
HCS 7367
Speech Perception Lab

Dr. Peter Assmann
 Fall 2009

Signal-to-noise ratio (SNR)

- **SNR** = $20 \log_{10} [\text{rms}(\text{speech}) / \text{rms}(\text{noise})]$
 - Specified in decibels
 - When speech and noise have the same average rms level (SNR=0 dB), recognition scores are above 50% for listeners with normal hearing.

Signal-to-noise ratio (SNR)

- **SNR** = $20 \log_{10} [\text{rms}(\text{speech}) / \text{rms}(\text{noise})]$
 - Why is speech intelligible when the masker is presented at the same level as the speech (SNR=0 dB)?



Signal-to-noise ratio (SNR)

- Load a speech waveform
 - >> [sent1,rate]=wavread('sent1.mat');

Signal-to-noise ratio (SNR)

- Generate Gaussian (flat spectrum) noise
 - >> npts = length(sent1);
 - >> noise=randn(npts,1);

Signal-to-noise ratio (SNR)

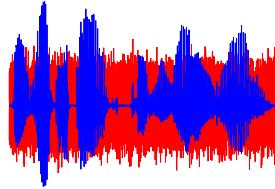
- Add the speech + noise at a 0 dB SNR.
- To do this you need to scale the noise to have the same **rms level** as the speech:
 - >> srms=rms(sent1);
 - >> nrms=rms(noise);
 - >> noise=noise.*(srms./nrms);

<http://www.utdallas.edu/~assmann/hcs7367/rms.m>

Signal-to-noise ratio (SNR)

- Plot the noise waveform in red and speech in blue:

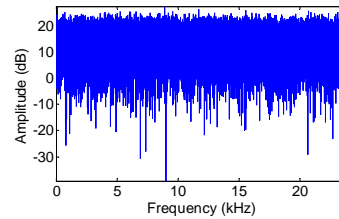
```
>> plot(noise, 'r')
>> hold on
>> plot(sent1, 'b')
>> axis off
```



Signal-to-noise ratio (SNR)

- Plot the spectrum of the noise:

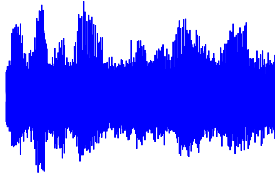
```
>> fp(noise,rate);
```



Signal-to-noise ratio (SNR)

- Mix the signals by summing the digital waveforms:

```
>> speech_plus_noise = sent1 + noise;
```



Signal-to-noise ratio (SNR)

- What about other SNRs (e.g. -10 dB SNR)?

```
>> srms=rms(sent1);
>> nrms=rms(noise);
>> dB=-10;
>> sf=10.^(dB/20);
>> noise=noise.*(srms./nrms)/sf;
>> speech_plus_noise = sent1 + noise;
```

Signal-to-noise ratio (SNR)

- How does white (flat spectrum) noise affect intelligibility?

- Consonant recognition (Miller and Nicely, JASA, 1955)
- Effects of set size (Miller, Heise and Lichten, J. Exp. Psychol. 1951)

Effects of noise on speech recognition

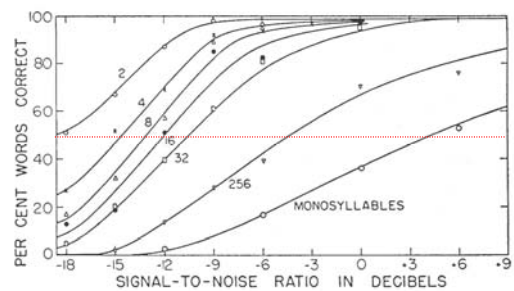


FIG. 2. Intelligibility of monosyllables as a function of the size of the text vocabulary. (Data are not corrected for effects of chance.)

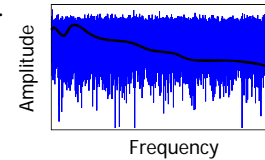
Source: Miller, Heise and Lichten, J. Exp. Psychol. 1951

Speech recognition in noise

- **Spectral properties of the noise:** white, pink, speech-shaped, competing speech, speech babble
- **Temporal properties of the noise:** steady vs. modulated or interrupted

White vs. speech-shaped noise

- White (flat-spectrum) noise has more energy at high frequencies, and therefore produces more masking of the higher formants and frication noises than speech-shaped noise.



Generating speech-shaped noise

- Compute the LTASS for the sentence, “*The watchdog gave a warning growl*”.
- ```
>> sent1_ltass=ltass(y,rate,twin,thop,nfft);
```

## Generating speech-shaped noise

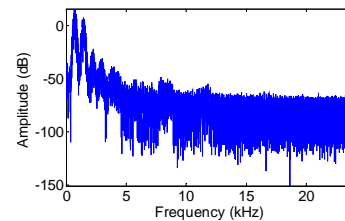
- Construct an FIR filter with this shape:
- ```
>> mag=10.^(sent1_ltass/20);  
>> freq=linspace(0,1,length(mag));  
>> b = fir2(512,freq,mag);
```

Generating speech-shaped noise

- Generate Gaussian white noise:
- ```
>> npts = length(sent1);
>> noise=randn(npts,1);
```
- Convolve the noise with the LTASS filter:
- ```
>> ssn=conv(noise,b);
```

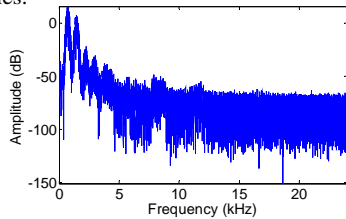
Generating speech-shaped noise

- Plot the spectrum of the SSN:
- ```
>> fp(ssn,rate);
```

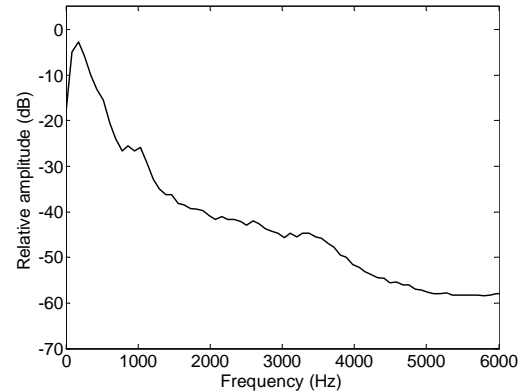


## Generating speech-shaped noise

- Problem: the “noise” resolves low-frequency harmonics. Solution: use large sample of speech with considerable variation in fundamental and formant frequencies.



LTASS for 256 HINT sentences (1 male talker)



## Generating speech-shaped noise

- Mix the signals by summing the digital waveforms:

```
>> speech_plus_ssn = sent1 + ssn;
```

## Appending zeros to a vector

- NOTE: When summing two signals of different length, Matlab generates an error:

??? Error using ==> plus

Matrix dimensions must agree.

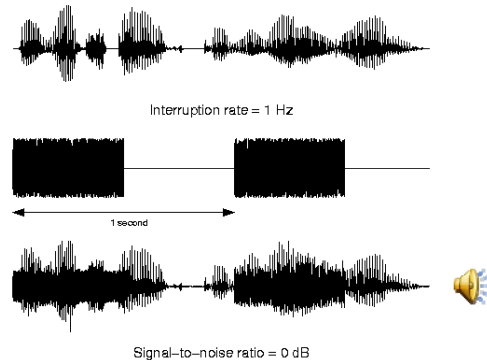
- One solution: pad the *shorter* vector with zeros. For example:

```
>> sent1(end+1:length(ssn))=0;
```

## Interrupted noise

- When the noise is **intermittent** rather than **continuous** there is a *release from masking*.
- The benefits of non-stationarity depend on the interruption rate and the *duty cycle* (on-off ratio) of the noise.

Miller and Licklider, J. Acoust. Soc. Am. 22: 167-173 (1950)



## Spectral analysis of speech

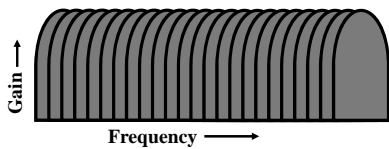
- *Why perform a frequency analyses of speech?*
  - Ear+brain carry out a form of frequency analysis
  - Relevant features of speech are more readily visible in the amplitude spectrum than in the raw waveform

## Spectral analysis of speech

- *But: the ear is not a spectrum analyzer.*
  - **Auditory frequency selectivity** is best at low frequencies and gets progressively worse at higher frequencies.

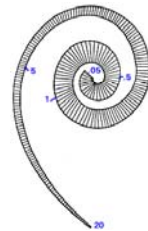
## Auditory filters

- Fletcher (1940) suggested that the peripheral auditory system could be modeled as a bank of linear bandpass filters with continuously overlapping center frequencies.



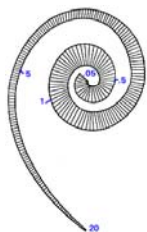
## Auditory filters

- Each point along the basilar membrane corresponds to a filter with a different center frequency, with center frequencies increasing roughly logarithmically from the apex to the base.



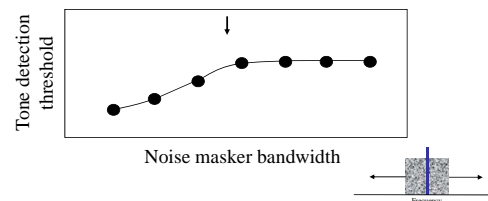
## Auditory filters

- Much of the length of the human basilar membrane is devoted to the lowest kHz (F1 range of speech) with the majority of nerve fibers responding best to low-to-mid-frequencies.



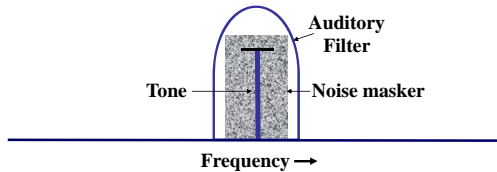
## Critical Bandwidth

- Fletcher (1940) band-widening experiment
  - The threshold for detecting a pure tone in the presence of a bandpass noise masker increases as the noise bandwidth increases, until the width of the band exceeds the *critical bandwidth* of the auditory filter.



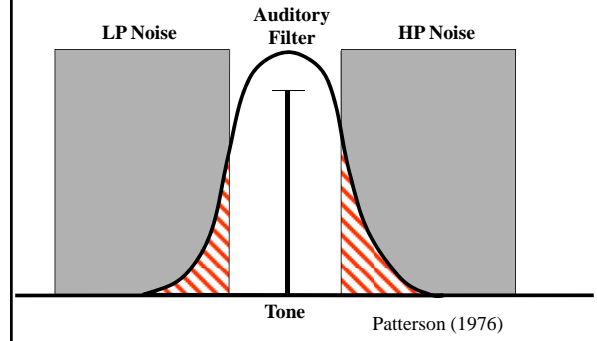
## Power spectrum model of masking

- Detection of probe tone in the presence of a noise masker depends of the relative power of probe and noise passed by the auditory filter centered on the tone (Patterson, 1976).



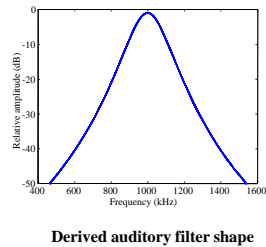
## Notched noise method

Measure tone threshold as a function of notch width

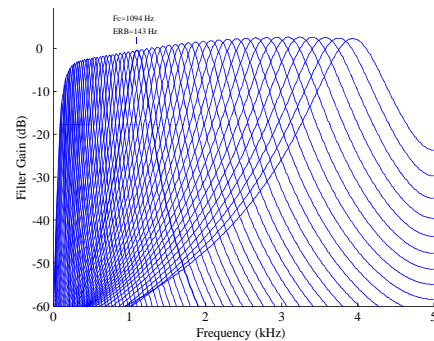


## Notched noise method

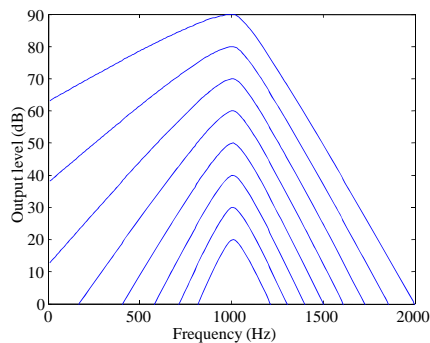
- Patterson (1976) estimated auditory filter shapes from the function relating **tone threshold** to **notch width**.
- The derived filters have a rounded top and steep skirts, with bandwidths 10-15% of filter center frequency.



## Auditory filter shapes as a function of frequency

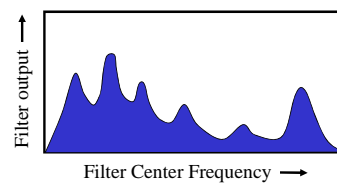


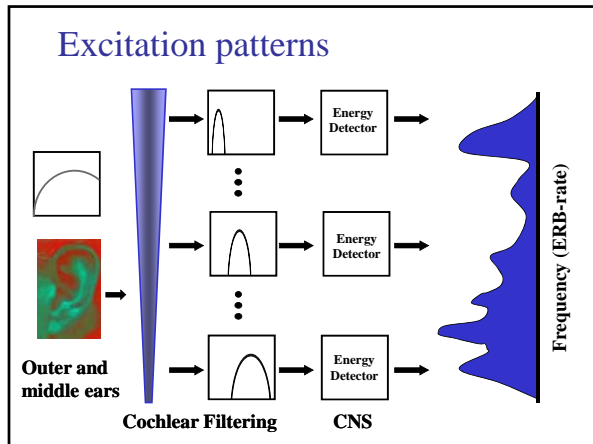
## Auditory filter shapes as a function of level



## Excitation patterns

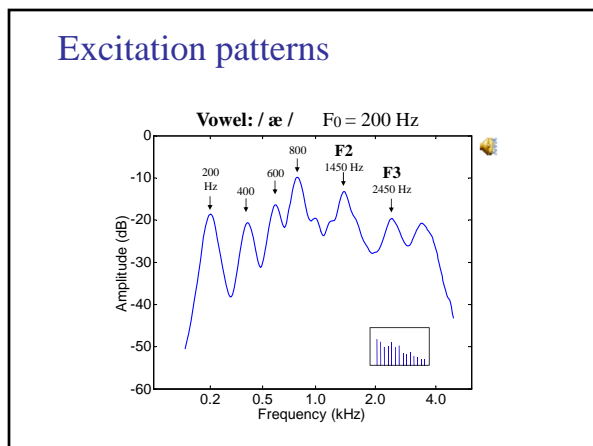
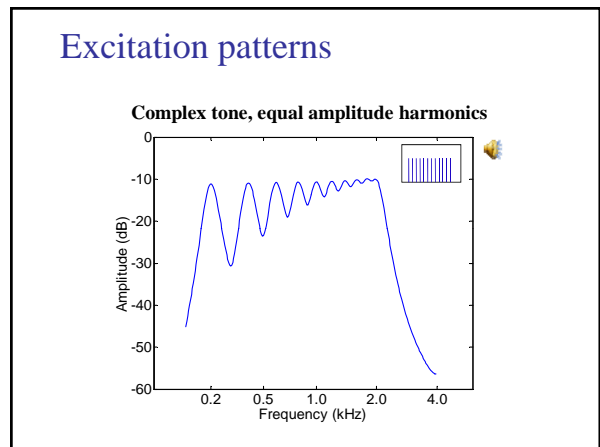
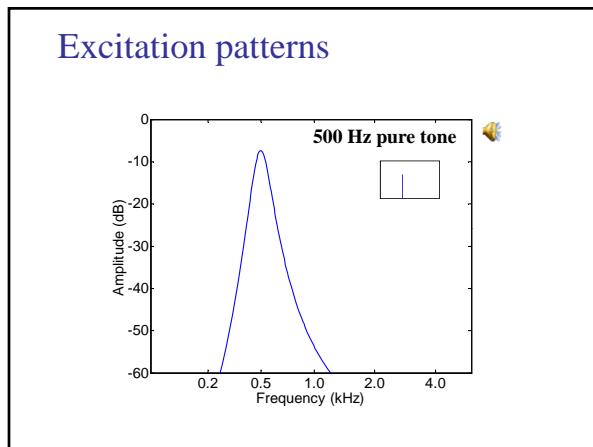
- **Auditory excitation patterns** show the composite output of a bank of simulated auditory filters as a function of filter center frequency.





### Excitation patterns

- Excitation patterns provide a good model of auditory frequency selectivity and masking: frequency components that are resolved by the auditory system produce distinct peaks in the excitation pattern.

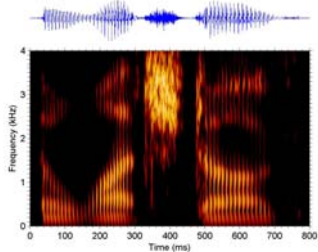


### Demo: harmonic synthesis

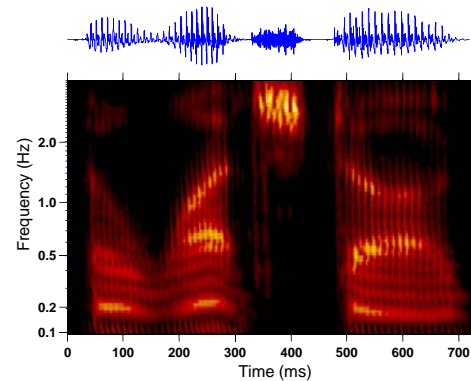
- Additive harmonic synthesis: vowel /i/ 🗣️
- Cumulative sum of harmonics: vowel /i/ 🗣️
- Additive synthesis: “wheel” 🗣️
- Cumulative sum of partials: 🗣️

## Speech spectrogram

- *Running amplitude spectra* (codes amplitude changes in different frequency bands over time).



## Auditory filterbank spectrogram



## Vowels in noise

- Extract a brief segment from near the midpoint of one of the 12 vowels recorded in lab 1.
- For each segment, create a sample of white noise of the **same duration** and **equal rms level** (0 dB SNR).
- Add the noise to the vowel and listen to the mixture. How does white noise at a 0 dB signal-to-noise ratio affect vowel quality?

## Vowel representations

- To generate the noise, use the built-in Matlab random number generator, `rand.m`. Since this generates numbers uniformly distributed between 0 and 1, you need to subtract 0.5 to distribute them around zero:
  - » `noise=rand( size(y) ) - 0.5;`
- To equate the rms levels of vowel and noise, use the function `rms.m` on the class web page:
  - » `noise = noise .* ( rms(y) ./ rms(noise) );`

## Vowels in noise

- Use the functions `fp.m` and `lpcp.m` to compute FFT and LPC spectra for the vowel alone and the vowel+noise mixture. (Note: `lpcp.m` incorporates 6 dB/oct pre-emphasis).
- What happens to the spectrum when white noise is added? How are the peaks in the LPC spectrum affected by the noise? Are some formant peaks and some vowels affected more than others? Why?

## Vowel representations

- Next, compute **auditory excitation patterns** for each vowel alone and vowel+noise mixture.
- Estimate the frequencies of resolved peaks in the excitation pattern. At low frequencies, these will correspond to **harmonics** of the fundamental. At higher frequencies, they may represent **formants** or clusters of formants.
- Compare the excitation patterns in quiet and in noise. What happens to the peaks?

## Excitation patterns

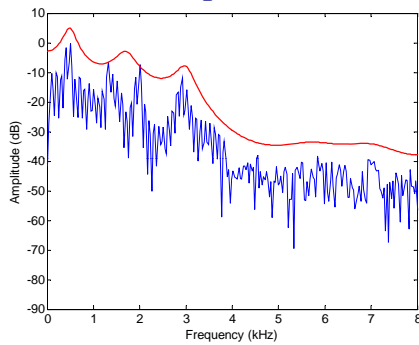
- To equate the rms level of the vowel and noise, use the function `rms.m` on the class web page:  
<http://www.utdallas.edu/~assmann/hcs7367/rms.m>
- To calculate the excitation patterns, use the function `gtep.m`:  
<http://www.utdallas.edu/~assmann/hcs7367/gtep.m>
- You'll also need Malcolm Slaney's Auditory Toolbox:  
<http://rvl4.ecn.purdue.edu/~malcolm/interval/1998-010/AuditoryToolbox.zip>  
<http://rvl4.ecn.purdue.edu/~malcolm/interval/1998-010/>

## Excitation patterns

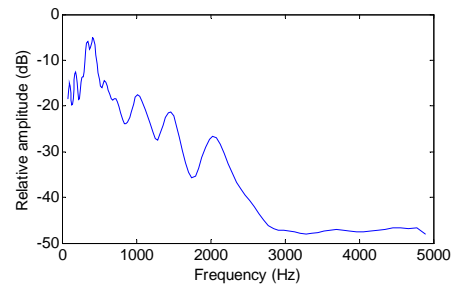
```
>> help gtep
gtep: Gammatone filter bank excitation pattern
Usage: [cf,expat]=gtep(y,rate,nfilt,flo,fhi,absflag);
cf: filter center frequencies on ERB-rate scale
expat: excitation pattern
y: waveform vector
rate: sample rate in Hz (default 10000 Hz)
nfilt: number of filter channels (default 128)
flo, fhi: CF limits (default 80 and rate/2=5000 Hz)
absflag: set to 0 if absolute threshold compensation is
NOT desired. (default 1)
See also: MakeERBFilters, ERBFilterBank
```

## LPC and FFT spectra

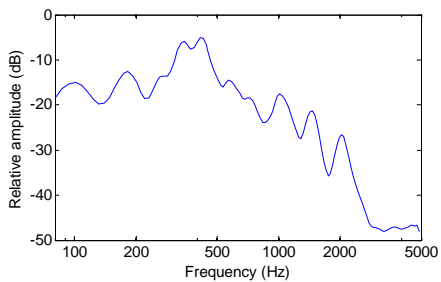
“herd”



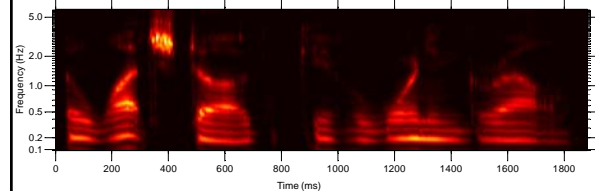
## Excitation pattern



## Excitation pattern: log frequency scale

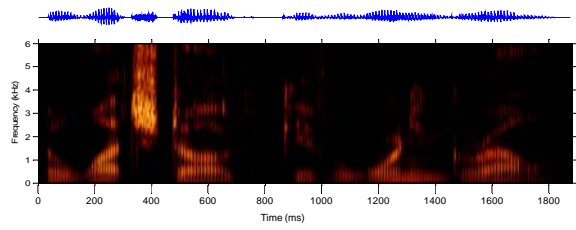


## Gammatone filter bank spectrogram



- Low-frequency features represent harmonics
  - High-frequency features represent formants
- ```
>> gtsfp(y,rate,128);
>> colormap(hot);
```

FFT spectrogram



Correlogram representation

- acgmovie_demo