Appendix H

Velest tools

Here I propose a set of python tools to use with the VELEST output parameters. There are four principal functions. The first (best_vpvs), reads the 'invers.out' output folder and detects the VELEST iteration that produced the lowest RMS errors and saves the new velocity model as 'input2.mod'. This new model can now be used for the next VELEST inversion. A copy of this model is moved to a new folder and named using the RMS value, such so, later on, it is easier to identify the model that produced the lowest RMS value. It additionally plots the new velocity model and respective DATVAR, SQRD and RMS, for comparison. The second function (plot_P_vms), plots all velocity models produced by the different VELEST inversions, including the initial, final and 1D-minimum models as it is shown in Fig. H.1. The third (plot_veloutdif), plots the differences (latitude, longitude, depth and origin time) between the original catalogue and after the VELEST inversion, considering the output file 'velout.dif' (Fig. H.2). The fourth function (quality_solutions) considers all VELEST runs and plots the RMS for each run (Fig. H.2). VELEST can be automatized using a combination of these python tools with bash scripting. An example is shown in Fig. H.3.

Source code:

https://github.com/veronica-antunes/PHD_GGB/blob/master/Velest_tools.py

Functions:

best_vpvs(infile = 'invers.out', outfile_vel = 'input2.mod', xlim = [0,10], vm_path = 'all_vm_models')

plot_P_vms(initial_models, min1D_model, final_models = None, all_models = '*.mod', xlim = [2,13], ylim = [-5,60], outfile = 'all_vms.png')
plot_veloutdif(velfile = 'velout.dif')

quality_solutions(invers_path)

Parameters:

infile (optional): VELEST output file with information about the current VELEST run and iteration process. Includes the VELEST input parameters and the different velocity models produced in each iteration. Default is 'invers.out';

outfile_vel (optional): name of the new velocity model after the VELEST run taken from the 'invers.out' file. Default is 'inpput2.mod';

xlim (optional): x limit values for plotting the new velocity model. Default is (0,10) or (2,13), depending on the function;

vm_path (optional): Path to save the new model that is named with the RMS value, in order to make it easier to decide the best model later on. All models from the different VELEST inversion should be saved in this path for later comparison. Default is 'all_vm_models';

initial_models: list of all initial models to be plotted. The algorithm will consider the name of these models in the legend but will discard the first two strings (this can be used to add a 'i_' in the beginning of the name in all initial models, in order to list them easier, e.g., using 'i_*');

min1Dmodel: name of the final 1D minimum velocity model, resulted from all VELEST inversions (the string rule described above does not apply here);

final_models (optional): list of all final models to be plotted. It applies the same string rule as in the initial_models (e.g., 'f_*'). Default is None (does not plot the final models);

all_models (optional): list of all velocity models to be plotted (e.g., the intermediate models). Default selects all velocity model files ('*.mod');

ylim (optional): y limit values for plotting. Default is (-5,60);

outfile (optional): name of the outfile plot. Default is 'all_vms.png';

velfile (optional): Velest output file name with information about the difference between the initial catalogue and the catalogue after the VELEST run. Default is 'velout.dif';

invers_path: Path of all 'invers.out' files from all inversions. The tool uses the date of the file creation to numerate the runs.

Necessary Packages:

- Obspy;
- Matplotlib;
- Numpy;
- Glob;
- Shutil.

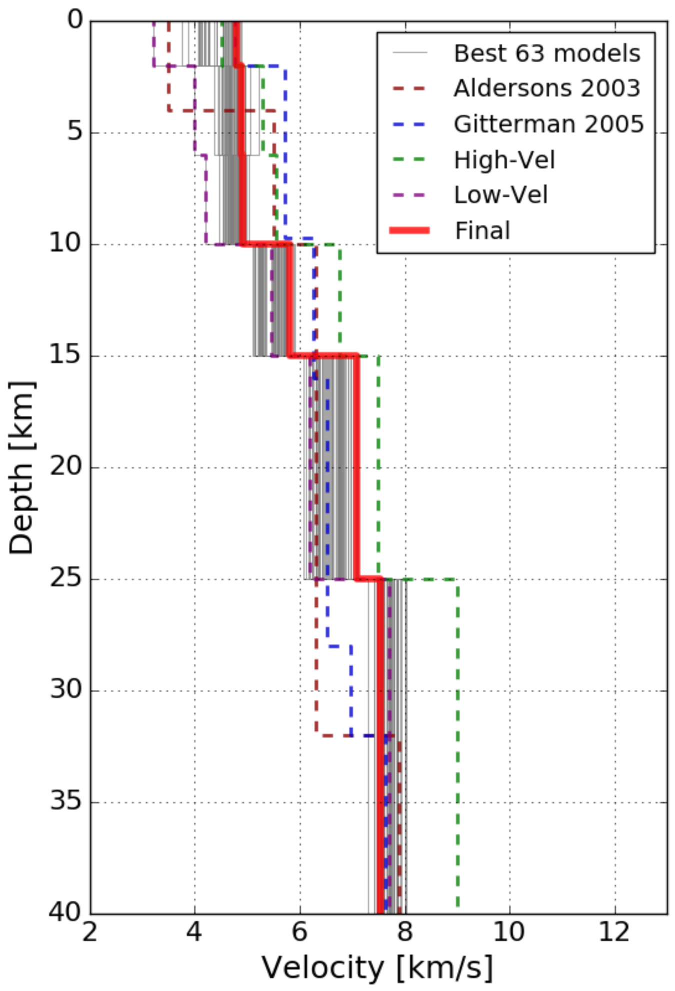
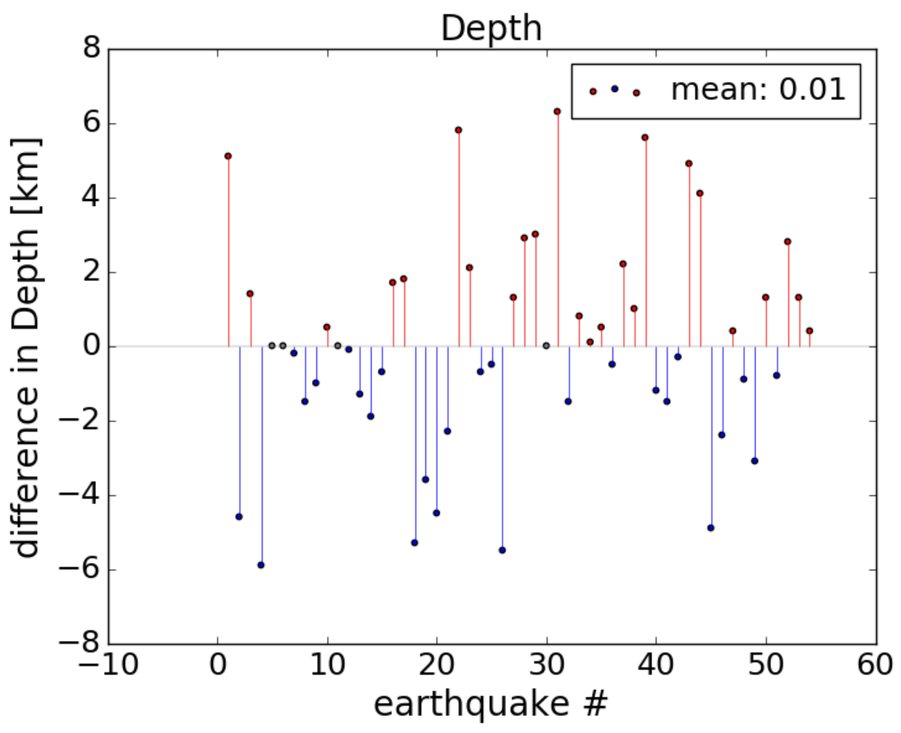


Figure H.1 – Example output for the plot_P_vms function.



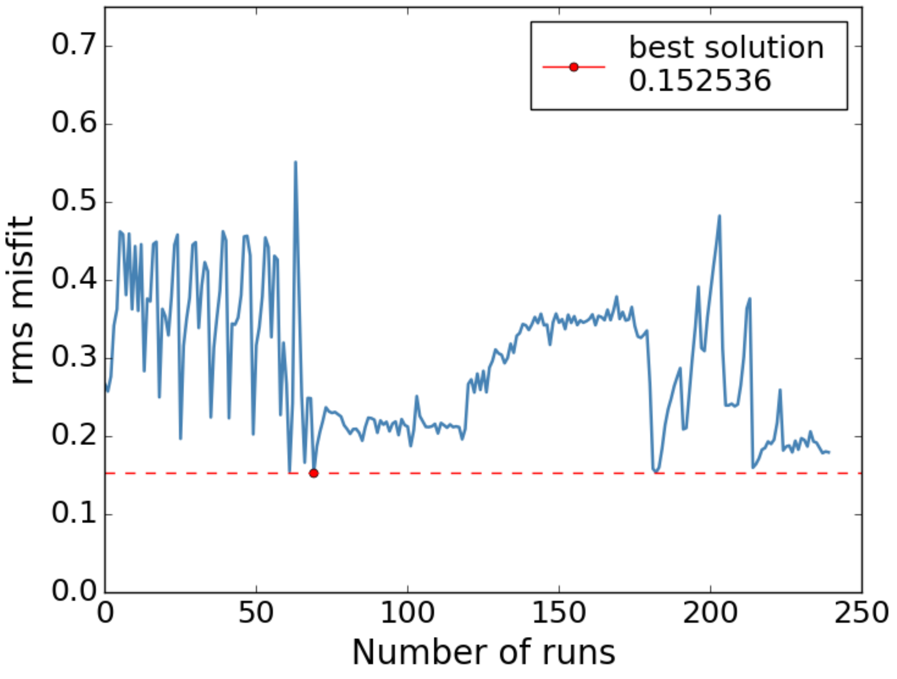


Figure H.2 – Example output for the plot_veloutdif (upper) and quality_solutions (bottom) functions.

```
#!/bin/bash

#Example of script to automatize VELEST using the python functions

n_runs=120

for i in {1..$n_runs}
do

        if [ -d $1 ]
        then
                echo "Directory already exists"
                exit 0
        fi

        velmenu << EOF
        velout.nor
        a
        y
        c
        q
        EOF

        wait

        echo 'checking best solution and plotting'
        #python check_best_solution.py & wait
         python <<< 'from Velest_tools import best_vpvs; best_vpvs()' & wait

        mkdir $1

        mv data.cnv $1/
        mv gmap.cur.kml $1/
        mv hypsum.out $1/
        mv print.out $1/
        mv velout.dif $1/
        mv data.nor $1/
        mv hypmag.out $1/
        mv input.mod $1/
        mv input.png $1/
        cp sta_cor.out station_true.sta
        mv sta_cor.out $1/
        cp velout.nor $1/
        mv fin_hyp.cnv $1/
        mv hyp.out $1/
        mv invers.out $1/
        mv ttt.ttt $1/

        cp select.out $1/
        cp selstat.lis $1/
        cp station.sta $1/
        cp station_true.sta $1/
        cp velest.cmn $1/

        mv input2.mod input.mod
        cp input2.png input_$1.png
        mv input2.png input.png

        wait
done
```

Figure H.3 – Example of script to automatize VELEST using bash scripting and some of the tools described above.