Documentation of data processing of *Modelling electroglottographic data with wavegrams and generalised additive mixed models* (Pilot study)

Stefano Coretta

2019-05-16

1 Wavegram analysis

This script extracts the wavegram data of the dEGG signal.

```
<<<script header>>>

<<<preamble w>>>

<<<main loop w>>>

<<<smoothing>>>
```

The preamble defines a few settings for filtering and smoothing, and the results file.

```
lower = 40
upper = 10000
smoothWidth = 11
results$ = "../data/datasets"
createDirectory(results$)
data$ = "../data/raw"
resultsHeader$ = "file,token,time,sequence,sample,amplitude"
resultsFile$ = "'results$'/wavegram.csv"
writeFileLine: resultsFile$, resultsHeader$
fileList = Create Strings as file list: "fileList", data$
numberOfFiles = Get number of strings
```

Each file is read and the voiced/unvoiced intervals in the EGG channel are detected with `To TextGrid (vuv)`. The signal is pass-band filtered before detection to remove hardware high-frequency noise.

```
#### Files loop ####
for file from 1 to numberOfFiles
    selectObject: fileList
    fileName$ = Get string: file
    fileBareName$ = fileName$ - ".wav"
    sound = Read from file: "'data$'/'fileName$'"
    sound2 = Extract one channel: 2
    # signal is inverted when recorded
    Multiply: -1
    Filter (pass Hann band): 100, 0, 100
    pointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "no", "yes"
    textGrid = To TextGrid (vuv): 0.02, 0.001
    numberOfIntervals = Get number of intervals: 1

    <<<vowel loop w>>>
```

```
endfor
```

The script loops through each vowel in the signal and extracts measurements from a 500 ms window around the midpoint (relative to the voicing interval). The loop first calculates the derivative of the EGG signal (the dEGG) and then loops through the each glottal cycle to find the time of dEGG maximum and minimum.

```
#### Vowel loop ####
token = 0
for interval to numberOfIntervals
    selectObject: textGrid
    intervalLabel$ = Get label of interval: 1, interval
    if intervalLabel$ == "V"
        token += 1
        start = Get start time of interval: 1, interval
        end = Get end time of interval: 1, interval
        vowelDuration = end - start
        midPoint = start + (vowelDuration / 2)
        # Warning: The following two lines are easily breakable
        selectionStart = midPoint - 0.05
        selectionEnd = midPoint + 0.05
        selectObject: sound2
        selection = Extract part: selectionStart, selectionEnd, "rectangular",
            ...1, "yes"

        <<<degg w>>>

        <<<period loop>>>

        removeObject: selection
    endif
endfor
```

The following chunk defines the dEGG calculation procedure. Before calculating the dEGG, the EGG signal is smoothed with a moving average function with smooth width 11 (time lags are adjusted by shifting the raw time by the lag). A PointProcess object is created, which will be used for the detection of the start of the glottal cycles. The calculated dEGG is smoothed again with the same moving average and time lag fix. The peaks in the dEGG signal are detected with `To point process (periodic, peaks)`. These correspond to dEGG maxima.

```
eggSmooth = Filter (pass Hann band): lower, upper, 100
@smoothing: smoothWidth
sampling_period = Get sampling period
time_lag = (smoothWidth - 1) / 2 * sampling_period
Shift times by: time_lag
Rename: "egg_smooth"
eggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"

selectObject: eggSmooth
deggSmooth = Copy: "degg_smooth"
Formula: "self [col + 1] - self [col]"
@smoothing: smoothWidth
sampling_period = Get sampling period
time_lag = (smoothWidth - 1) / 2 * sampling_period
Shift times by: time_lag
deggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
```

The EGG PointProcess object is looped through, and the time of each EGG minimum is obtained. The glottal period (glottal cycle duration) is calculated as the time between two consecutive EGG minima.

```
selectObject: eggPointProcess
eggPoints = Get number of points
meanPeriod = Get mean period: 0, 0, 0.0001, 0.02, 1.3

sequence = 0

for point to eggPoints - 2
    selectObject: eggPointProcess
    point1 = Get time from index: point
    point2 = Get time from index: point + 1
    point3 = Get time from index: point + 2
    selectObject: eggSmooth
    eggMinimum1 = Get time of minimum: point1, point2, "Sinc70"
    eggMinimum2 = Get time of minimum: point2, point3, "Sinc70"
    period = eggMinimum2 - eggMinimum1

    <<<wavegram>>>

    sequence = sequence + 1
endfor
```

For each glottal cycle, the normalised amplitude of the dEGG signal is extracted every 10 samples (with a sampling frequency of 44100 Hz we can obtain around 40 samples per cycle). Normalisation of amplitude and sample time is achieved through unity-based rescaling (range 0-1). If the glottal period duration is greater than twice the duration of the average period, that period is not analysed.

```
if period <= meanPeriod * 2
    selectObject: deggSmooth
    minAmplitude = Get minimum: eggMinimum1, eggMinimum2, "Sinc70"
    maxAmplitude = Get maximum: eggMinimum1, eggMinimum2, "Sinc70"

    sampleStart = Get sample number from time: eggMinimum1
    sampleEnd = Get sample number from time: eggMinimum2
    numberOfSamples = sampleEnd - sampleStart
    sample = sampleStart

    timeNorm = (eggMinimum1 - selectionStart) /
        ...(selectionEnd - selectionStart)

    while sample <= sampleEnd
        amplitude = Get value at sample number: 1, sample

        amplitudeNorm = (amplitude - minAmplitude) /
            ...(maxAmplitude - minAmplitude)

        sampleNorm = (sample - sampleStart) /
            ...(sampleEnd - sampleStart)

          # At sample rate 44100 Hz, each period has around 400 samples.
          # Extract data from every 10 samples (around 40 samples per cycle)
          # to reduce data size.
          sample = sample + 10

        resultLine$ = "'fileBareName$','token','timeNorm','sequence','sampleNorm','amplitudeNorm'"

        appendFileLine: resultsFile$, resultLine$
    endwhile
endif
```

2 Tracegram of modal and breathy phonated vowels

The following script extracts the dEGG maximum and minimum trajectories from the analysed portion of the vowel tokens. This is a ligthweight alternative to wavegram data. The dEGG maximum and minimum can be plotted in time for a condensed visual representation of changing glottal activity.

```
<<<script header>>>

<<<preamble t>>>

<<<main loop t>>>

<<<smoothing>>>
```

The preamble sets a few variables for reading the files.

```
lower = 40
upper = 10000
smoothWidth = 11
results$ = "../data/datasets"
createDirectory(results$)
data$ = "../data/raw"
resultsHeader$ = "file,token,time,egg_minimum,degg_maximum,degg_minimum"
resultsFile$ = "'results$'/tracegram.csv"
writeFileLine: resultsFile$, resultsHeader$
fileList = Create Strings as file list: "fileList", data$
numberOfFiles = Get number of strings
```

Each file is read and the voiced/unvoiced intervals in the EGG channel are detected with `To TextGrid (vuv)`. The signal is pass-band filtered before detection to remove hardware high-frequency noise.

```
for file to numberOfFiles
  selectObject: fileList
  fileName$ = Get string: file
  fileBareName$ = fileName$ - ".wav"
  sound = Read from file: "'data$'/'fileName$'"
  sound2 = Extract one channel: 2
  Multiply: -1
  Filter (pass Hann band): 100, 0, 100
  pointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "no", "yes"
  textGrid = To TextGrid (vuv): 0.02, 0.001
  numberOfIntervals = Get number of intervals: 1

  <<<vowel loop t>>>
endfor
```

The script loops through each vowel in the signal and extracts measurements from a 500 ms window around the midpoint (relative to the voicing interval). The loop first calculates the derivative of the EGG signal (the dEGG) and then loops through the each glottal cycle to find the time of dEGG maximum and minimum.

```
token = 0
for interval to numberOfIntervals
  selectObject: textGrid
  intervalLabel$ = Get label of interval: 1, interval
  if intervalLabel$ == "V"
    token += 1
    start = Get start time of interval: 1, interval
    end = Get end time of interval: 1, interval
    vowelDuration = end - start
    midPoint = start + (vowelDuration / 2)
```

```
    # The following two lines are easily breakable
    selectionStart = midPoint - 0.25
    selectionEnd = midPoint + 0.25
    selectObject: sound2
    selection = Extract part: selectionStart, selectionEnd, "rectangular",
      ...1, "yes"

    <<<degg t>>>

    <<<tracing loop>>>

    removeObject: selection
  endif
endfor
```

The following chunk defines the dEGG calculation procedure. Before calculating the dEGG, the EGG signal is smoothed with a moving average function with smooth width 11 (time lags are adjusted by shifting the raw time by the lag). A PointProcess object is created, which will be used for the detection of the start of the glottal cycles. The calculated dEGG is smoothed again with the same moving average and time lag fix. The peaks in the dEGG signal are detected with `To point process (periodic, peaks)`. These correspond to dEGG maxima.

```
eggSmooth = Filter (pass Hann band): lower, upper, 100
@smoothing: smoothWidth
sampling_period = Get sampling period
time_lag = (smoothWidth - 1) / 2 * sampling_period
Shift times by: time_lag
Rename: "egg-smooth"
eggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"

selectObject: eggSmooth
deggSmooth = Copy: "degg-smooth"
Formula: "self [col + 1] - self [col]"
@smoothing: smoothWidth
sampling_period = Get sampling period
time_lag = (smoothWidth - 1) / 2 * sampling_period
Shift times by: time_lag
deggPointProcess = noprogress To PointProcess (periodic, peaks): 75, 600, "yes", "no"
```

The EGG PointProcess object is looped through, and the time of each EGG minimum is obtained. The glottal period (glottal cycle duration) is calculated as the time between two consecutive EGG minima. For each glottal cycle, the dEGG maximum and minimum are found. If the period is greater than twice the duration of the average period, that period is not analysed.

```
selectObject: eggPointProcess
eggPoints = Get number of points
meanPeriod = Get mean period: 0, 0, 0.0001, 0.02, 1.3

for point to eggPoints - 2
  selectObject: eggPointProcess
  point1 = Get time from index: point
  point2 = Get time from index: point + 1
  point3 = Get time from index: point + 2
  selectObject: eggSmooth
  eggMinimum1 = Get time of minimum: point1, point2, "Sinc70"
  eggMinimum2 = Get time of minimum: point2, point3, "Sinc70"
  period = eggMinimum2 - eggMinimum1

  if period <= meanPeriod * 2
```

```
    selectObject: deggPointProcess
    deggMaximumPoint1 = Get nearest index: eggMinimum1
    deggMaximum = Get time from index: deggMaximumPoint1

    if deggMaximum <= eggMinimum1
      deggMaximum = Get time from index: deggMaximumPoint1 + 1
    endif

    selectObject: deggSmooth
    deggMinimum = Get time of minimum: deggMaximum, eggMinimum2, "Sinc70"

    deggMaximumRel = (deggMaximum - eggMinimum1) / period
    deggMinimumRel = (deggMinimum - eggMinimum1) / period

    time = (eggMinimum1 - selectionStart) / (selectionEnd - selectionStart)

    resultLine$ = "'fileBareName$','token','time','eggMinimum1',
      ...'deggMaximumRel','deggMinimumRel'"

    appendFileLine: resultsFile$, resultLine$
  endif
endfor
```

The following code defines the moving average smoothing function.

```
procedure smoothing : .width
  .weight = .width / 2 + 0.5

  .formula$ = "( "

  for .w to .weight - 1
    .formula$ = .formula$ + string$(.w) + " * (self [col - " +
        ...string$(.w) + "] + self [col - " + string$(.w) + "]) + "
  endfor

  .formula$ = .formula$ + string$(.weight) + " * (self [col]) ) / " +
    ...string$(.weight ^ 2)

  Formula: .formula$
endproc
```

3 Script header

The following is the header of the scripts.

```
########################################
# This is a script from the project 'Vowel duration and consonant voicing: An
# articulatory study', Stefano Coretta
########################################
# MIT License
#
# Copyright (c) 2016 Stefano Coretta
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
```

```
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#######################################
```