiment on one station, or all the experiments on one or more stations. It computes medians for the rows (or columns) of the correlation matrix. The plots with with the correlation median below its threshold are marked as "Rejected".

The selected band-pols are saved in the file
`selections_st_<station-name(s)>_m<median-threshold>.txt`. The format of any line in the file `selections_st*.txt` allows its substitution in option `-s` of the script `pcc_select.py`.

Arguments:
`-m <threshold>`: threshold for the correlation median, -1 to 1 (default 0.5).
`-s <a station letter>`, like E, G, H ... (or in lower case, e, g, h ...).
`-d <pcc_datfiles directory>`, like `/data/geodesy/3686/pcc_datfiles`
`-o <output directory name>` where `.png` graphs and `.txt` logs are saved.
`-b`: show band-pol plot in X-window.
`-c`: show cross-correlation matrix.
`-n`: do not create plots and do not save them in .png files.
`-a`: Make `.png` plots and `.txt` files for all available data under directory in -d (like `-d /data/geodesy`). If more stations are given in -s, like `-s EY` or `-s VIGH`, only data for those stations are plotted and saved in -o directory. If -a is present, -b and -c are ignored (for too many windows would be opened).
-h print the help text.

Examples:

(1) Select good band-pols in experiment 3658 from station E. Save the band-pol plot in directory `pltE`. Save the correlation matrix plot (-c). Show both plots on the screen (-p). The process of successive selection of good band-pols is logged in text file `bandpol_E_3658_VT8204.txt`. The plots are saved as `bandpol_E_3658_VT8204.png` and
`xcorrmx_E_3658_VT8204.png`:

```
%run select_bandpols.py -s E -d /data/geodesy/3658/pcc_datfiles\jb/ \
                        -o pltE -b -c
```

(2) Select good band-pols in all the experimental data from station V located under directory

/data/geodesy/. Save the band-pol plots in directory pltV. Save the correlation matrix plots (-c). The key -b (show plots on the screen) is ignored when -a is used. The processes of successive selection of good band-pols are logged in text file `bandpol_V_<exp.code>_<exp.name>.txt`. The plots are saved as `bandpol_E_<exp.code>_<exp.name>.png` and `xcorrmx_E_<exp.code>_<exp.name>.png`. The diagnostic messages are logged in the file `diagnostics_st_E.txt`:

```
%run select_bandpols.py -s V -d /data/geodesy/ -o pltV -b -c -a
```

(3) Select good band-pols in all the experimental data from stations I and Y. Save the band-pol plots in directory `pltV`. Save the correlation matrix plots (-c). The processes of successive selection of good band-pols are logged in the text file `bandpol_<station>_<exp.code>_<exp.name>.txt`. The plots are saved as `bandpol_<station>_<exp.code>_<exp.name>.png` and `xcorrmx_<station>_<exp.code>_<exp.name>.png`. The diagnostic messages are logged in the file `diagnostics_st_IY.txt`:

```
%run select_bandpols.py -s IY -d /data/geodesy/ -o pltIY -c -a
```

(4) Fig. 2 shows an example of `select_bandpols.py` graphic output for the command:
`%run select_bandpols.py -s E -d /data/geodesy/3715/pcc_datfiles/ -o 3715 -m 0.98 -b -c`

3. Script `generate_pcmt.py`: Bulk Generation of PCMT Files

The selected good band delay data are averaged by the script `pcc_select.py`, which saves the results in PCMT data files. The script `generate\_pcmt.py` automates the use of selections generated by the script `select_bandpols.py`. It runs `pcc_select.py` in a loop for all the experiments mentioned in the file `selections_st_<station-name(s)>_m<median-threshold>.txt`.

Usage (two arguments are positional):
`%run generate_pcmt.py <band-pol-selections-file-name> <directory-to-save-pcmt>`:

Example:

```
%run generate_pcmt.py pltE_m0.97/selections_st_E_m0.97.txt pcmt_E_m0.97/
```

4. Assessment of validity of the automated band-pol selection using the PCMT files

The automated band-pol selection based on the correlation matrix needs tweaking of the column median threshold. Previously, the good band delay data were selected manually and averaged by the script `pcc_select.py`, and the results were saved in the PCMT data files. Assuming the manual selection provides the best results, the averaged automatically selected channels can be compared against those manually selected. As a criterion of proximity of the manual and automated results, the Root Mean Square (rms) of the difference between the averaged delay processes

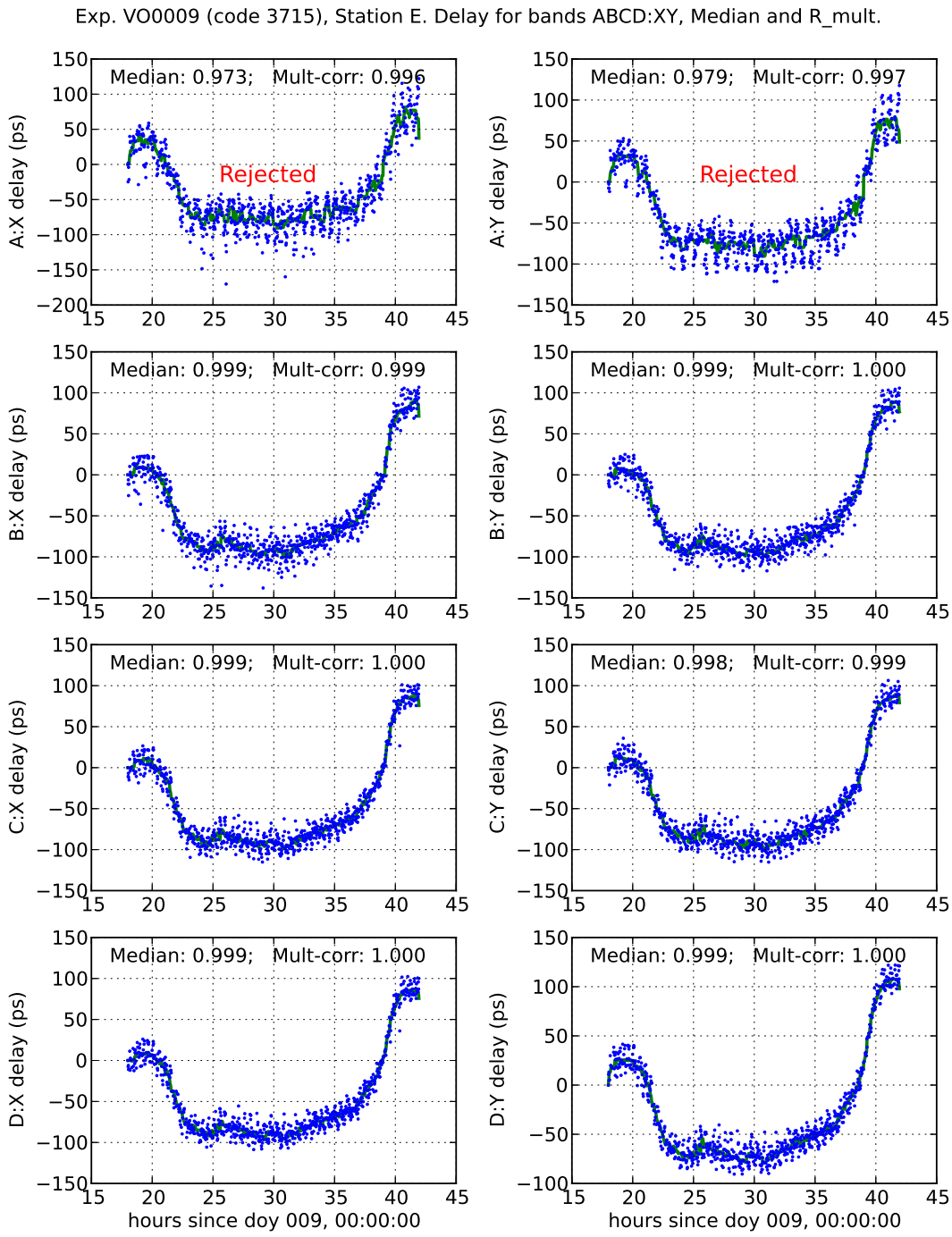

Fig. 2.— A plot of all the band-pol channels performed by `select_bandpols.py`. The channels A:X and B:Y were rejected because the median of cross-correlations with all other channels is less than 0.98. Another parameter, the multiple correlation coefficient (Mult-corr), is printed next to Median for comparison.

saved in the PCMT files has been chosen. Two scripts are written to visualize and assess goodness of the automated band-pol selections: `rms_pcmt.py` and `hist_pcmt.py`.

4.1. Script `rms_pcmt.py`:

This script compares two PCMT files. It is invoked with two command line parameters: the name of the PCMT file created with manual band-pol selections with that created with automated selection. It plots the averaged cable delays for both, their difference, and the difference with its trend removed. The difference trend is obtained as the result of its median filtration with the 21 point base. The standard deviation of the untrended difference is printed. The plot is saved on disk as a png file.

Example:

```
%run rms_pcmt.py /data/geodesy/3693/pcc_datfiles/vt9148gs.pcmt.BC.XY.dat \
                 pcmt_m0.90/VT9148gs.pcmt.BCD.XY.dat
```

An example of `rms_pcmt.py` plot is shown in Fig. 3.

4.2. Script `hist_pcmt.py`

This script makes bulk calculation of the standard deviations of the untrended differences for all the PCMT files available on the `demi.haystack.mit.edu` server under the directory `/data/geodesy/`. The names of the available PCMT files are read from the file `pcmt.txt`. For each of the available stations, the script plots 3x4=12 histograms of the rms difference between PCMT files. Each plot has 3x4=12 histograms for a station at the same median threshold value. The histograms are built for all the stations available. Also, it plots graphs of average RMS of difference (ps) vs median thresholds.

Note that the script `hist_pcmt.py` requires special format for the names of directories with the generated PCMT files. They, in turn, must be obtained with the use of the selection files for ALL the stations. Here is an example of preparing the data for `hist_pcmt.py` for one value of the median threshold, 0.95. This is performed by invoking the two previously described scripts, one after another:

```
%run select_bandpols.py -s EGHIVY -d /data/geodesy/ -o pltEGHIVYm0.95 -m 0.95 -a
%run generate_pcmt.py pltEGHIVYm0.95/selections_st_EGHIVY_m0.95.txt pcmt_m0.95
```

Thus, the PCMT data used by `hist_pcmt.py` must be in directories `pcmt_m0.xx`, where `xx` are the two-digit decimal hundredth of the median thresholds.

The first parameter on the command line is the output directory name. then the set of

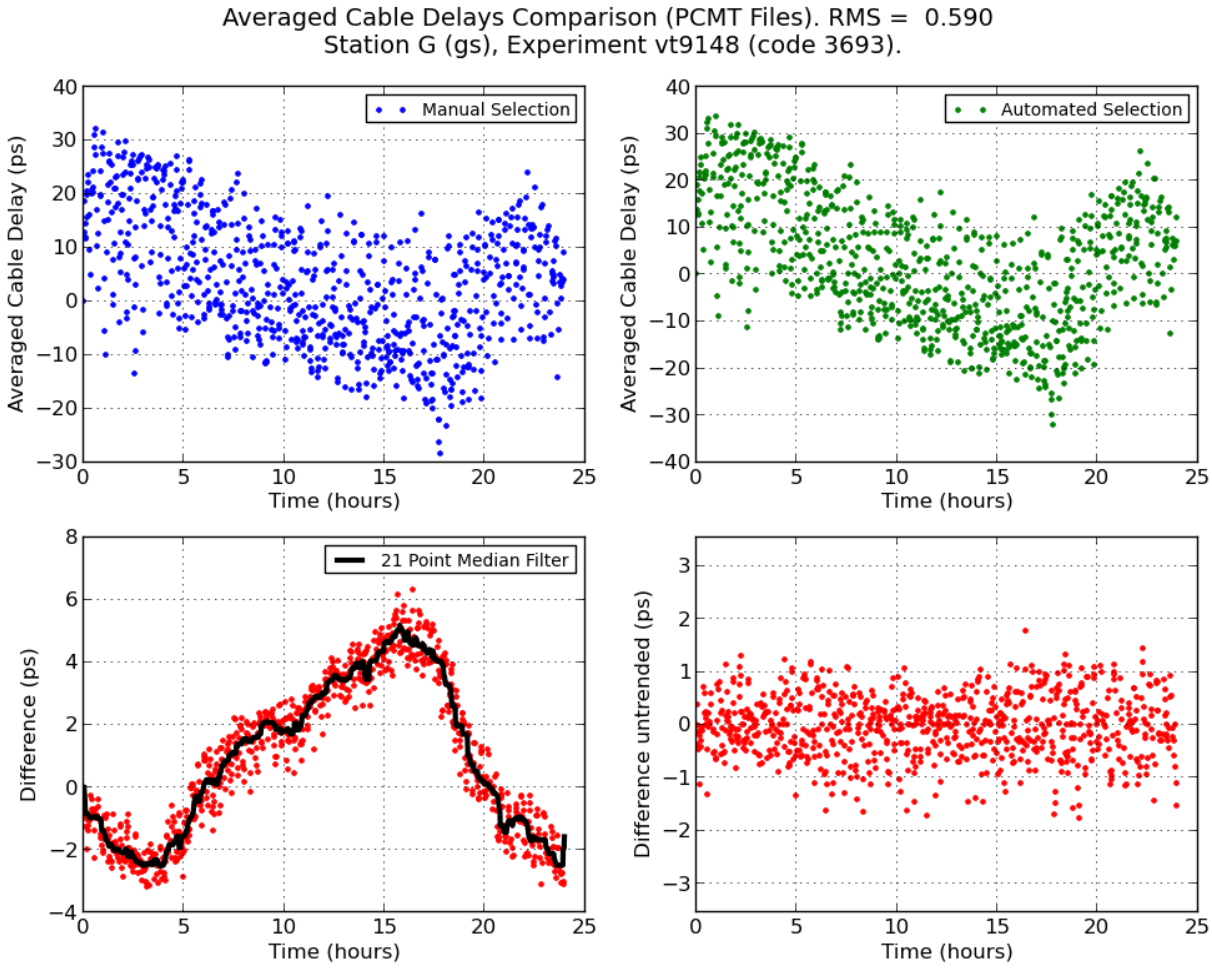

Fig. 3.— An example of `rms_pcmt.py` output.

thresholds is specified on the command line. The threshold values can be either numbers in [0..1] at the precision of two decimals after the dot or just 2-digit percent values. I.e., the following numbers are considered equivalent: 0.78 or .78 or 78

Example:

```
%run hist\_pcmt.py hist\_pcmt1 50 70 90 91 92 93 94 95 96 97 98 99
```

or, which is the same:

```
%run hist\_pcmt.py hists/ .5 0.7 90 0.91 .92 .93 .94 0.95 96 97 98 99
```

The script `hist_pcmt.py` plots the histograms for only first twelve median threshold values. However, the user can provide on the command line as many median values as there are directories `pcmt_m0.xx`. The main result, the average rms of the manual/automated differences versus the median threshold values, is plotted for every available station. The plot is saved in the file

`avg_rms_vs_thresh.png`. For each station the optimal median threshold value is found. This plot is shown in Fig. 4.

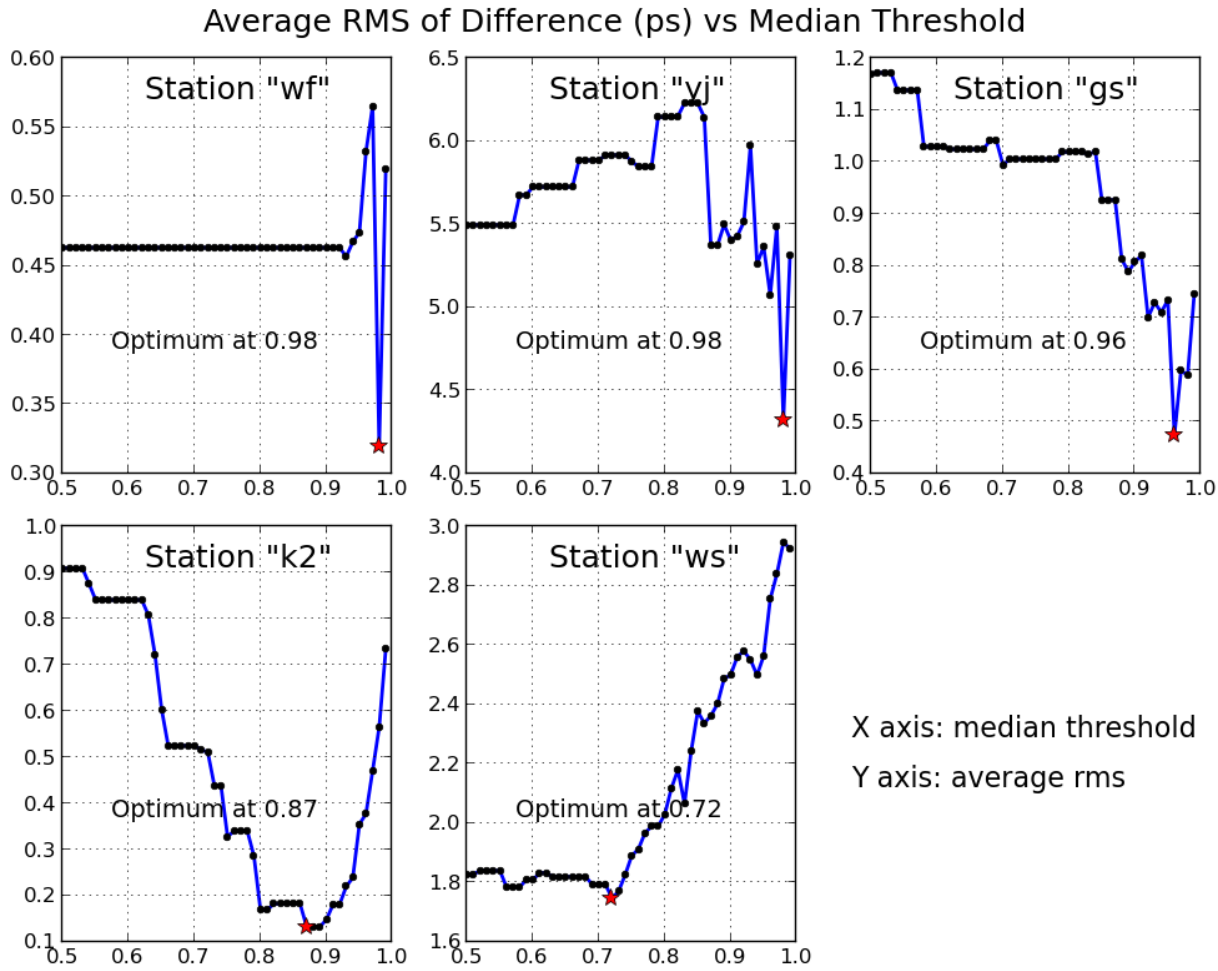

Fig. 4.— Results of the search for the best median thresholds for every station.

5. Appendix

The software is located at the github repository
`https://github.com/benkev/vgos`
The user can create a local copy at a disk location using the command
`git clone https://github.com/benkev/vgos.git vgos`

Currently, the software does not require compilation or installation.