ECE 421
Introduction to Signal Processing

Dror Baron
Associate Professor
Dept. of Electrical and Computer Engr.
North Carolina State University, NC, USA
Project 1 - Bandpass Sampling
Modulating sinusoids

- Input $x_{in}(t) = \cos(2\pi F_t)$
  - $F \in [F_l, F_h]$

- Modulate by oscillating signal
  $x_{os}(t) = \cos(2\pi F_c t)$

- Output $x_{fm1}(t) = 0.5 \cos(2\pi (F_c + F)t) + 0.5 \cos(2\pi (F_c - F)t)$
  - Sample at rate $F_{sa}$ resulting in $x_{fm1}(n) = x_{fm1}(t=n/F_{sa})$

- Demodulator $x_{fm2}(t) = x_{fm1}(t)x_{os}(t)$
Lowest sampling rate?

- Nyquist relies exclusively on highest frequency, $F_h$

- Consider midpoint, $F_{\text{mid}} = 0.5(F_h + F_l)$

- If $B = F_h - F_l \ll F_{\text{mid}}$, sampling at Nyquist seems wasteful

- Alternate approach:
  - Sample at $F_{sa} = F_{\text{mid}}/k$ for some integer $k$
  - $F_{\text{mid}}$ gets aliased-folded to DC (zero frequency)
  - If $F_{sa} > B$, in principle have enough information to recover anything in band $[F_l, F_h]$
Project 2 – Audio Equalization
Audio signals

- Audio signals have been processed digitally for decades due to relatively low bandwidths
  - Phone system samples speech signals @ 8 KHz (4 KHz bandwidth)
  - CDs (early 80’s) sample music @ 44.1 KHz (22 KHz bandwidth)
    - People hear 20 Hz – 20 KHz

- Different frequency components correspond to different sounds
  - Attenuating/amplifying different bands may alter (improve?) listening experience
  - Audio equalizers (stereo systems) allow users to adjust bands
Audio signals

- Design finite impulse response filters (MATLAB’s fir1 command) to partition audio signal into 5 bands
- Multiply each band by gain ($G_1$ through $G_5$)
- Sum them up
- Listen to output
Project 3 – Group Testing
Motivation

- We have N objects
- Among them, K << N special objects
- Want to identify (test) special ones

- **Example1**: testing for defective car parts
  - Objects are car parts
  - Want to identify bad ones

- **Example2**: testing for fake coins

- **Example3**: testing for disease
What is group testing?

- How many measurements/tests $M$ do we need to identify $K << N$ special objects?

- Conventional approaches uses $N$ individual tests

- Group test identifies whether group has any special objects

**Example:** weigh $B$ coins
  - Weight fine? None of them are special
  - Else need more information
Adaptive group testing

- Partition into batches of size $B$

- Two parts
  - Part1: test $N/B$ batches
  - Part2: if batch contains special object(s), retest individually

- Adaptive part designs Part2 based on Part1

- Having 2 parts doubles the latency
Non-adaptive group testing

- Non-adaptive meaning no second part

Notation / problem setting:
- $x$: length-$N$ vector, $x(n)=1$ for special object, else 0
- $A$: binary matrix, $M$ rows, $N$ cols
  - Row $m$ ($A_m$) corresponds to measurement $m$
  - Col $n$ ($A^n$) corresponds to object $n$ in $x(n)$
- $y=Ax$: measurements vector; $y(m)=1$ if $A_m x > 0$, else 0.

- **Goal**: go from $(y,A)$ to $x$
  - Related to compressed sensing (modern SP topic)
More

- If $y_m=1$, multiple objects $n$ measured by $y_m$ could be special
  - Matrix $A$ must test each object $n \in \{1, \ldots, N\}$ multiple times
  - Doing so provides more tests to evaluate each $x_n$

- Group testing with noisy measurements
  - If $y(m)$ should be 1 but is incorrectly 0, will (mistakenly) think none of the objects it tests are special
  - Else, if $y(m)$ should be 0 but is incorrectly 1, will search for (nonexistent) special object

- Measurement noise is active research area; need more measurements $M$
Tasks

- **Adaptive group testing**
  - Will empirically compute optimal batch size $B$
  - Simulate $N$ coins (large $N$, $10^5$ or more)
  - Partition into batches of size $B$ (multiple values of $B$)
  - Take $N/B$ measurements in Part 1
  - Part 2 requires $B \cdot \#\text{positives}$ more measurements
  - Plot total $\#\text{measurements}$ as function of $B$
  - Determine optimal $B$ empirically

- **Non-adaptive**
  - We provide $(A,y)$; you find $x$
  - Must solve this for student id’s of all group members
  - Solving manually is feasible
  - Automating it is more “educationally valuable” 😊
Project 4 – Image Compression
Need data compression for images/video

- Images/video require tons of data
  - \(~80\%\) of residential Web traffic in 2019
  - Zoom still often needs hours to process files (March 2021)

- Lossless data compression
  - Can recover the input signal perfectly (\(y=x\))
  - Requires lots of data (e.g., 4 bits/pixel for images)

- Losslessly compressed video would require many GB per minute \(\rightarrow\) storage / communication bottleneck
Lossy data compression

- Can drastically reduce communication / storage bandwidth using lossy data compression

- Output $y$ no longer equals $x$

- Small distortion between $x$ and $y$ allows significant savings

- Rate distortion theory – fundamental trade-off between coding rate and distortion (input $x$, output $y$)
Image compression with JPEG

- Partition input image into 8x8 blocks
- Compute discrete cosine transform (DCT) of each block
- Quantize (discretize) DCT coefficients to discrete levels

- Zigzag scan
  - Low freqs first

- Run length coding
  - Compresses runs of zeros
Notation for Project 5

- Input $x \in \mathbb{R}^{M \times N}$
  - $M$ rows, $N$ columns
  - Gray scale (pixel intensity in $\mathbb{R}$) instead of color ($\mathbb{R}^3$)

- Encoder $f$: $\mathbb{R}^{M \times N} \rightarrow \{0,1\}^+$

- Decoder $g$: $\{0,1\}^+ \rightarrow \mathbb{R}^{M \times N}$
  - $\hat{x} = g(f(x))$ (output $\hat{x}$; can call it $y$)

- Distortion $D(x, \hat{x}) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} (x_{m,n} - \hat{x}_{m,n})^2$

- Coding rate $R = \frac{|b|}{MN} = \frac{|f(x)|}{MN}$
  - Distortion and rate both normalized per-element
Algorithmic steps

- **Partition into patches**
  - Each patch comprised of $P$ (typically 8x8=64) pixels
  - $MN$ pixels in $x \rightarrow MN/P$ patches
  - Denote patch $p \in \{1, 2, ..., MN/P\}$ by $x_p$

- **Discrete cosine transform (DCT)** $X_p = \text{DCT}(x_p)$
  - DCT is sparsifying transform - most DCT coefficients are small; large coeffs are sparse (infrequent)
  - Large coeffs tend to cluster at small $M$, $N$ (lower frequencies)
  - Example on page 3 of Project 5

- **Quantization** $q(\hat{X}_p) = \text{round}(X_p/\Delta)$, $\hat{X}_p = \Delta \cdot q(\hat{X}_p)$
  - Example on page 4
Algorithmic steps

- **Partition into patches**
  - Each patch comprised of \( P \) (typically 8x8=64) pixels
  - \( MN \) pixels in \( x \) \( \rightarrow \) \( MN/P \) patches
  - Denote patch \( p \in \{1,2,...,MN/P\} \) by \( x_p \)

- **Discrete cosine transform (DCT)** \( X_p = \text{DCT}(x_p) \)
  - DCT is sparsifying transform - most DCT coefficients are small; large coeffs are sparse (infrequent)
  - Large coeffs tend to cluster at small \( M, N \) (lower frequencies)
  - Example on page 3 of Project 5

- **Quantization** – \( q(\hat{X}_p) = \text{round}(X_p/\Delta), \hat{X}_p = \Delta \cdot q(\hat{X}_p) \)
  - Example on page 4
Algorithmic steps

- **Partition into patches**
  - Each patch comprised of \( P \) (typically \( 8 \times 8 = 64 \)) pixels
  - \( MN \) pixels in \( x \) \( \rightarrow \) \( MN/P \) patches
  - Denote patch \( p \in \{1,2,...,MN/P\} \) by \( x_p \)

- **Discrete cosine transform (DCT)** \( X_p = \text{DCT}(x_p) \)
  - DCT is sparsifying transform - most DCT coefficients are small; large coeffs are sparse (infrequent)
  - Large coeffs tend to cluster at small \( M, N \) (lower frequencies)
  - Example on page 3 of Project 5

- **Quantization** \( q(\hat{X}_p) = \text{round}(X_p/\Delta), \hat{X}_p = \Delta \cdot q(\hat{X}_p) \)
  - Example on page 4
Encoding step

- JPEG uses zigzag scan, run length coding, and Huffman coding
  - Huffman codes convert individual symbols to bit strings; then concatenate them

- *Arithmetic codes* convert probability $p$ to roughly $-\log_2(p)$ bits
  - Intuition: each bit can separate between 2 possible messages
  - $K$ bits equivalent to $2^K$ messages
  - Consider short string $\alpha\beta$; $-\log_2(\Pr(\alpha\beta))=-\log_2(\Pr(\alpha))-\log_2(\Pr(\beta))$
  - Many symbols encoded $\Rightarrow$ total length = sums(lengths($\Pr(q(coeffs))$))
Project 6(?) – Frequency Modulation
AM vs. FM

- **Amplitude modulation (AM)**
  - Input $x_a(t)$
  - Modulated (multiplied) by carrier, $x_{os}(t) = \cos(2\pi F_c t)$
  - Two copies of spectrum around $\pm F_c$

- **Frequency modulation (FM)**
  - Input becomes phase in oscillator generating carrier freq
  - $y(t) = \cos(2\pi F_c t + \beta x_a(t))$
  - Modulation index $\beta$

- AM and FM both commonly used for radio transmissions
- Also used in digital communication systems
- Spectral characteristics of FM more complicated
Small modulation index

- Recall $y(t) = \cos(2\pi F_c t + \beta x_a(t))$
- Sinusoidal input, $x_a(t) = \sin(2\pi F_m t)$

- Consider small modulation index $\beta$

  \[
  \cos(2\pi F_c t + \beta x_a(t)) = \cos(2\pi F_c t + \beta \sin(2\pi F_m t)) \\
  = \cos(2\pi F_c t) \cos(\beta \sin(2\pi F_m t)) - \sin(2\pi F_c t) \sin(\beta \sin(2\pi F_m t))
  \]

- Note $\sin(x) \approx x$ for small $x \rightarrow \sin(\beta \sin(2\pi F_m t)) \approx \beta \sin(2\pi F_m t)$

  \[
  \cos(2\pi F_c t + \beta x_a(t)) \approx \cos(2\pi F_c t) - \sin(2\pi F_c t) \beta \sin(2\pi F_m t)
  \]

- Will characterize spectrum using *Bessel functions*
Retired Projects
Project 2 - DTMF Detection
What is DTMF?

- Dual tone multi frequency
  - Baseband system for communicating between phone equipment
  - Developed in analog era @ audio frequencies

- Encodes phone buttons as 2 tones

- Each row/column correspond to tone

- Each button touched results in tone
  - Lasts 0.5 sec; sampled 4k times @ 8k samples/sec
  - Example: 8 digit number requires 32k samples
How do we detect DTMF?

- How does DTMF look in time domain?
  - Sum of two tones (sinusoids)
  - Might appear periodic
  - Unclear how to detect

- Easier in frequency domain
  - Two pairs of symmetric spikes
  - Negative/positive pair per tone

- Detect DTMF in frequency domain
Denoising and Project 4
Where does denoising appear?

- Setting: \( y(t) = x_a(t) + n(t) \)
  - Noise \( n(t) \) could be Gaussian “bell curve” noise (thermal noise)
  - Speckle effects \( \rightarrow \) saturation \( \rightarrow \) sensor returns min/max values
    - Called “salt & pepper” noise

- Audio processing – record is scratched
  - Large amplitude noise resembling salt & pepper

- Image processing (Project 4)
  - From 1D signal to 2D image

- Modern example:
  - In iterative estimation algorithms, \( n(t) \) is estimation error
  - Error \( n(t) \) becomes smaller over iterations
How can we denoise?

- Need to start somewhere – exploit *structure* in input $x_a(t)$

**Types of structure?**

- *Band limited* signals – signal occupies few freqs w/large coeffs; broadband noise w/small coeffs
- Knowledge about how much energy (on average!) signal/noise have in different spectral bands
  - More signal $\rightarrow$ barely attenuate
  - More noise $\rightarrow$ attenuate a lot
- Beyond Fourier coefficients and frequency...
  - *Piecewise constant* signal $x_a(t)$ jumps at times $< T_{-1} < T_0 < T_1 < ...$
Median filters

- We saw how moving average filters are simple low passes, and can be used to denoise Gaussian noise.

- Noise with outliers (e.g., salt & pepper noise) does not work well with standard low passes.

- Non-linear median filters better for outliers.

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(n)</td>
<td>12</td>
<td>13</td>
<td>11</td>
<td>105</td>
<td>10</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Mean</td>
<td>...</td>
<td>12</td>
<td>43</td>
<td>42</td>
<td>43</td>
<td>12</td>
<td>...</td>
</tr>
<tr>
<td>Median</td>
<td>...</td>
<td>12</td>
<td>13</td>
<td>11</td>
<td>14</td>
<td>12</td>
<td>...</td>
</tr>
</tbody>
</table>
Project 4

- Specifically deals with 2D images and two types of noise
  - Gaussian noise
  - “Salt & pepper” noise

- Main concepts in project:
  - From 1D signals to 2D images
  - Linear filters (we’ve seen these) good for Gaussian-like noise
  - Non-linear median filters good for salt & pepper
    - For each 3*3 patch surrounding pixel in middle, take median of 9 numbers in patch
      \[
      \begin{pmatrix}
      1 & 4 & 2 \\
      4 & 37 & 8 \\
      3 & 5 & 4
      \end{pmatrix}
      \]
    - Median \(\begin{pmatrix}
      1 & 4 & 2 \\
      4 & 37 & 8 \\
      3 & 5 & 4
      \end{pmatrix}\) = 4

- Medians less sensitive to outliers!
What about noise?

- Noise (e.g., static) typically added to signal

- Time domain – signal/noise might be comparable in magnitude

- Frequency domain perspective
  - Energy of signal concentrated into 4 prominent spikes
  - Noise energy modest as before
  - Signal (@ those spikes) larger than noise