NOTE: this file contains a summary of the workflow for processing polar questions and continuation rises analysed in our project and presented in our papers. See README.docx for description of the folder contents.

This file contains instructions for the steps we followed for the curve fitting analysis of intonation in the Greek in Contact project. The instructions below assume that a number of .wav and Praat TextGrid pairs have been created. This process has been applied to the processing of intonation in three types of utterances (declaratives, polar questions, and continuation rises) in six language varieties (Athenian Greek, Cretan Greek, Asia Minor Greek, Corfiot Greek, Cypriot Greek, Venetian Italian, and Turkish). You can read some of the results published at https://greekincontact.phon.ox.ac.uk/research. Note that separate scripts, not included in this document, are necessary to complete some of the steps below, which will be pointed out when they are mentioned. The commands included below have been created to run in a Linux environment but should also work in a Mac terminal. The main data of the project have been deposited to https://doi.org/10.5287/bodleian:keKQbgg7Y.

Here is a summary of the steps we follow (see after the summary for more details for each step):

1. Process sound recordings in Praat to create pairs of .wav files and corresponding TextGrid files.

a. Each utterance should be in a separate wav file.

b. There should be three interval tiers in the TextGrid: orthographic transcription, translation, and interval of the nuclear stressed vowel.

2. Further process the wav and TextGrid files to prepare them for curve fitting.

a. Extract the labels and their timing from TextGrid tiers into plain text files (.lab xlabel files).

b. Extract f\_0 values every 10 ms from the wav files.

3. Correct pitch tracking errors such as pitch halving and pitch doubling.

4. Fit a 10th order polynomial to the corrected f\_0 of each input file.

5. Use the output of the 10th order fitting to define a smaller portion of the file, the Region of Interest (ROI), for further analysis.

6. Fit a 4th order polynomial to the Region of Interest.

7. Use the output of the 4th order fitting (polynomial coefficients) for statistical comparisons within and between dialect groups.

More details:

\*\* 1. Processing of sound recordings \*\*

The original sound files, obtained from various sources, come in many different audio formats. Bring all sound recordings to the same format. Steps to follow:

1.1. Use your favourite audio tools to end up with 16-bit, monophonic wav files, sampled at 16000 S/s.

1.2. Use your favourite Praat script or any other method to cut recordings which contain multiple utterances into small wav files, each containing one utterance.

1.3. In the filename of the cut wav files also include "d", "q" and "c" labels for "declarative", "question" and "continuation rise". For each language variety separate the cut files into three folders: declaratives, polars and continuation rises. The filenames and folder/directory names you use should preferably not contain any spaces or punctuation marks, other than ".", "-", "\_". They may contain Greek, Turkish, or accented letters, but NB these may require special treatment.

1.4. Use your favourite Praat script or any other method to automatically go through all the little wav files and create blank TextGrids for them. Our script created three interval tiers, from top to bottom: "orthographic", "translation", "main\_stress".

1.5. Manually complete the annotations in the tiers. For main\_stress the annotation may differ depending on the clause type and language variety. Place boundaries at the beginning and end of the stressed vowel in the nuclear word and mark the interval with "v" (=vowel).

\*\* 2. Prepare Praat TextGrid files for curve fitting \*\*

Steps to follow:

2.1. Convert the TextGrid files to plain text files using e.g. the commands below. Remember to use the appropriate folder name where your files are located to replace "athenian" (for example "turkish", "AMG", "italian", "cypriot"). The awk scripts tg2orthlab, tg2translab and tg2stresslab (in the folder "processing\_praat\_TextGrids") need to be copied in the working folder. They extract information from corresponding TextGrid tiers (i.e. the labels in the tier and each label’s timing information) to text files such as “filename.orth.lab”, “filename.trans.lab” etc. The commands only work properly if the TextGrid actually does have tiers labelled "orthographic", "translation", "main\_stress". To run them you need the AWK programming language. For Mac systems, make sure the latest version of the AWK programming language and Xcode software are installed.

For example,

for i in athenian/\*.TextGrid;

do

cat $i | awk -f tg2orthlab >$i.orth.lab;

cat $i | awk -f tg2translab >$i.trans.lab;

cat $i | awk -f tg2stresslab >$i.stress.lab;

done

and similarly for files in the other varieties, in their folders.

2.2. Alternatively, for filenames containing non-utf-8 characters, use this command:

for i in \*.TextGrid;

do

iconv -f "$(file -b --mime-encoding $i)" -t utf-8 $i | awk -f tg2stresslab > $i.stress.lab;

done

2.3. Check whether the number of annotations on each tier agree (remember to substitute the appropriate folder name where your files are located to replace "athenian"):

for i in athenian/\*.TextGrid; do wc -l $i.\*.lab; done >athenian\_labelcounts

2.4. Pull all the information from the little “stress.lab” files together into one .csv file where each row will refer to one file. This file will be used later in Step 5.1.:

for i in \*stress.lab; do cat $i | awk '{print "'$i'" , $0}'; done >vstresses.csv

2.5. Make sure the wav files are all sampled at 16000 S/s. Requires permissions to overwrite .wav files. You will also need to have sox installed in your computer (http:// sox.sourceforge.net/).

for i in \*.wav;

do

sox $i -r 16000 temp.wav;

cp temp.wav $i;

done

2.6. Extract the f\_0 contours. This is done in our lab using ESPS (Entropic Signal Processing System) commands (see http://www.phon.ox.ac.uk/releases for more information on ESPS), but other f0 estimation algorithms are available

for i in \*.wav;

do

sox $i -r 16000 $i.raw;

/opt/esps/bin/btosps -f 16000 -t SHORT -c "" $i.raw $i.sd;

/opt/esps/bin/get\_f0 $i.sd $i.f0;

/opt/esps/bin/pplain $i.f0 | awk '{print $1}' >$i.f0.txt;

done

2.7. Create a text file containing a list of all the wav files to be used later in Octave for the curve fitting.

ls \*.wav >wavlist.txt

\*\* 3. Correct pitch tracking errors \*\*

3.1. Convert the textfile with the list of wav files into wavlist.m which will be used in Octave (to install OCTAVE go to https://wiki.octave.org/Category:Installation. For Mac systems, older versions don't run correctly, but the latest version (5.1.0 in April 2020) seems to work.):

cat wavlist.txt | awk '{print "\"" $0 "\";"}' >wavlist.m

3.2. Lightly edit wavlist.m: add "list = [" at the start, and "];" at the end (for an example see wavlist.m files in any of the "wavlist\_startlist" folders)

3.3. In Octave, load the "wavlist.m" file and the script "fitf0st.m" (in the "octave\_scripts" folder--for Mac users, there is a problem with the fltk command, so comment it out. Also for Mac the figures created must be saved in .eps files instead of .ps files) to check and correct pitch halving errors.

3.4. This script produces figures (for examples look in any "tabulated\_csv\_pdf\_files" folder) with three lines superimposed: the observed f\_0 curve (blue solid), the modelled f\_0 curve (red dashed) and a green solid line showing the difference between the two in semitones. Green dashed horizontal lines above and below the three f\_0 curves show an octave error (a sudden jump of 12 semitones or thereabouts: it can be 12 semitones + some small change in f0)--see any errs.pdf files.

3.5. After running the script, an eps file will be created for each .f0.txt file, containing the figures mentioned in 4. Join the output figures into one big pdf file with (this works for Mac systems):

source ~/.profile

cpdf all.pdf \*.eps

3.6. Visually inspect each curve to identify pitch tracking errors i.e. apparent halving and doubling errors leading to step-like deviations from the f\_0 contour.

3.7. Manually correct pitch doublings and pitch halvings in the f0.txt file of each curve by halving or doubling, accordingly, the incorrect f\_0 values. All potential octave jumps should individually inspected to check that they are octave errors, not actually-occurring large pitch changes; the error portions should then be corrected by doubling or halving (as appropriate) the f0 measurements in the ".wav.f0.txt" files throughout the mis-tracked interval. Other large or sudden jumps in f0 should be similarly inspected to determine their cause. Rapid pitch perturbations due to consonantal onsets or offsets can be retained unaltered, as they are a natural part of f0 contours. In other cases, rapid jumps could be attributed to environmental noises or to other speakers; such portions can be zeroed out in the measured f0 tracks.

\*\* 4. Fit 10th order polynomials \*\*

4.1. In Octave, use the same "wavlist.m" as in the previous steps to load the file list.

4.2. Access, load and run the Octave script ("fitf0.m"). The script needs to be in the directory you're working on, or if not, then you need to access it from within Octave.

4.3. This script fits 10th order polynomials to the f\_0 of each input file. Turning points (minima and maxima) of the 10th order polynomials are obtained automatically, by differentiation, and fitted to the corrected f\_0 tracks.

4.4. The outputs of the fitting are:

4.4.1. Figures for each wav file that is processed (examples in "tabulated\_csv\_pdf\_files")

4.4.2. A "maxmin.csv" file which contains 11 fields in columns: filename, mean\_f\_0 (in Hz) for the whole utterance, and the 9 maxima and minima times (see maxim files in "tabulated\_csv\_pdf\_files")

4.5. Join the output figures into one big file:

rm allfigs.ps

cat \*wav.ps | gs -dNOPAUSE -dBATCH -sDEVICE=ps2write -sOutputFile=allfigs.ps -

ps2pdf allfigs.ps

\*\* 5. Define Region of Interest \*\*

We define the relevant portion of the end of each utterance to be used for the analysis, called the Region of Interest, ROI. The ROI may be different depending on the analysis adopted for each tune. In our project, the beginning of ROI is set at the start of the nuclear vowel. Also we calculate the alignment of significant turning points, Ls and/or Hs, with respect to the nuclear vowel.

5.1. The start of the ROI is defined as the starting time of the nuclear vowel, which can be found in "vstresses.csv" created in Step 2.3.

5.2. The location of the final peak and/or the minimum before the final peak can be found in "maxmin.csv".

5.3. To do this, save "maxmin.csv" as maxmin.ods (or any other spreadsheet editor), highlight the desired times (the final peak, the low or both) and edit out the other turning point times. Then save maxmin.ods as startlist.csv using space as delimiter and add a column with the vowel start time.

5.4. In the analysis for polar questions, for example, startlist.csv has 4 fields: filename, mean\_f0, final trough (L) time (selected from the "maxmin.csv" file) and vowel start time.

\*\* 6. Fit 4th order polynomials \*\*

For this step we fit a 4th order (or a cubic) polynomial to the f\_0 of the ROI. For each dialect under analysis we follow these steps.

6.1. Make an .m file (parallel to "wavlist.m") for the parameters in startlist.csv:

cat startlist.csv | awk '{print $2 " " $4 ";"}' >startlist.m

6.2. Lightly edit "startlist.m" to add "params = [" at the start and "];" at the end [INSERT IMAGE].

6.3. Load "wavlist.m" and "startlist.m" in Octave.

6.4. Run "refitf0\_3rd\_or\_4th\_order.m" (in "octave\_scripts" folder) to get (as output):

6.4.1. Polynomial coefficients for the tune under analysis in "aparams.csv" and "cparams.csv" (the latter is orthogonalized)

6.4.2. Figures for each wav file that is processed

6.5. Join the output figures into one big file (for examples look in any "tabulated\_csv\_pdf\_files" folder):

rm allfigs.ps

cat \*wav.ps | gs -dNOPAUSE -dBATCH -sDEVICE=ps2write -sOutputFile=allfigs.ps -

ps2pdf allfigs.ps

\*\* 7. Use polynomial coefficients for comparisons \*\*

In this final step, the numeric outputs of the 4th order fitting can be used for various comparisons. Some are illustrated below. You can read more about the various methods we have followed in our published papers in the address given above. The polynomial coefficients resulting from the analysis for each dialect describe the shape of the f\_0 contours. Comparisons of the coefficients thus reveal differences in the shape of the f\_0 curves between dialects.

7.1. Check that "aparams.csv" and "cparams.csv" are separated by spaces, not commas.

7.2. Save the numeric coefficients from aparams.csv:

cat aparams.csv | awk '{print $2, $3, $4, $5, $6, $7, $8, $9}' >aparams.txt

cat cparams.csv | awk '{print $2, $3, $4, $5, $6, $7, $8, $9}' >cparams.txt

7.3. Here is information about each field:

$2 to $6 are the polynomial coefficients (from quartic to constant).

$7 is utterance mean f\_0.

$8 is peak f\_0 (in Hz).

$9 is peak height, in semitones above the utterance mean.

7.4. The lowest polynomial coefficient models the average pitch, and then in increasing order, the slope of the data, the curve as a parabola, the curve as a wave shape and as a more complex wave shape. If the sign is negative, the patterns are flipped about the horizontal axis.

7.5. In addition to the statistical analysis of the polynomial coefficients, a selection of the parameters/coefficients can be plotted in Octave or in a statistics package (see "stats" folders within the "Athenian\_AsiaMinorGreek\_Turkish\_comparison" and "Athenian\_Cretan\_Venetian\_comparison" folders).