

Michael Epstein

## Computational Procedure for Estimating Loudness from 1-kHz tone-burst otoacoustic emissions

See Epstein and Silva, 2009 for procedural and stimuli details.

Generally, stimuli are 1-kHz tones windowed with 4-ms Gaussian windows (2-ms up, 2-ms down). Windowed tones are then end-padded with silence to generate a stimulus length of 41.7 ms.

Many trials are presented at each SPL for which a loudness estimate is desired. (Epstein and Silva used 1000 trials per SPL). Stimuli are presented and recordings are made via the ER-10c ear-probe system. Any similar substitute could also be used.

The recordings are split in half and each half is averaged resulting in two time-domain averages of tone-burst presentation and the resulting response.

The following reference analysis code outlines the procedure by which these recordings are transformed into single estimates of loudness for each level. The value of loudness given is only useful for relative comparison with the loudness estimates at other levels within the same listener. This procedure run across a wide range of levels then provides a loudness function, which can be used to derive information about loudness growth and cochlear compression.

```
ffflen = LENGTH OF YOUR ENTIRE SIGNAL IN POINTS; (In Epstein and Silva 2009 = 2002 points)
analen = LENGTH OF THE ANALYSIS WINDOW IN POINTS; (In Epstein and Silva 2009 = 960 points)
fs= SAMPLE RATE (In Epstein and Silva 2009 = 48000)
cf=1000; (Center Frequency for analysis – this procedure has been demonstrated successfully for 1000 Hz)
FRatio = THE RATIO OF THE FREQUENCY BOUNDARIES IN THE ANALYSIS WINDOW RELATIVE TO THE CENTER FREQUENCY (CF) (In Epstein and Silva 2009 = 2)
```

```
CFLowerBound = cf / FRatio;
CFUpperBound = cf * FRatio;
```

```
Start_Time= DELAY TIME IN RECORDING TO BEGIN ANALYSIS IN ms (In Epstein and Silva 2009 = 10 ms)
```

```
(USE A HANNING WINDOW AS IN EPSTEIN AND SILVA 2009)
```

```
wind=Hanning(analen);
windcor=10*log10(sum(wind.^2)/analen);
f=linspace(0,25000,ffflen/2);
f(ffflen/2:ffflen)=linspace(25000,fs/ffflen, ffflen/2);
tn=linspace(0,0.02*(ffflen-1),ffflen);
flt1=(f > CFLowerBound).*(f < CFUpperBound );
flt1=flt1.';
```

```
filt=(f>(CFLowerBound)).*(f<(CFUpperBound));  
filt=filt.;
```

```
start=round(Start_Time*50);
```

MAKE THE FOLLOWING COMPUTATIONS FOR EACH LEVEL TO FIND THE REAL, POSITIVE PORTION OF THE CROSS-SPECTRUM

```
x1= LOAD TIME DOMAIN AVERAGE 1 HERE  
x2= LOAD TIME DOMAIN AVERAGE 2 HERE
```

```
r1=real(iffilt(filt1.*fft(x1)));  
r2=real(iffilt(filt1.*fft(x2)));  
e1=start-1;  
b2=start+size(wind);
```

```
r1(1:e1)=0;  
r2(1:e1)=0;  
r1(b2:fftlens)=0;  
r2(b2:fftlens)=0;
```

```
q1=r1;  
q2=r2;  
q1(start:b2-1)=q1(start:b2-1).*wind;  
q2(start:b2-1)=q2(start:b2-1).*wind;
```

```
ns=(q1-q2)/sqrt(2);
```

```
Q1=fft(q1);  
Q2=fft(q2);  
cs=real(Q1.*conj(Q2));  
cs=cs.*(cs>0)+1e-30*(cs<=0);
```

LOUDNESS ESTIMATE IS COMPUTER ON A LOG SCALE PROPORTIONAL TO LEVEL  
LoudnessEstimateForASingleLevel=10\*log10(sum(filt.\*cs))-10\*log10(analen\*fftlens)+69-windcor;