Engin 100: Music Signal Processing Project #2: Technical Specifications

- Reverse-engineer touch-tone phone signals
- Technical specs for touch-tone synthesizer
- Technical specs for touch-tone transcriber
- Analysis of transcriber performance in noise

Reverse Engineer Touch-Tone Phone Signals

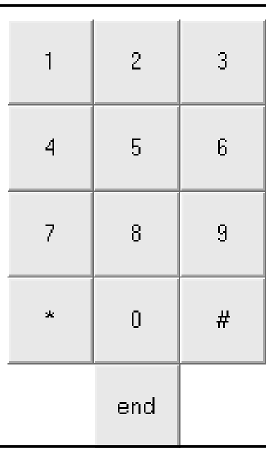
- Analyze spectra of touch-tone phone signals. Use abs(fft). Frequency pattern on keyboard.
- Synthesize (in Matlab) a touch-tone keypad. Straightforward: similar to Proj. #1 keyboard.
- Transcribe touch-tone signals→phone number. Look for specific frequencies-don't need fft.
- Analyze transcriber performance in white noise

Engin 100: Music Signal Processing Project #2: Technical Specifications

- Reverse-engineer touch-tone phone signals
- Technical specs for touch-tone synthesizer
- Technical specs for touch-tone transcriber
- Analysis of transcriber performance in noise

Analyze/Synthesize Touch-Tone Spectra

- 12 keys on phone keypad in file touch.wav. Sampled at 8192 Hertz; durations ½ second.
- 12 signals: Analyze each using abs(fft). Each signal is sum of several sinusoids.
- Relate frequencies to touch-tone keypad.
- Synthesize touch-tone keypad (cf. Proj. #1). Get phone keypad that looks like next slide.



Engin 100: Music Signal Processing Project #2: Technical Specifications

- Reverse-engineer touch-tone phone signals
- Technical specs for touch-tone synthesizer
- Technical specs for touch-tone transcriber
- Analysis of transcriber performance in noise

Transcribe Touch-Tone Signals

- Input: Touch-tone signals stored in touch.wav
- Output: String of numbers: 8 6 7 5 3 0 9 (no -)
- Do not need * or # (not in phone numbers).
- Do not need abs(fft): Find specific frequencies: >>Y=(X*cos(2*pi*[0:N-1]'*[F1…FM]/S)).^2 + (X*sin(2*pi*[0:N-1]'*[F1…FM]/S)).^2
- X=signal (row);N=length(X);S=sampling rate;
- [F1…FM]=vector of specific frequencies (<9!)

Transcribe Touch-Tone Signals

- Get: Y=row vector of M numbers. I=1 to M:
- Interpret: If Y(I)<threshold, FI not present.
- Interpret: If Y(I)>threshold, FI IS present.
- Best: [Z,I]=max(Y);→maximum location I
- Decode: Frequencies present→phone digit.
- Pattern: Don't need a lot of if statements. Can do quickly using rem(I+3*(J-1),11)
- Output: Phone digit for each segment of X.

Summary of Specifications

- Length of each phone digit known: ¼ second.
- Sampling rate known: 8192 samples/second.
- Touch-tone signal written to file touch.mat.

- DON'T use abs(fft): too much computation!
- DON'T use many if: too much computation!
- Output: String of phone digits without hyphen.

Engin 100: Music Signal Processing Project #2: Technical Specifications

- Reverse-engineer touch-tone phone signals
- Technical specs for touch-tone synthesizer
- Technical specs for touch-tone transcriber
- Analysis of transcriber performance in noise

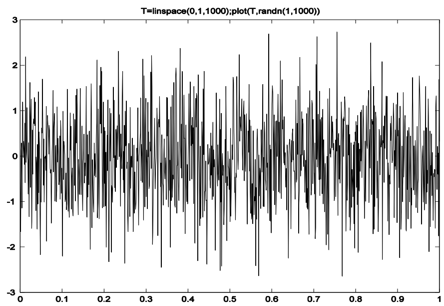
Transcriber Performance in Noise

- Phone signals (whether landline or wireless) have noise present. Haven't considered yet.
- Noise: What exactly is noise?
- Performance: How well does your transcriber work when noise is present (as in real world)?
- Figure of merit: Numerical measurement of performance of a system (detector, estimator).

Zero-Mean Gaussian White Noise

- Good model for many actual sources of noise.
- At each time t_0: $n(t_0)$ Gaussian distribution.
- At any 2 times t_0 and t_1, no matter how close:
- $n(t_0)$ and $n(t_1)$ are completely uncorrelated: $n(t_0)$ value has no influence on $n(t_1)$ value.
- See next slide for a typical sample function.

Sample Function: White Noise



Touch-Tone Performance Measure

- Noise level rises→transcriber gets digits wrong.
- What counts: noise level relative to signal level.
- Signal-to-noise ratio (SNR) in decibels (dB):
  SNR=10$\log_{10}$[sum(X.^2)/sum(N.^2)] where:
  X=vector of signal & N=vector of noise values.
- Performance plot: SNR is on horizontal axis; the error rate (percentage) is on vertical axis.

Touch-Tone Performance Measure

- Transcriber gets some digits wrong.
- Error rate: Fraction of digits gotten wrong.
- Need many digits to get accurate measure.
- At each SNR: Use 100 digits, count #wrong.
- Random digits vs. same digit each time?
- Plot: error rate vs. SNR for several SNRs.
- Random noise: SNR varies with noise values?

Performance: What to do?

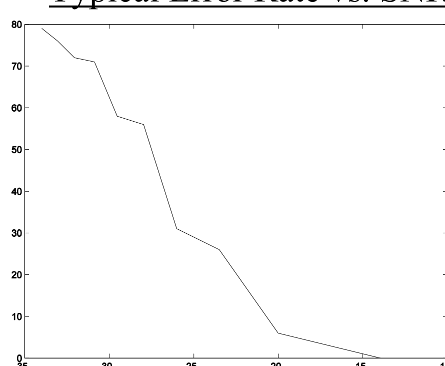
- Any transcriber customer will want to see your plot of error rate vs. SNR. Threshold.
- What transcriber error rate is acceptable?
- What noise level can transcriber tolerate?
- What to do if need better performance?
- Error-correction: Digital Comm: EECS 455

Outline of a Matlab Program for Transcriber Performance in Noise

clear; X=[signal for clicking on “1”]; for S=1:10; NS=0; ER=0;
for I=1:100; N=5*S*randn(1,2048); Y=X+N; NS=NS+sum(N.^2);
ER=ER+[0 if transcriber outputs “1”; 1 otherwise]; end; E(S)=ER;
SNR(S)=10*log10(sum(X.^2)/(NS/100)); end; plot(SNR,E)

What’s going on here? I’m using 2 loops (ugh!) for clarity.
Use the signal for clicking on “1” each time; makes things easier.
Since noise is white, it has the same strength at all frequencies.
100 trials (I=1:100) at each of 10 (S=1:10) different noise strengths.
Count #errors (ER) in 100 trials; this is error rate as a percentage.
NS/100=average noise power over 100 trials; sum(X.^2)=signal power.
E(S) and SNR(S) are error rate and SNR at noise level specified by S.

Typical Error Rate vs. SNR Plot



Note the threshold at an SNR of –15 dB. Above this, the noise is swamping the signal, but your transcriber still works!