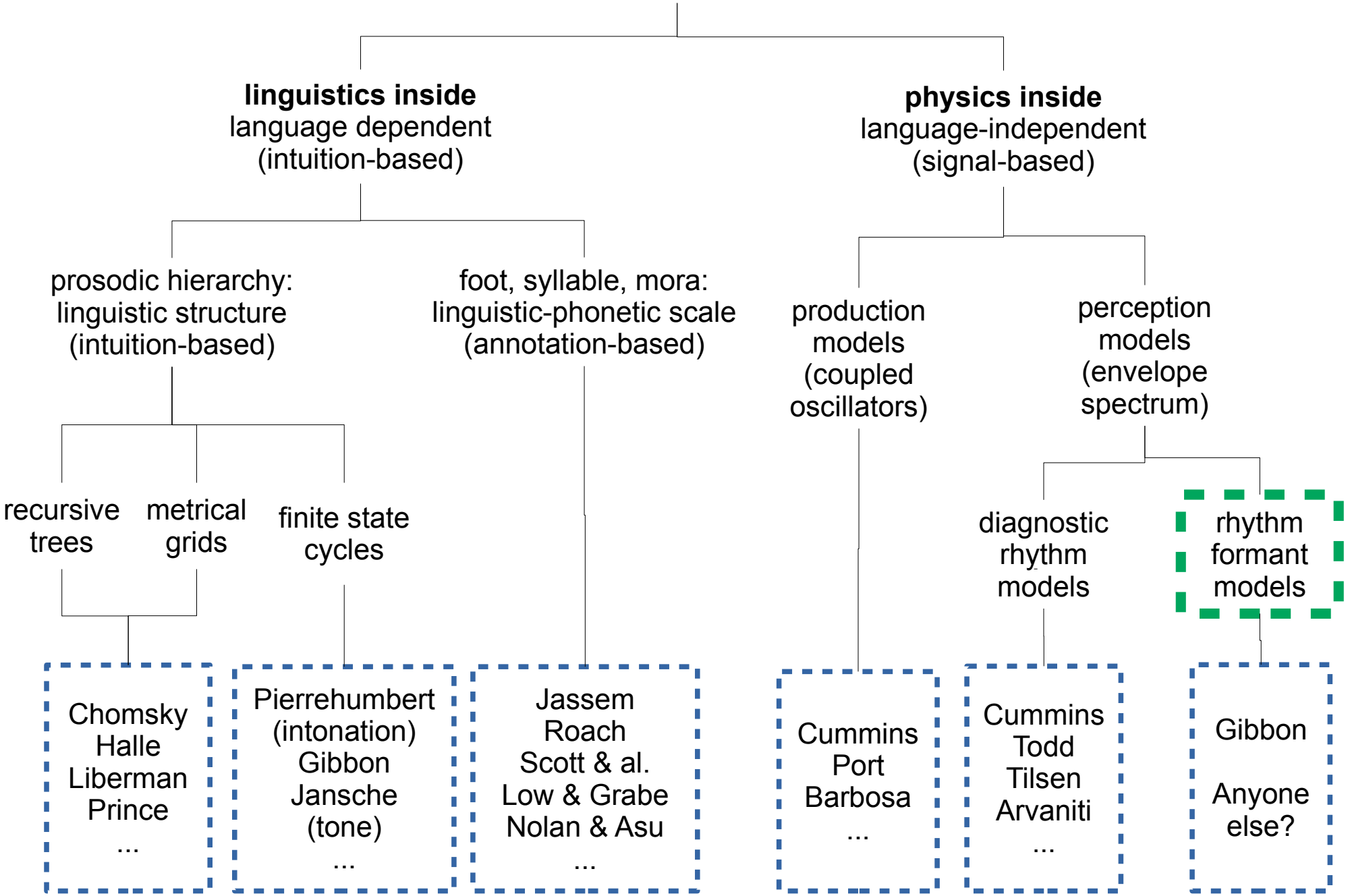
Software Tools
for
Rhythm Formant Analysis

Dafydd Gibbon

Bielefeld University, Germany
Jinan University, Guangzhou, China

Typology of Rhythm Description Frameworks

Typology of Rhythm Description Frameworks



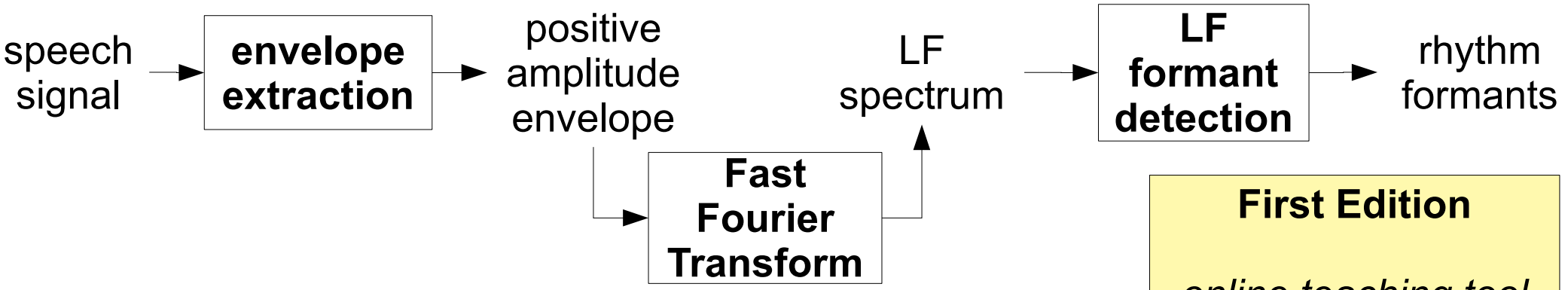
Adding a Rhythm Theory and an Analysis Method

As part of a Rhythm Perception Theory?

Rhythm Formant Analysis Tool Specification

Requirements: Rhythm Formant Theory and Rhythm Formant Analysis method
Properties: rhythm as oscillation, multiple variable rhythms, asymmetry

Design:
Low Frequency amplitude envelope spectrum maxima derived with
1) Hilbert Transform or Peak Picking → *amplitude modulation envelope*
2) Fourier Transform and spectrum → *low frequency spectrum*
3) Rhythm Candy Bars → *highest magnitude frequencies*
4) Rhythm Formant pattern → *histogram-like representation*



Implementation:
Python 3.n with Numpy, SciPy, MatPlotLib, PyAudio, TkInter
Single applications with individual configuration files.

Evaluation:
1) Test bench comparison with isochrony metric heuristics
2) Case studies on pragmatic interpretation of rhythm variation

First Edition

online teaching tool CRAFT

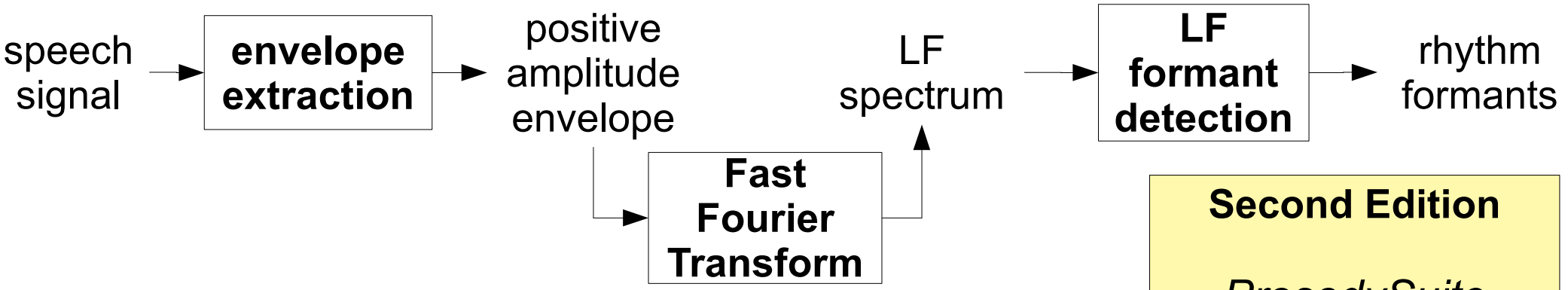
Localhost CRAFT

Remotehost CRAFT

Rhythm Formant Analysis Tool Specification

Requirements: Rhythm Formant Theory and Rhythm Formant Analysis method
Properties: rhythm as oscillation, multiple variable rhythms, asymmetry

Design:
Low Frequency amplitude envelope spectrum maxima derived with
1) Hilbert Transform or Peak Picking → *amplitude modulation envelope*
2) Fourier Transform and spectrum → *low frequency spectrum*
3) Rhythm Candy Bars → *highest magnitude frequencies*
4) Rhythm Formant pattern → *histogram-like representation*



Implementation:
Python 3.n with Numpy, SciPy, MatPlotLib, PyAudio, TkInter
Single applications with individual configuration files.

Evaluation:
1) Test bench comparison with isochrony metric heuristics
2) Case studies on pragmatic interpretation of rhythm variation

Second Edition

ProsodySuite

CLI / GUI, Python3

Teaching & Research in China and Iran

Theoretical Foundation: Rhythm Formant Theory

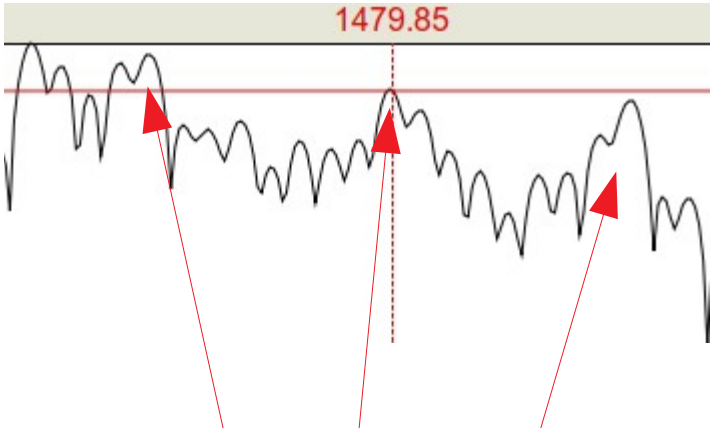
Generalising the concept of *formant*

High Frequency Formants (HF Formants)

1. Formants are the resonant frequencies of the vocal tract.
2. Formants are distinctive frequency components of speech.

Role of HF formants, f > 600Hz : voiced segments (mainly)

[a] in “five”: 1st, 2nd, 3rd formants



[i] in “five”: 1st, 2nd, 3rd formants

Generalisation to Low Frequency Formants (LF Formants)

1. Formants are the resonant frequencies of the vocal tract.
2. Formants are distinctive frequency components of speech.

Role of LF formants, f < 20Hz : rhythms

Examples:

4.3Hz LF formant:
relates to a syllable sequence of mean duration 235ms.

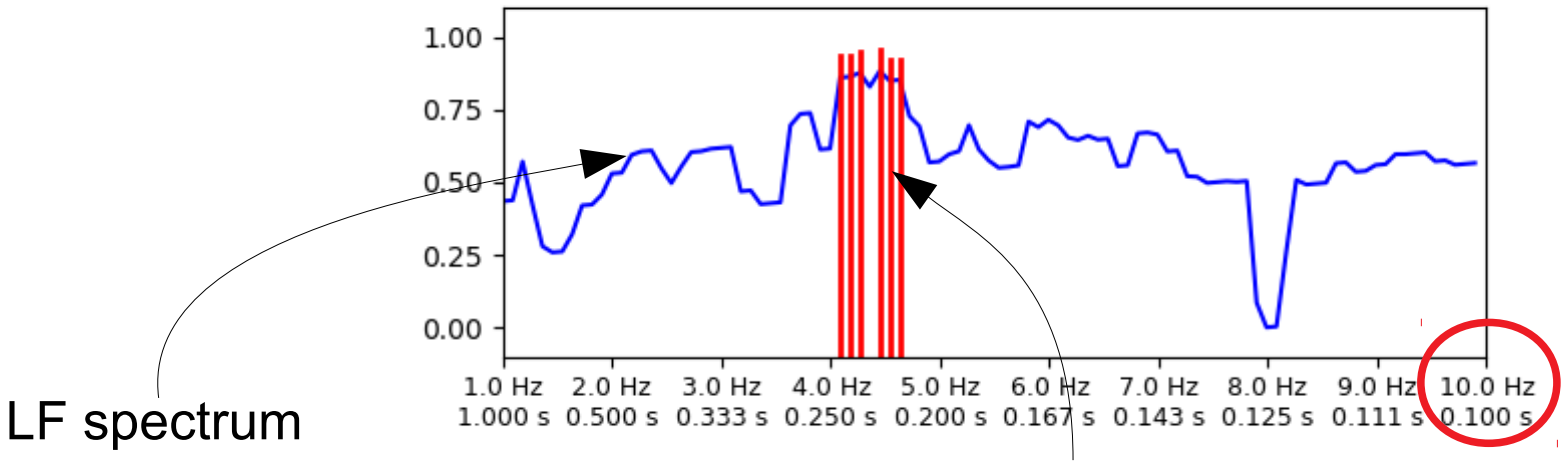
2 Hz LF formant:
relates to a prosodic word sequence, mean duration 500ms.

0.5 Hz VLF formant:
relates to phrasal and other rhetorical rhythms.

Generalisation to Low Frequency Formants (LF Formants)

~~1. Formants are the resonant frequencies of the vocal tract.~~
2. Formants are distinctive frequency components of speech.

Role of LF formants, f < 20Hz : rhythms



Rhythm Formant: RF ≈ 4.3Hz
Interpretation: This RF is syllable-related!
Prediction: If so, mean(|syll|) ≈ 1/RF ≈ 235ms

Terminology: Why call these low frequency zones *formants*?

Terminology: Why call these low frequency zones *formants*?

Comments of some phonetician colleagues:

- agreement:

  Yay, great idea to generalise *formant*!

- critique:

  Wait, this is confusing, ‘formant’ is a fixed technical term!

Of course I agree with the agreement ☺

But the critique deserves a more detailed answer.
It is more than a terminological issue.

Terminology: Why call these frequency zones *formants*?

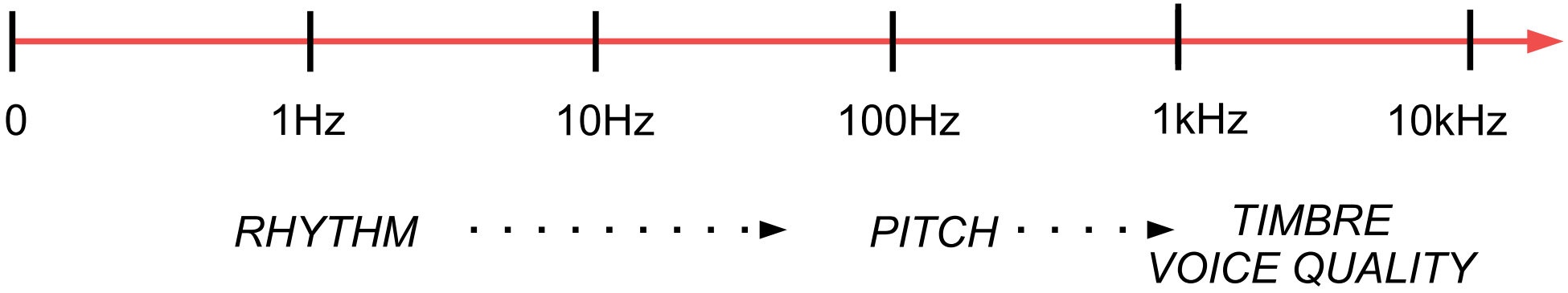
Response:

1. nobody has a copyright on technical terms:
   - the term is also used in musicology:
     in the acoustic domain - timbre of musical instruments
   - many technical terms are polysemous, even ambiguous:
     - frequency in Hz. vs. frequency of words in corpus
     - segment as speech phone or as arbitrary slice / section
     - tone in as lexical or accentual tone (and vs. muscle tone)
2. 'formant' in 'LF/HF formant' is not merely polysemous:
   - mathematically identical - just different frequencies
   - a legitimate generalisation of 'formant'
3. it is necessary to distinguish between
   1. frequency zone – the physical definition
   2. rhythm formant – the functional definition

Rhythm Generalisation in Musicology: Musical Relativity Theory

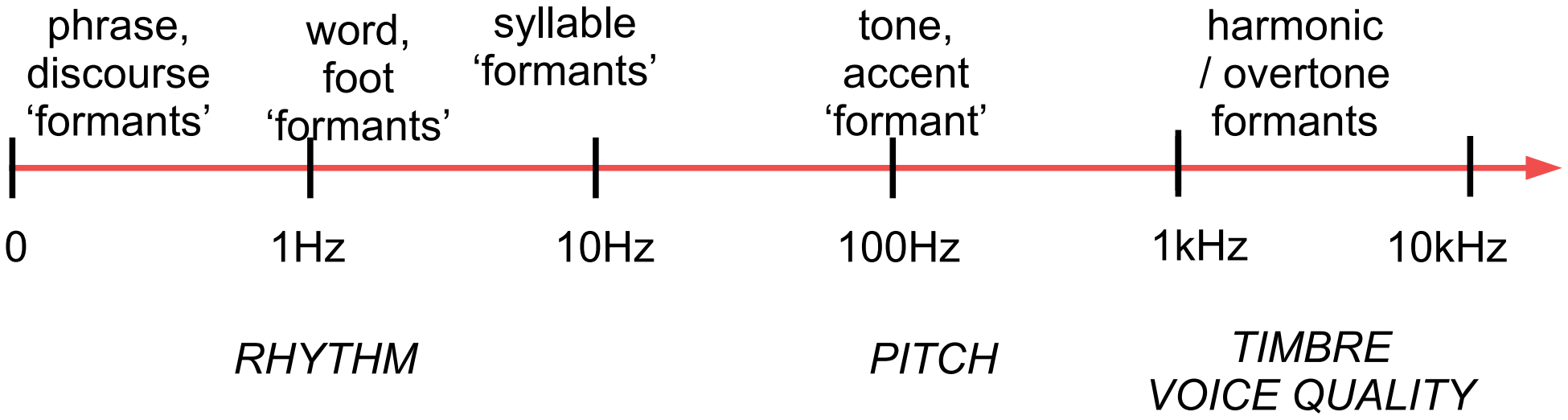
Compare:

- very low notes in organ music which mutate into rhythmic beats
- helicopter rotor oscillation as rhythm vs. the pitch of a car engine
- very low vocal pitch which mutates into creaky voice
  (cf. Mandarin Dipping Tone, Tone 3)

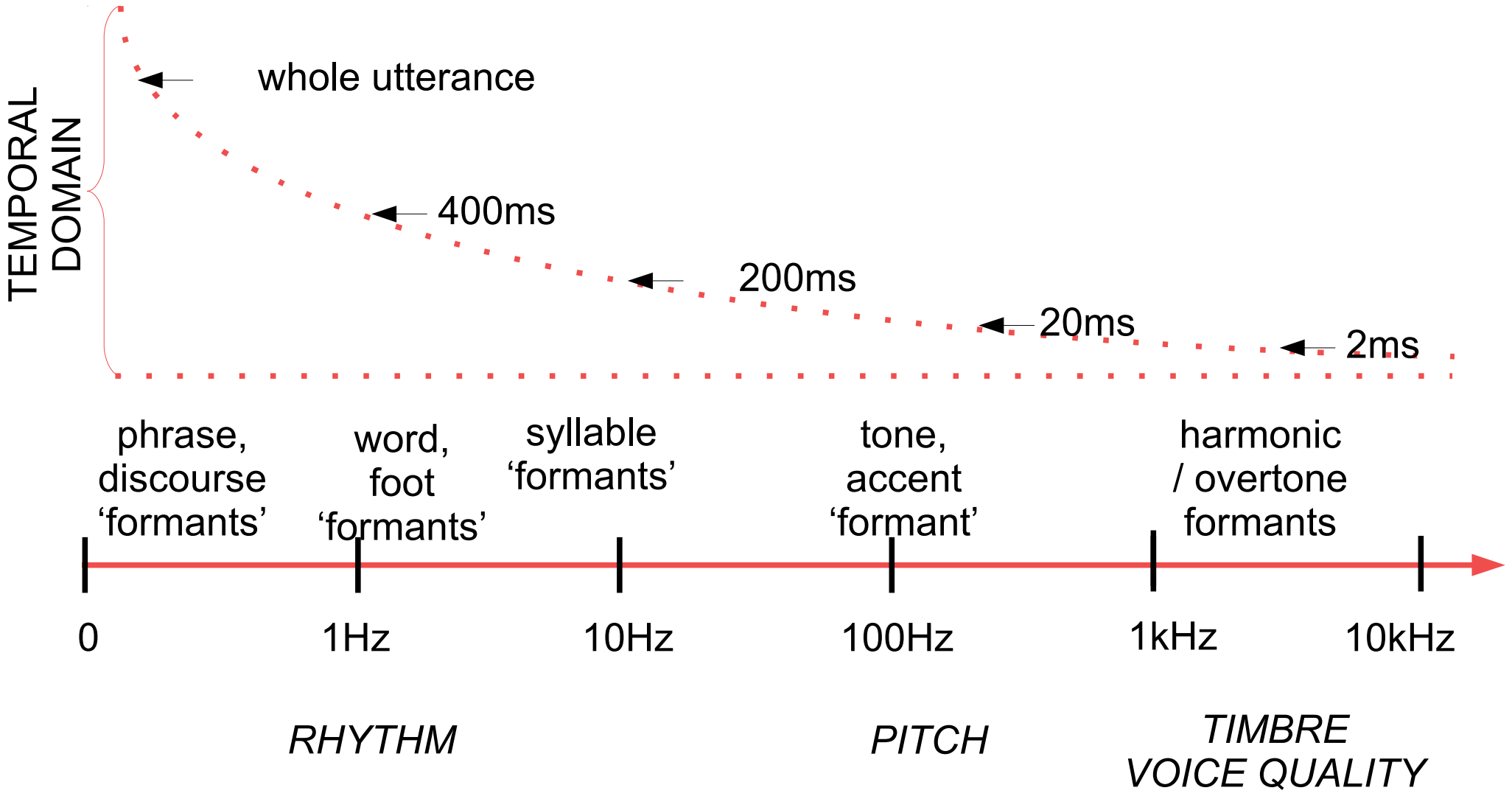


Source: the classic in musicology:
Cowell, Henry. 1930. *New Musical Resources*. New York: Alfred A. Knopf Inc.

The Generalisation in Musicology: Musical Relativity Theory



The Generalisation in Musicology: Musical Relativity Theory



Rhythm Formant Analysis

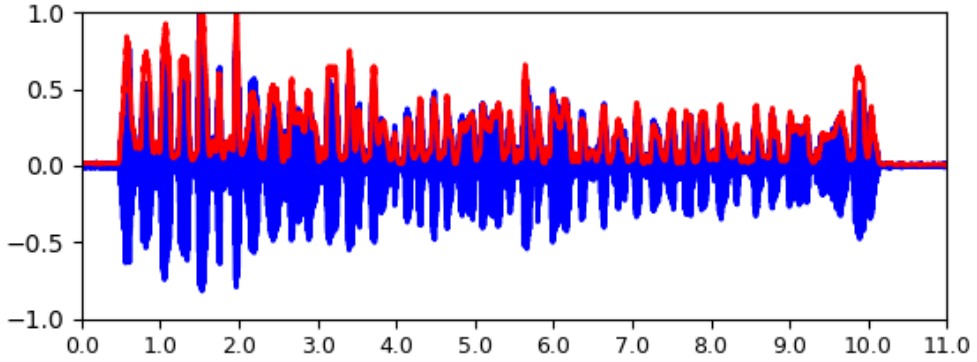
- Low Frequency Formant Extraction Method -

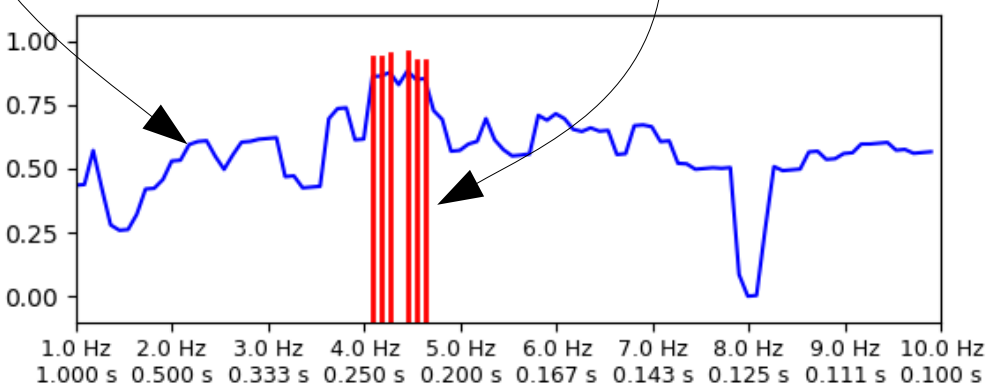
language-independent
automatic identification of speech rhythms
in syllables, words, discourse
embedded in a generalised formant theory

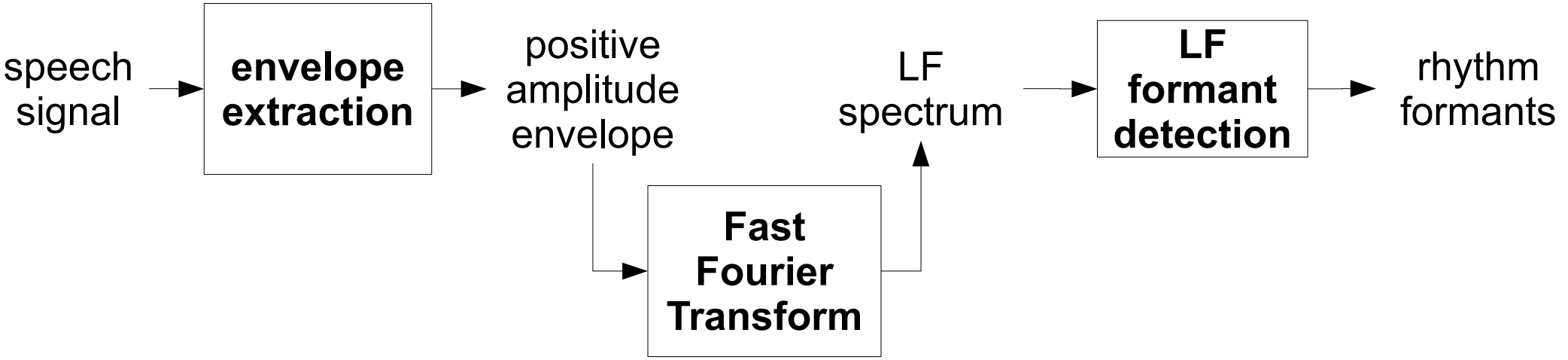
LF Formant Extraction Method

normalised low frequency spectrum

frequency zone with highest magnitude, interpreted as a ***rhythm formant***



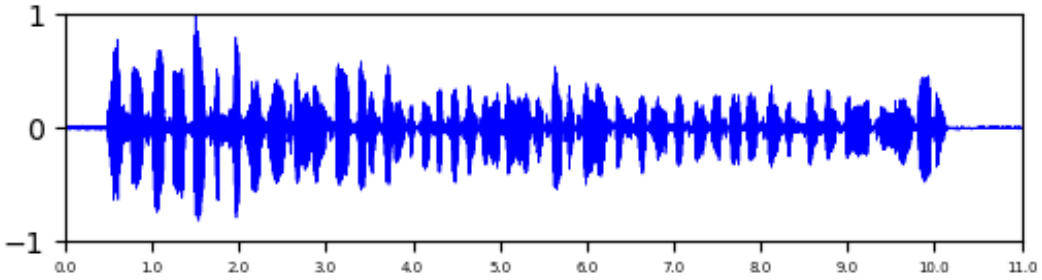




Waveform, Envelope Extraction

Envelope Extraction

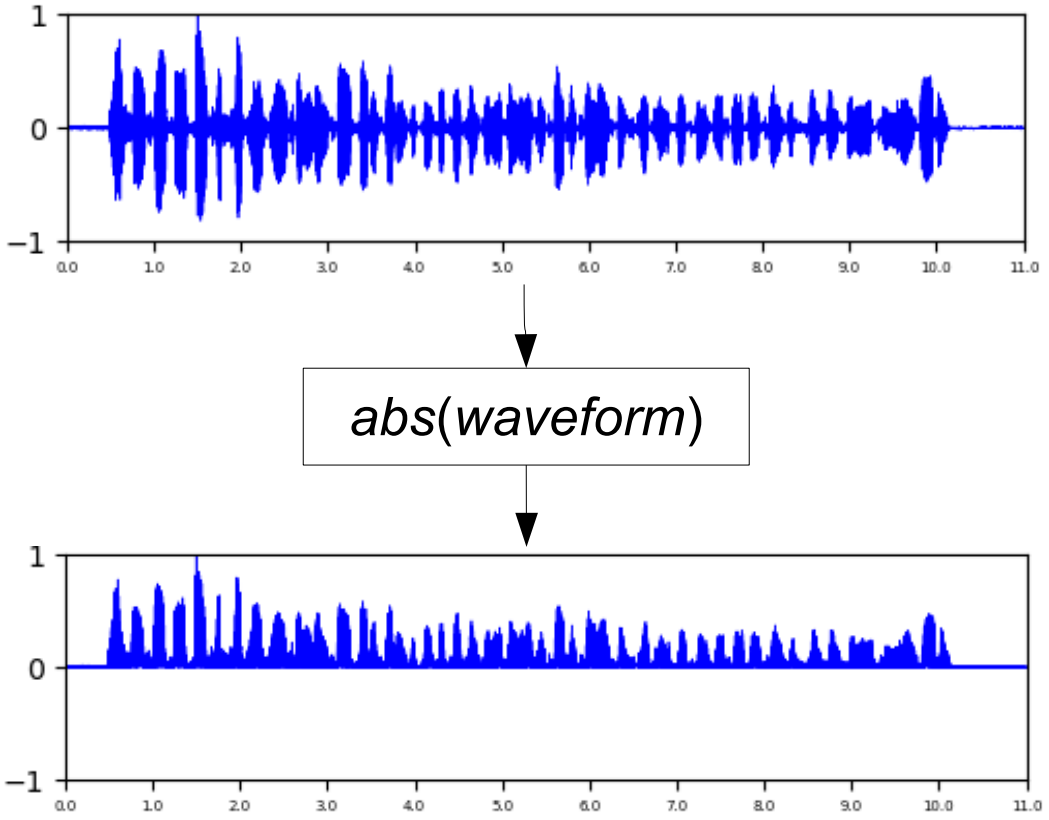
Waveform
(Oscillogram)



Envelope Extraction (time domain to time domain)

Waveform
(Oscillogram)

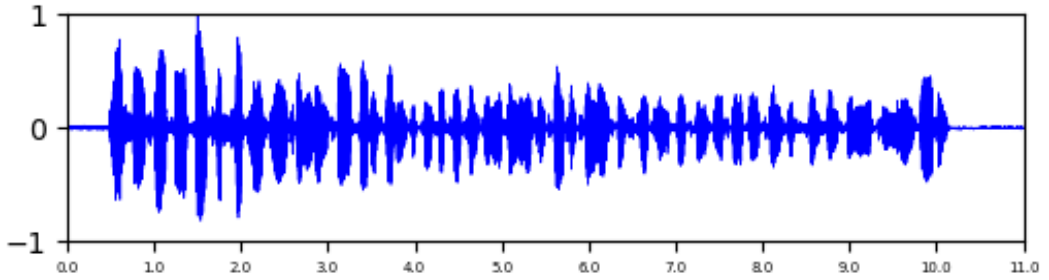
Alternatives (not identical):
- *Rectification*
- *Absolute*



Positive rectified waveform,
Absolute waveform

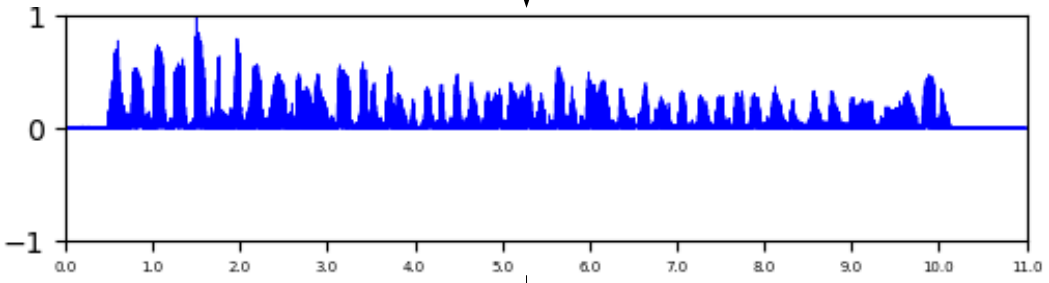
Envelope Extraction (time domain to time domain)

Waveform
(Oscillogram)



Alternatives (not identical):
- *Rectification*
- *Absolute*

abs(*waveform*)

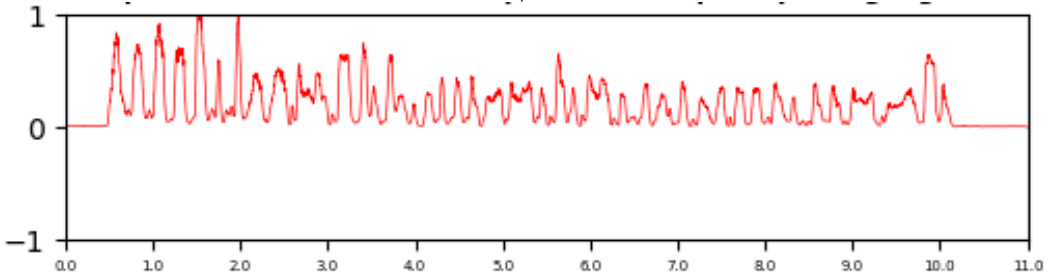


Positive rectified waveform,
Absolute waveform

Alternatives:
- *Hilbert Transform*
- *Peak-picking in moving window*

+ low pass filter

peakpick(abs*waveform*)



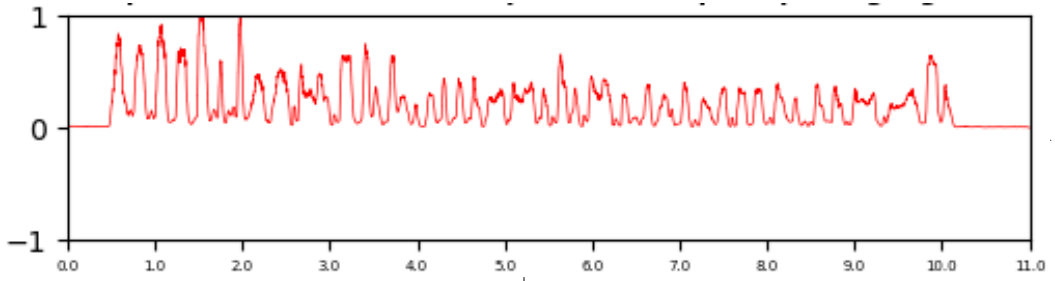
Amplitude Envelope,
Amplitude Modulation Envelope,
Intensity Outline, ...

Spectrum and Slope Normalisation

LF spectral Analysis (time domain to frequency domain)

Amplitude Modulation Envelope

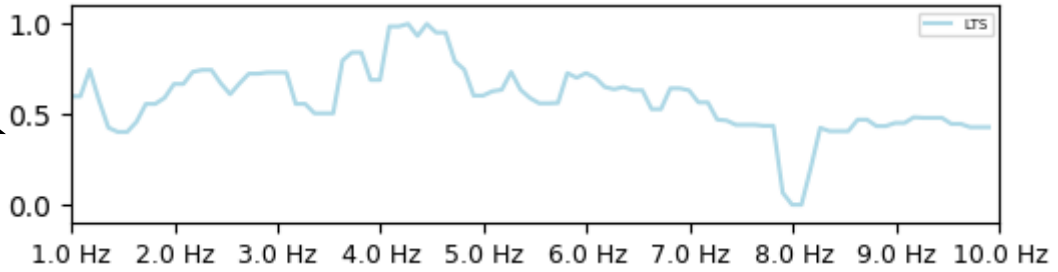
time domain



Fast Fourier Transform
window > 3s … whole utterance

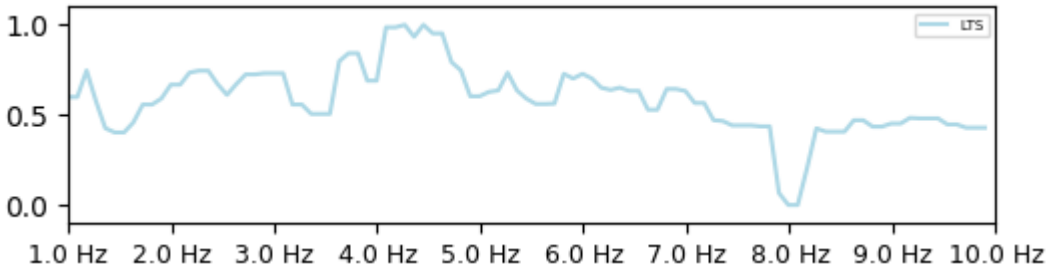
frequency domain

Low Frequency Spectrum
1 Hz, …, 10 Hz



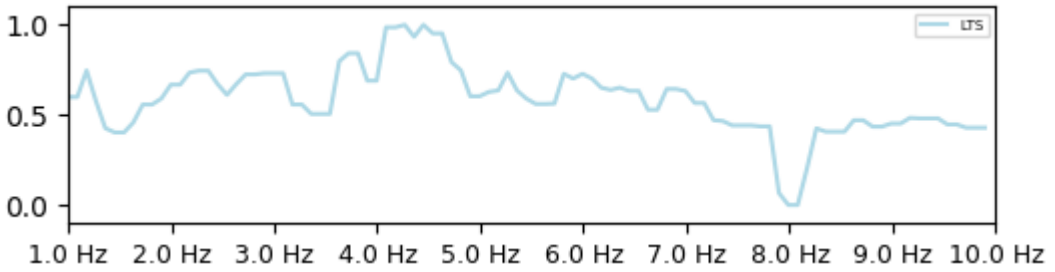
Spectrum Normalisation (frequency domain to frequency domain)

Low Frequency Spectrum
1...10 Hz

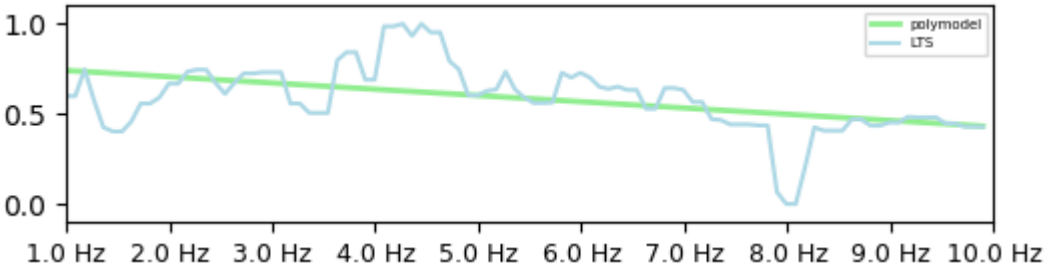


Spectrum Normalisation (frequency domain to frequency domain)

Low Frequency Spectrum
1...10 Hz

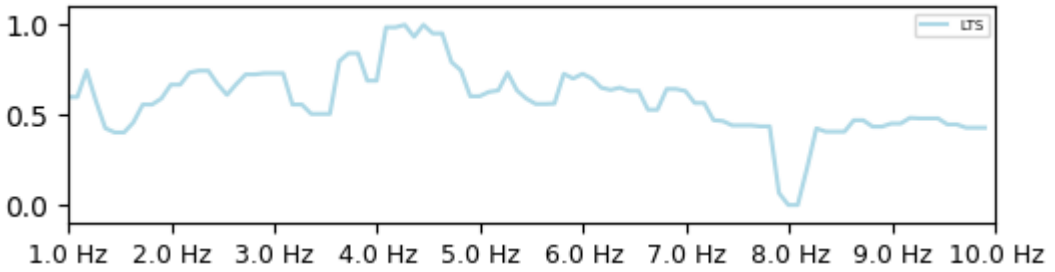


Regression Line
for slope normalisation

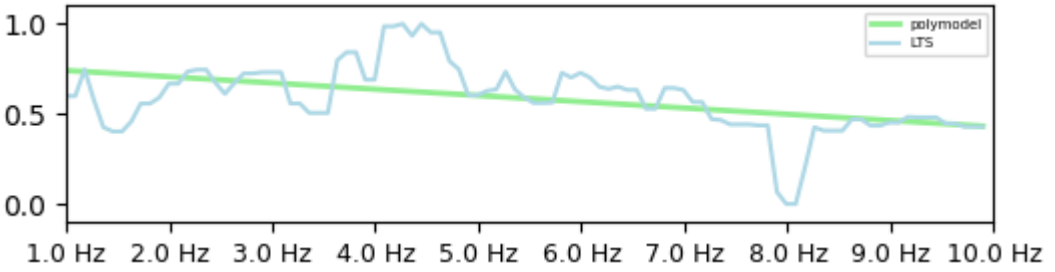


Spectrum Normalisation (frequency domain to frequency domain)

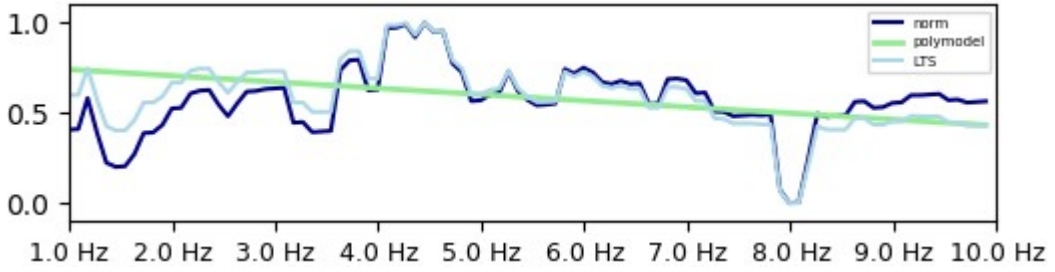
Low Frequency Spectrum
1...10 Hz



Regression Line
for slope normalisation

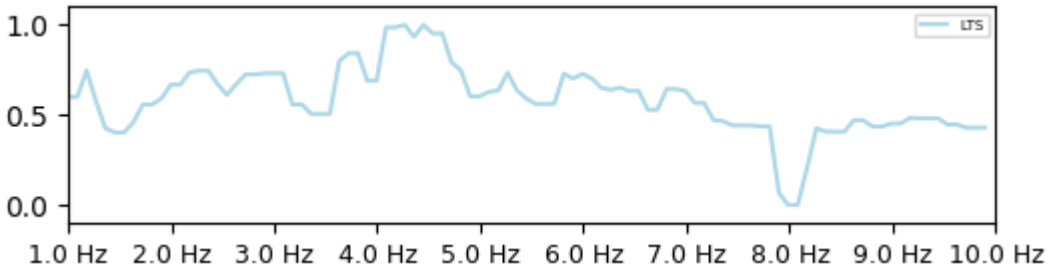


Raised residuals
spectrum – regression
(slope normalised spectrum)

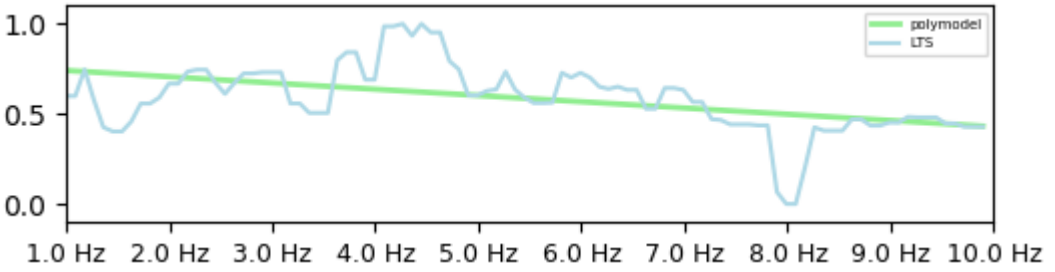


Spectrum Normalisation (frequency domain to frequency domain)

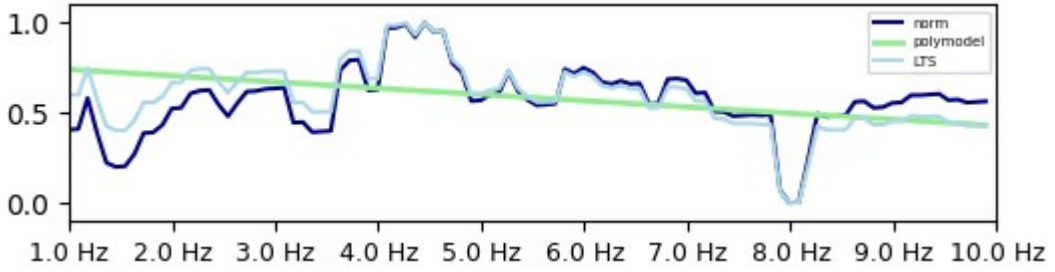
Low Frequency Spectrum
1...10 Hz



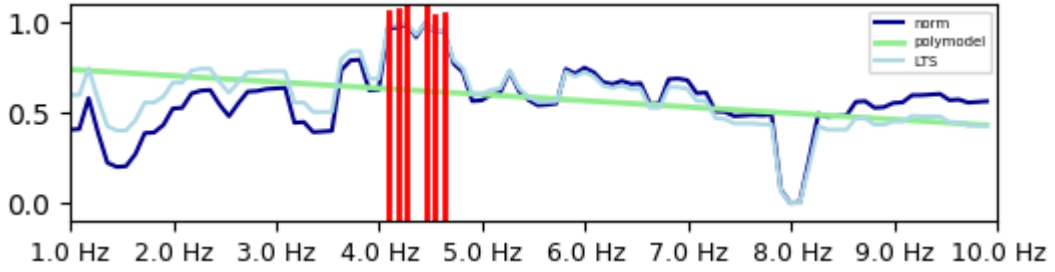
Regression Line
for slope normalisation



Raised residuals
spectrum – regression
(slope normalised spectrum)



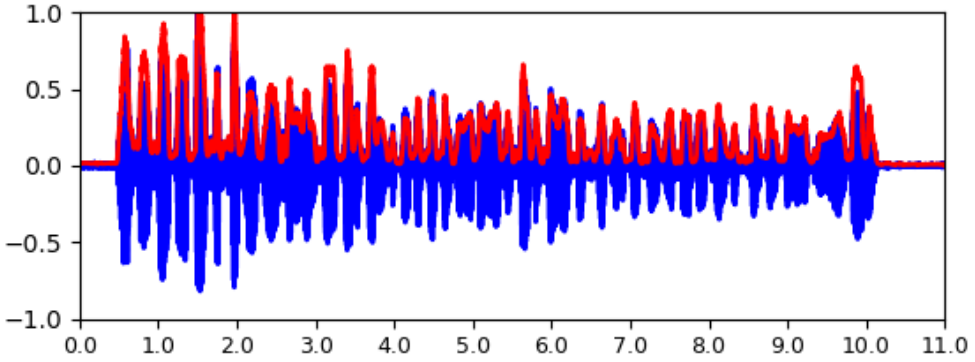
The highest magnitude
frequencies (‘Rhythm Candy’)
(‘rhythm bars’)

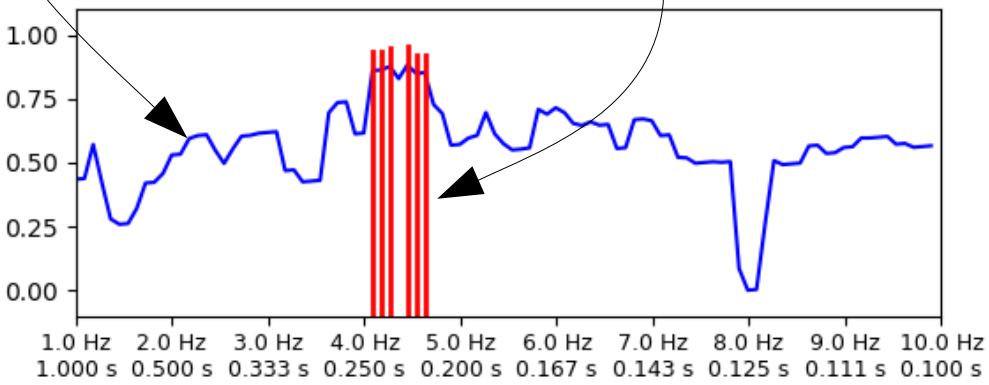


Low Frequency Formant Extraction: Rhythm Candy

normalised low frequency spectrum

'Rhythm Candy', highest magnitude frequency cluster, interpreted as a ***Rhythm Formant***

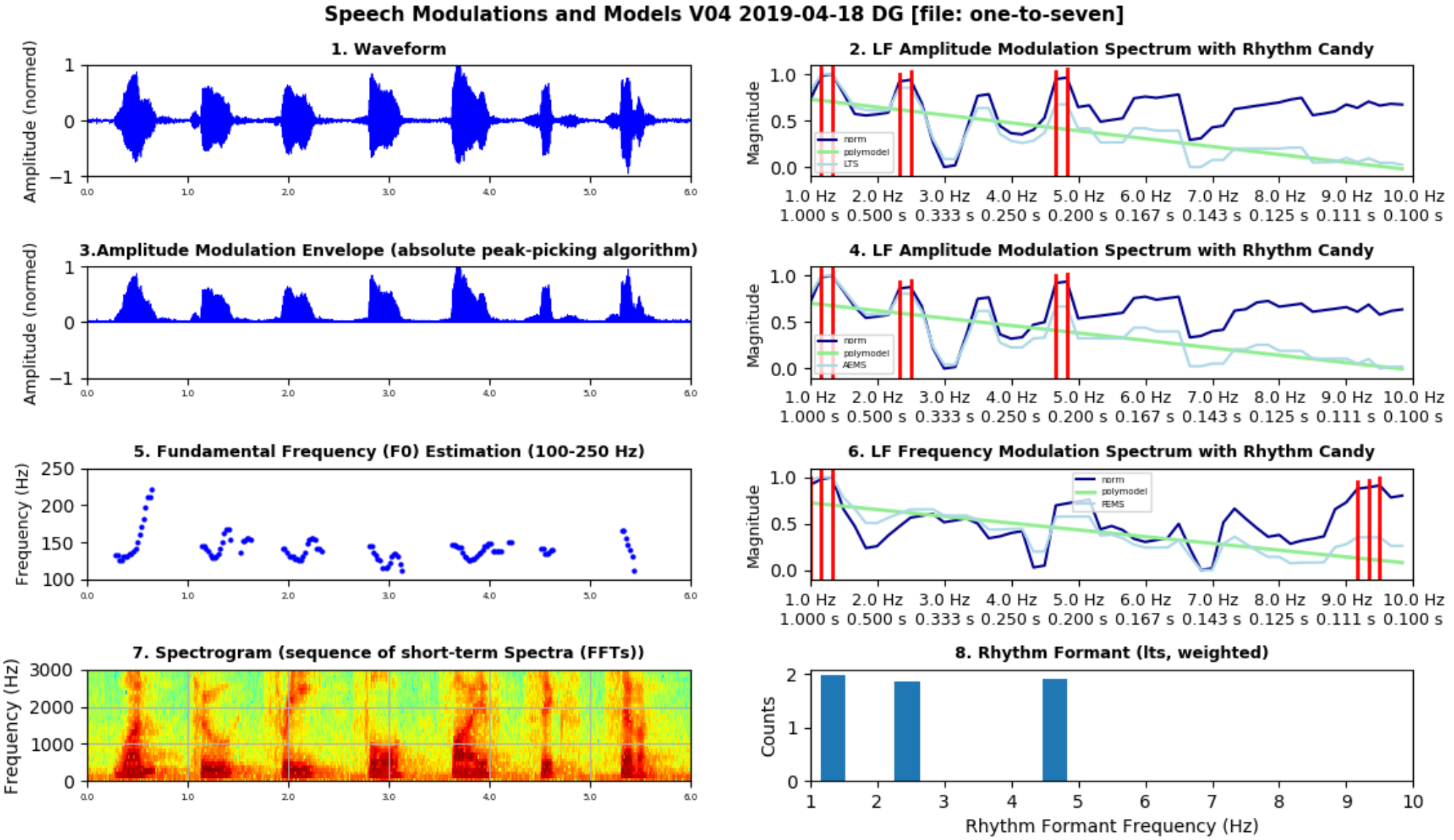




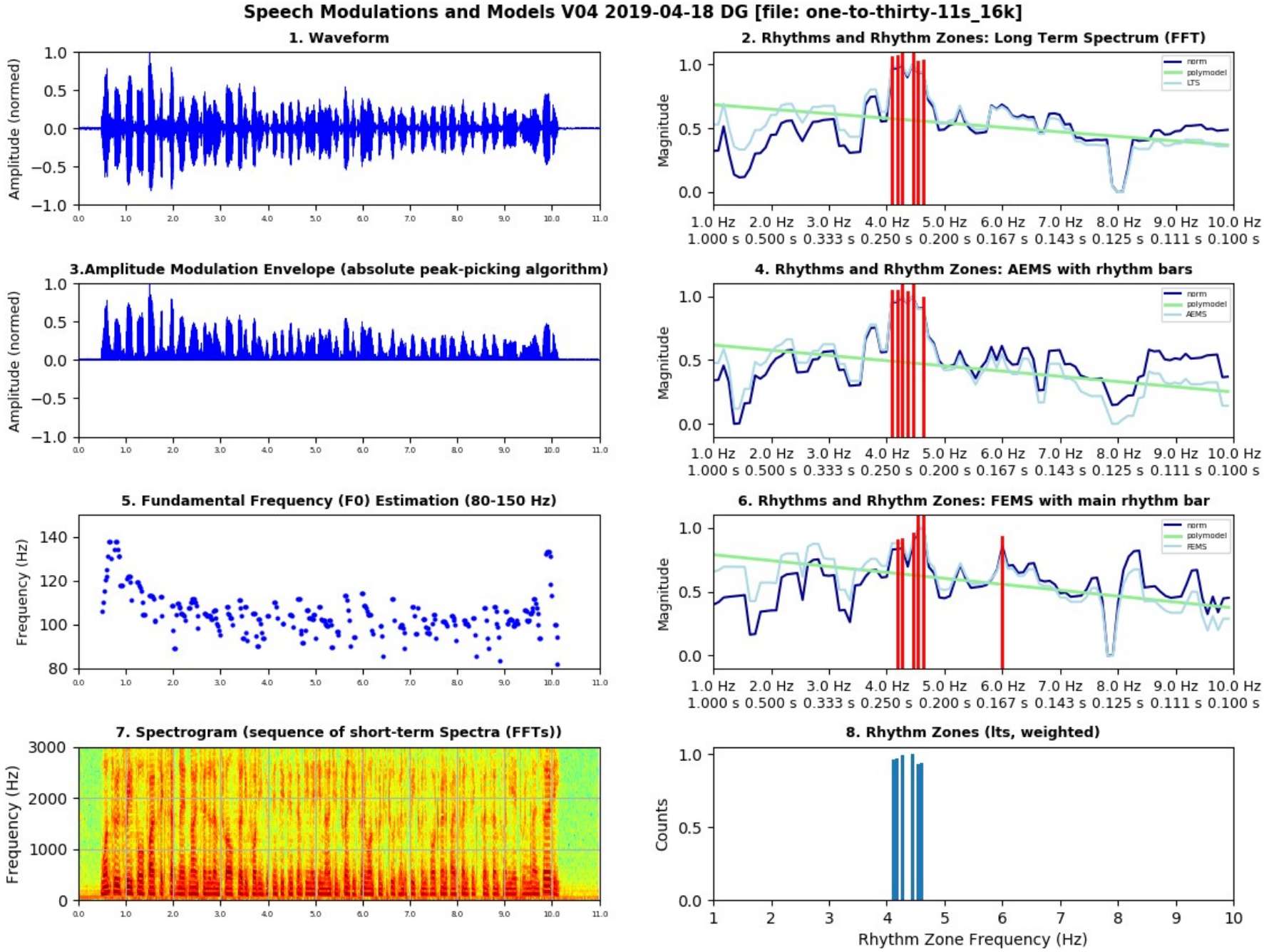
Fast Fourier Transform
+
Slope Normalisation

Rhythm Formant Analysis in Context: Counting – a Clear Case

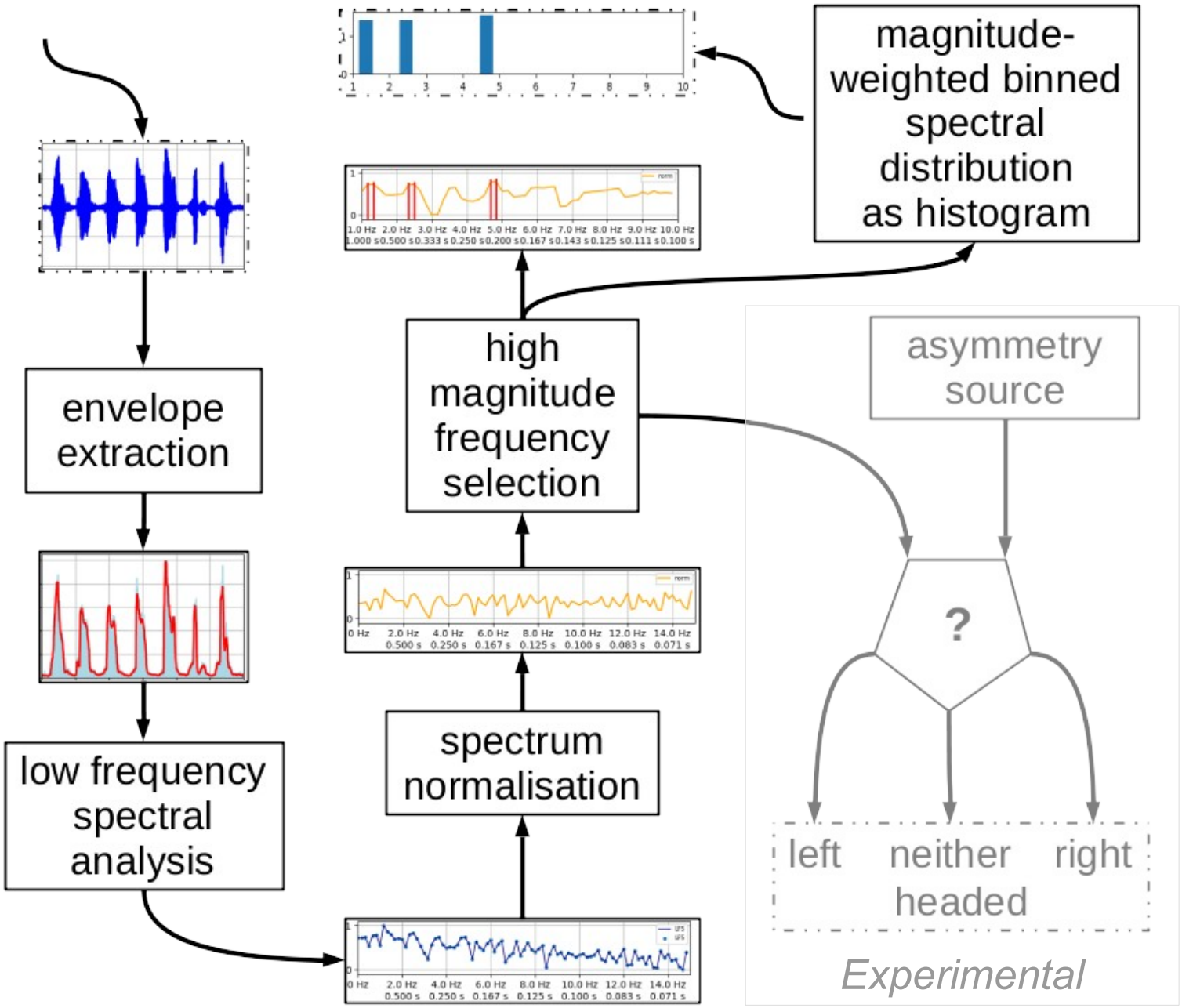
Rhythm Formant Analysis: Rhythm Candy in Context



Rhythm Formant Analysis: Rhythm Candy in Context



Summary: Rhythm Formant Analysis Dataflow



Implementation

Configuration Parameters as Python Variables

```
# Graph configuration

fontsize = 9
graphwidth = 12
graphheight = 9

# Graph format is *png* or *svg*
graphformat = png


# Sound input parameters, channels: left, right, merge
channel = merge

# Signal segment in secs
# ("signalend = 0" means the whole signal).
signalstart = 0.0
signalend = 30.0
```

Configuration Parameters as Python Variables

```
# Envelope parameters
envwin = 20
envmedianfilt = 501
envheight = 1.1
waveformshow = False

# Spectrum amplification exponent
spectrumpower = 2

# Normalisation function
spectrumpoly = 1

# Long term spectrum frequency range (Hz min / max; samples)
lfsmin = 1
lfsmax = 10
lfsmedfilt = 3

# Rhythm bars
rhythmcount = 6
rhythmlabel = False
```

Envelope Extraction

```
def makeenvelope(signal, peakwin, envmedianfilt):

    signalabs = abs(signal)
    peaksrange = range(len(signalabs)-peakwin)
    envelope = [ max(signalabs[i:i+peakwin]) for i in peaksrange ]

    padleft = [envelope[0]] * int(round(peakwin/2.0))
    padright = [envelope[-1]] * int(round(peakwin/2.0))
    envelope = np.asarray(padleft + envelope + padright)

    envelope = medfilt(envelope,envmedianfilt)
    envelope = envelope / float(max(envelope))

    return envelope
```

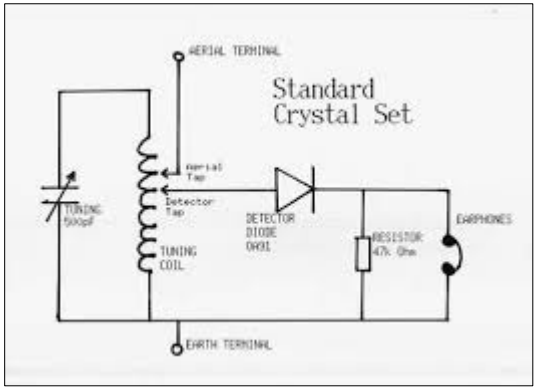
OR:

```
    envelope = scipy.signal.hilbert(signal)
```

☺

Envelope Extraction

```
def makeenvelope(signal, peakwin, envmedianfilt):

    signalabs = abs(signal)
    peaksrange = range(len(signalabs)-peakwin)
    envelope = [ max(signalabs[i:i+peakwin]) for i in peaksrange ]

    padleft = [envelope[0]] * int(round(peakwin/2.0))
    padright = [envelope[-1]] * int(round(peakwin/2.0))
    envelope = np.asarray(padleft + envelope + padright)

    envelope = medfilt(envelope,envmedianfilt)
    envelope = envelope / float(max(envelope))

    return envelope
```



Fourier Transform: Specification

- Fourier Transform
  - DFT
  - FFT

formulae

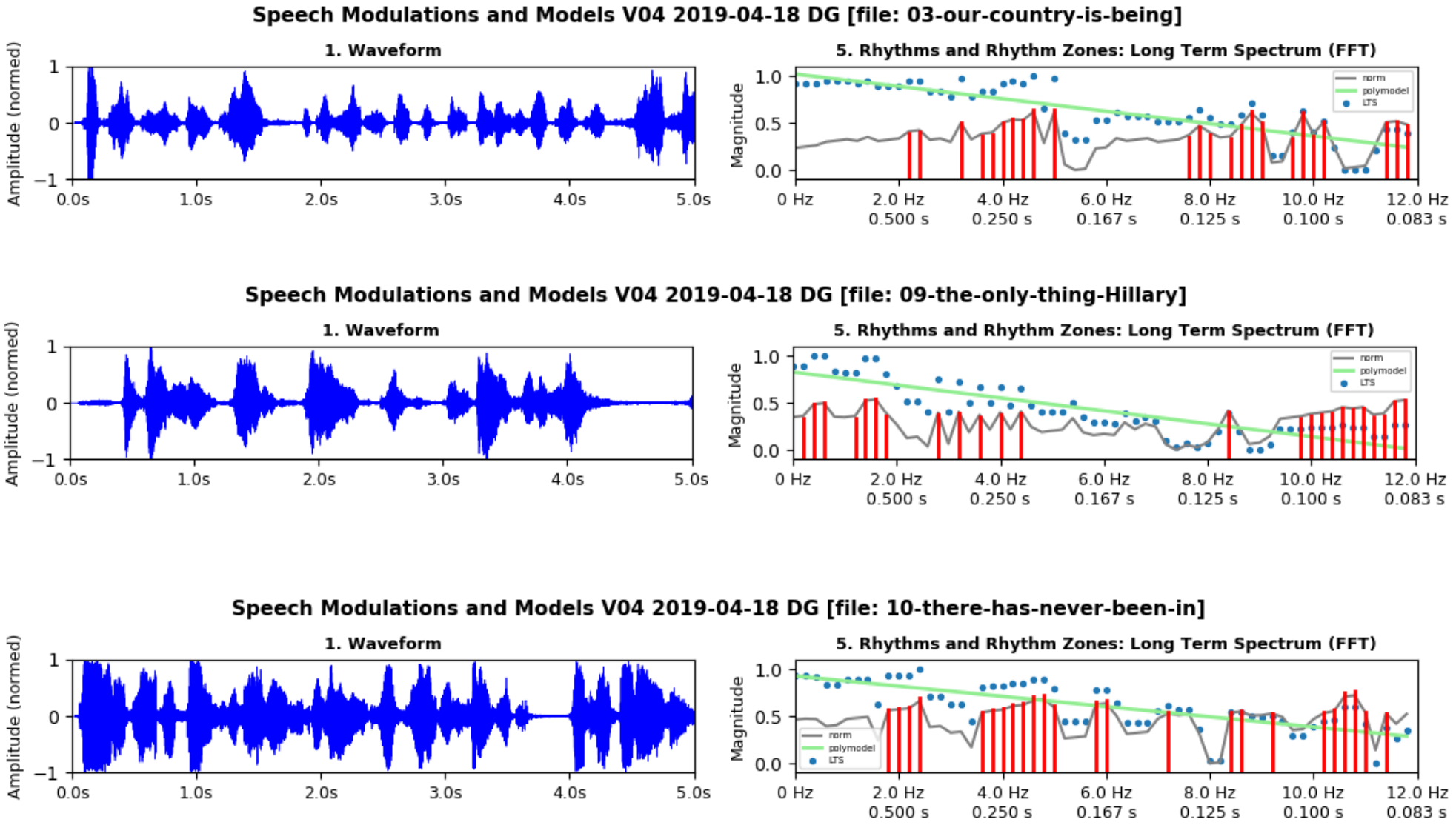
- Essentially:
  - spectrum, spectral slice:
    - a window (segment of signal) of a specified length
    - sine waves at harmonically related frequencies (number determined by desired resolution)
    - correlation with the speech signal
  - sequence of temporally adjoining or overlapping spectral slices (spectra)
- In principle, other waveforms apply in special cases

Fourier Transform: Implementation

```
def fft(signal, samplingrate):
    period = 1.0 / samplingrate

    mags = np.abs(np.fft.rfft(signal))**2
    freqs = np.abs(np.fft.rfftfreq(signal.size,period))
    mags = np.asarray([0.000001 if m==0 else m for m in mags])
    return freqs, mags

frequencies, magnitudes = fft(abs(signal), samplingrate)
magnitudes = scipy.signal.medfilt(magnitudes, medianfiltervalue)


hist_lfs, bins_lfs = np.histogram(frequencies, weights=magnitudes)
```

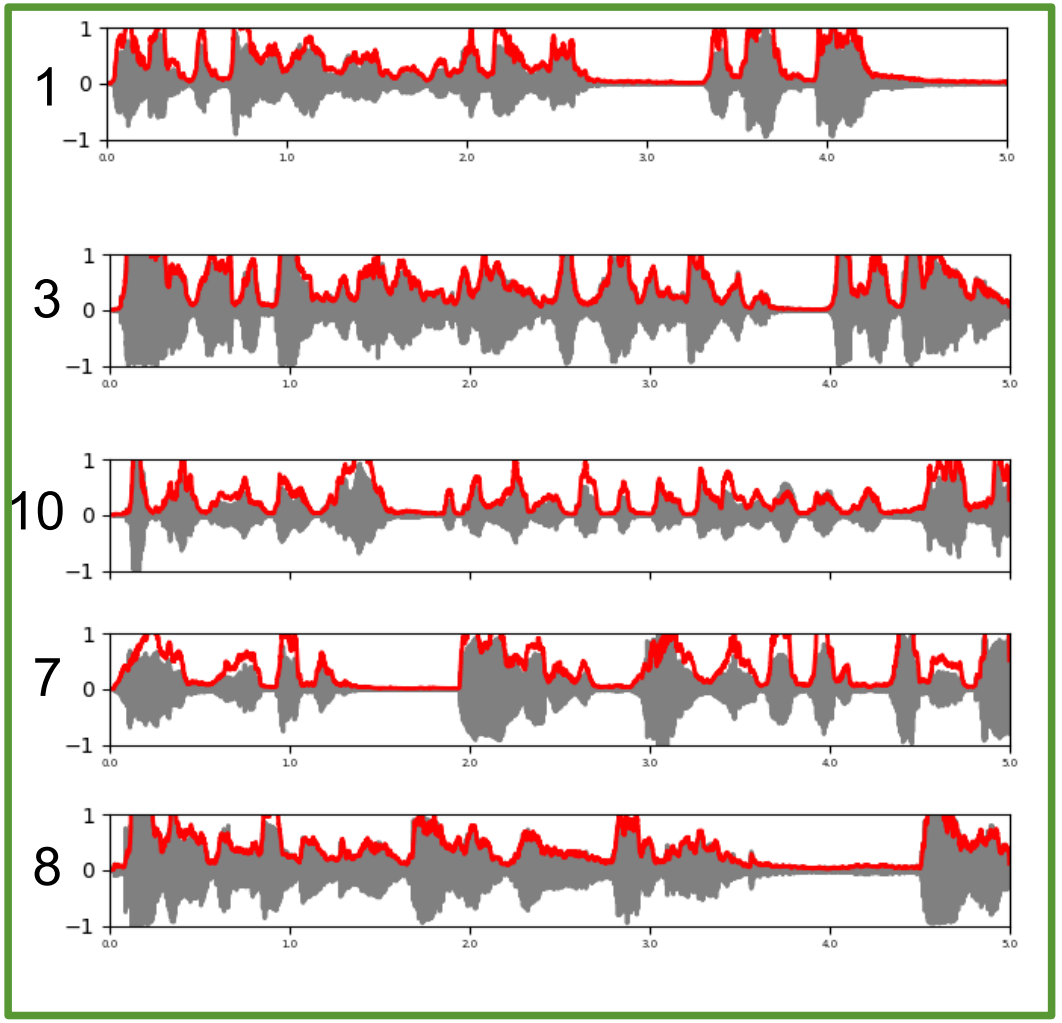
Test Case: Rhythm variation in public speech

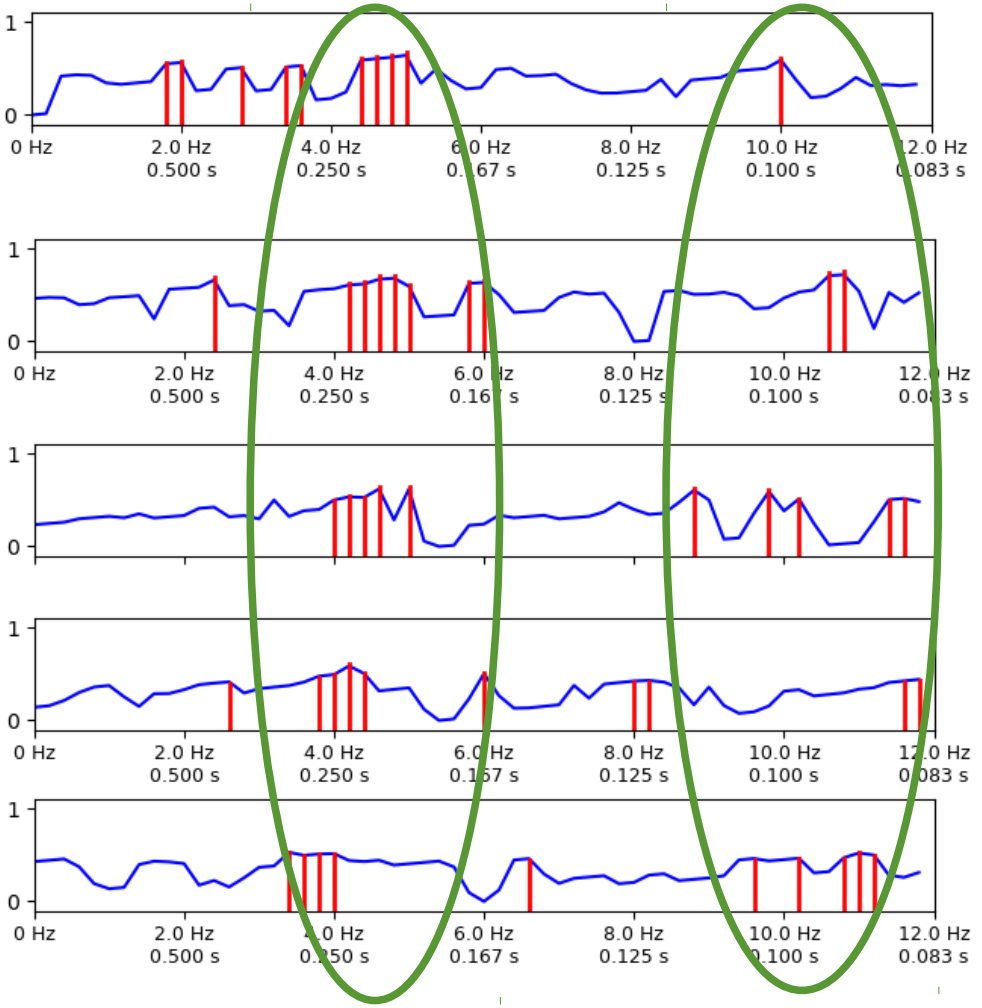
Rhythm Formant Theory: Case Study



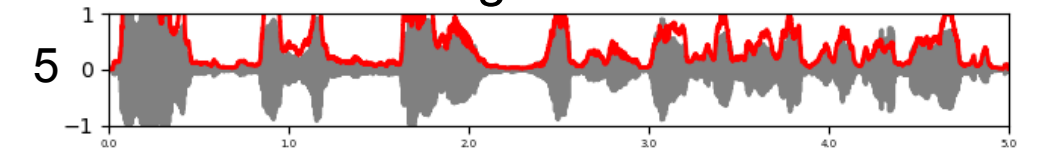
Rhythm Formant Theory: Non-Narrative Speech Style

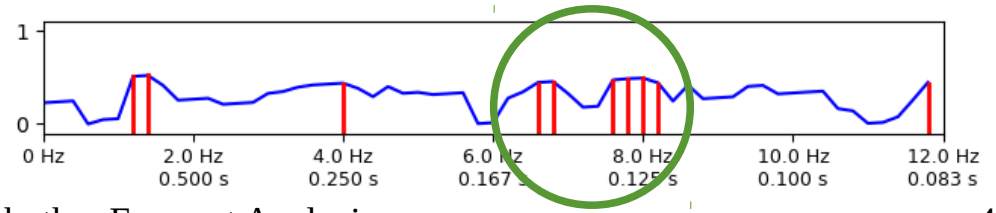
Long regularly spoken segments





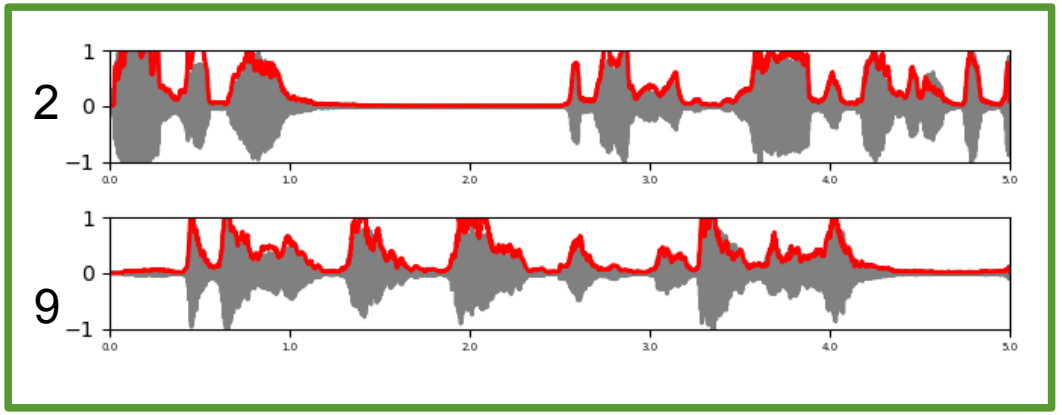
Slight outlier

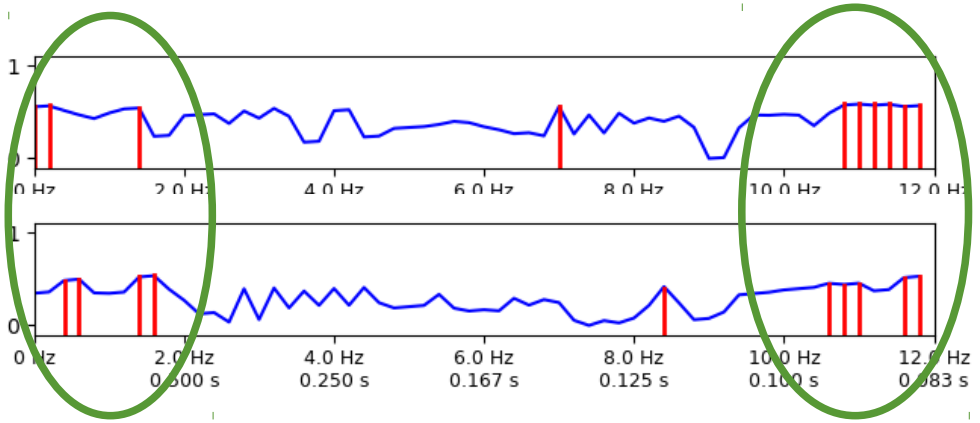




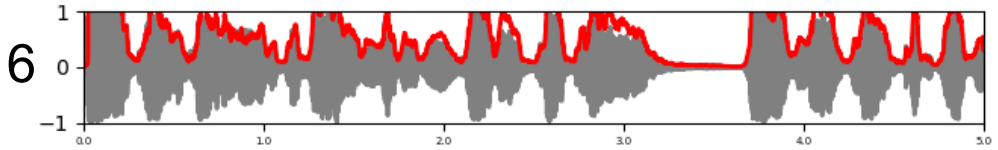
Rhythm Formant Theory: Non-Narrative Speech Style

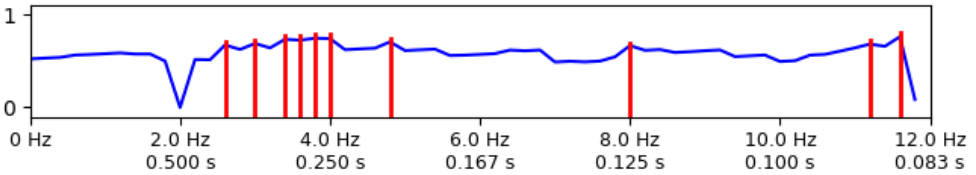
Short chunks, many pauses



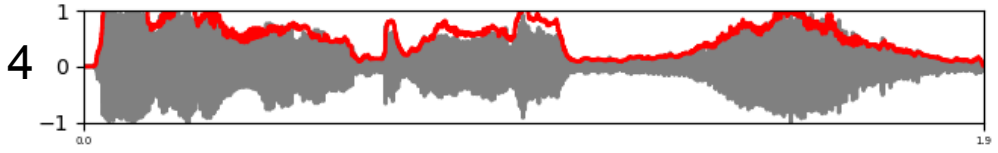


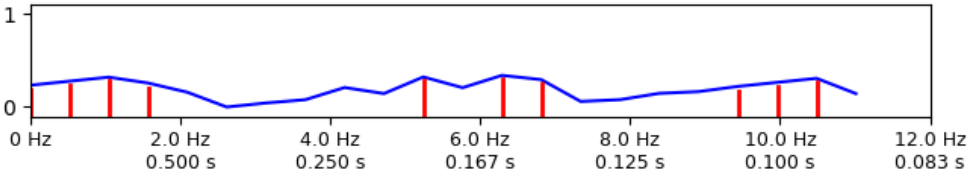
Functionally hybrid



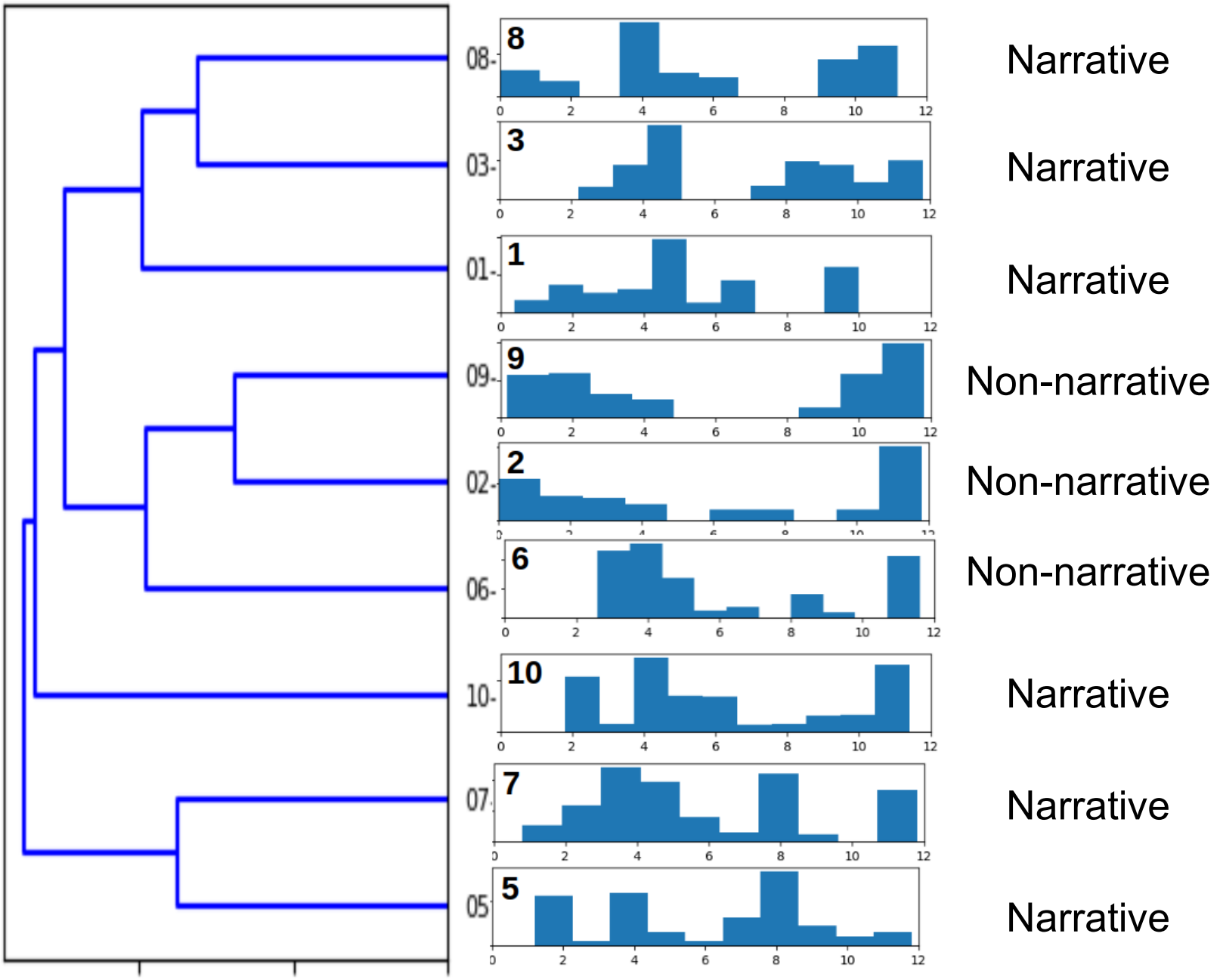


Very short





Automatic Classification based on pairwise Cosine Distance



Conclusion

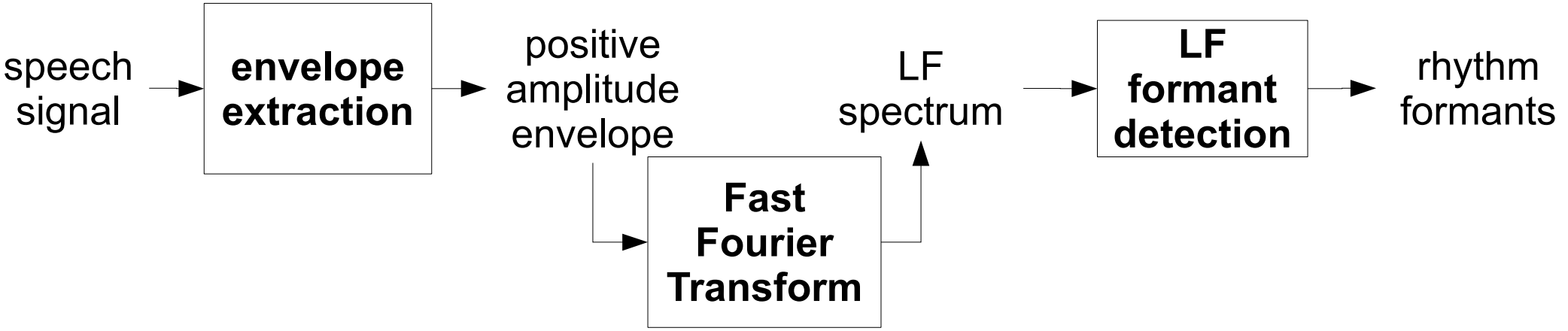
1) Fulfilment of software specification

2) Code Appendix

Fulfilment of Rhythm Formant Analysis Tool Specification

Requirements:
Implementation of Rhythm Formant Theory as a model of speech rhythm perception
Properties: rhythm as oscillation, multiple variable rhythms, asymmetrical rhythms

Design:
Low Frequency amplitude envelope spectrum maxima derived with
1) either Hilbert Transform or Peak Picking
2) Fourier Transform
3) Histogram of LF spectrum



Implementation:
Python 3.n with Numpy, SciPy and MatPlotLib libraries
Single module with separate configuration file.

Evaluation:
1) Test bench comparison with isochrony metric heuristics
2) Case studies on the pragmatic interpretation of rhythm variation

Code Appendix

Code Appendix: Configuration and Speech Input

```
#!/usr/bin/python3
# rfa-test.py, D. Gibbon, 2019-09-15, 2019-11-24
# Visualise waveform, envelope, low frequency spectrum


import sys, re; import numpy as np; import matplotlib.pyplot as plt
import scipy.io.wavfile as wave; from scipy.signal import medfilt

wavfilename = sys.argv[1]
fontsize = 9; graphwidth = 12; graphheight = 3
graphformat = "png"; showgraph = True
wavecolor = "blue"; abswavecolor = "lightblue"
envelopecolor = "r"
envwin = 20; envmedianfilt = 501; envheight = 1.1
rhythmzones = "aems"; rhythmcount = 6;
spectrumpower = 1; spectrumpoly = 1
freqmin = 1; freqmax = 10; magnitudesmedfilt = 3


samplerate, signal = wave.read(wavfilename)
if len(signal.shape) > 1: signal = (signal[:,0]/2.0 + signal[:,1]/2.0)
signal = signal/float(max(abs(signal)))
signalabs = np.abs(signal)
signallen = len(signal); signalsecs = float(signallen)/samplerate
signalstart = 0; signalend = signalsecs
```

Code Appendix: Envelope, FFT, Regression Definitions

```
def makeenvelope(signalabs, envwin, envmedianfilt):
    peaksrange = range(len(signalabs)-envwin)
    envelope = [ max(signalabs[i:i+envwin]) for i in peaksrange ]
    padleft = [envelope[0]] * int(round(envwin/2.0))
    padright = [envelope[-1]] * int(round(envwin/2.0))
    envelope = padleft + envelope + padright
    envelope = np.asarray(envelope)
    envelope = medfilt(envelope,envmedianfilt)
    envelope = envelope / float(max(envelope))
    return np.asarray(envelope)

def fft(signalabs,samplerate):
    period = 1.0/samplerate
    mags = np.abs(np.fft.rfft(signalabs))
    freqs = np.abs(np.fft.rfftfreq(signalabs.size,period))
    mags = np.asarray([0.000001 if m==0 else m for m in mags])
    mags[0] = mags[1]
    mags = mags / np.max(mags)
    return freqs, mags

def polyregline(x,y,d):
    x = range(len(y))
    fit, res, _, _, _ = np.polyfit(x, y, d, full=True)
    yfit = np.polyval(fit,x)
    return yfit
```

Code Appendix: Envelope, FFT, Regression Calls

```
envelope = makeenvelope(signalabs, envwin, envmedianfilt)

frequencies,magnitudes = fft(abs(envelope),samplerate)
magnitudes[0] = magnitudes[1]
magnitudes = medfilt(magnitudes,magnitudesmedfilt)



xmin = int(round(freqmin * len(frequencies) / frequencies[-1]))
xmax = int(freqmax * len(frequencies) / frequencies[-1])
frequencies = frequencies[xmin:xmax]

data = magnitudes[xmin:xmax] ** spectrumpower
data = (data - np.min(data)) / (np.max(data)-np.min(data))
x = np.arange(len(data))


polymodel = polyregline(x,data,spectrumpoly)
polyresid = data - polymodel
polyresidnorm = polyresid+abs(np.min(polyresid))
```

Code Appendix: Figure Definition, Waveform and Spectrum Plots

```
_,(pltwave, pltspectrum) = plt.subplots(nrows=1, ncols=2, figsize=(graphwidth, graphheight))


 pltwave.set_title("Waveform")
 siglen = len(signal); sigtime = 1.0 * siglen/samplerate
 print(siglen,sigtime)
 x = np.linspace(0,sigtime,siglen)
 pltwave.plot(x,signal)
 pltwave.plot(x,signalabs, color=abswavecolor)
 pltwave.plot(x,envelope, color=envelopecolor)


 pltspectrum.set_title("Normalised Low Frequency Spectrum with Rhythm Candy")
 pltspectrum.plot(frequencies,data, color='gray', label='norm')
 pltspectrum.plot(frequencies,polymodel, color='gray', label='norm')
 pltspectrum.plot(frequencies,polyresidnorm, color='b', label='norm')
 pltspectrum.set_xlim(freqmin, freqmax)
 pltspectrum.set_ylim(-0.1, 1.1)
 xtix = pltspectrum.get_xticks()
 xtixlabels = [ "%.1f Hz\n%.3f s"%(i,1.0/i) if i>0 else "0 Hz" for i in xtix ]
 pltspectrum.set_xticklabels(xtixlabels,fontsize=fontsize)
 pltspectrum.set_ylabel("Magnitude", fontsize=fontsize)
```

Code Appendix: Rhythm Candy Plot, Graph Display

```
if rhythmcount > 0:
    print("Rhythm Formant Analysis: Frequency and Magnitude")
    speclist = polyresidnorm.tolist()
    speclistsort = sorted(speclist)
    speclistsortrev = reversed(speclistsort[-rhythmcount:])
    for i, item in enumerate(speclistsortrev):
        b = speclist.index(item)      # actual vector position
        f = float(b)/signalsecs # vector position in Hz
        print(i,item,f,b)
        f = f + freqmin                    # if signal does not start at zero
        pltspectrum.axvline(f,ymin=0, ymax=item, linewidth=2, color='r')
```

```
plt.tight_layout(pad=3, w_pad=1, h_pad=1)
plt.savefig("RhythmFormantAnalysis.png")
if showgraph:
    plt.show()
```

Thanks!

Good luck with applications of the RFA tools (there are more)!

And do keep in touch!