ESE 531: Digital Signal Processing

Lec 23: April 20, 2017

Compressive Sensing

Previously

- Today
  - DTFT, DFT, FFT practice
  - Compressive Sampling/Sensing

Example 2:

A sequence $x = \{x[n], n = 0,1,\dots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? *(Show your analysis and explain clearly.)*

Example 2:

A sequence $x = \{x[n], n = 0,1,\ldots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? *(Show your analysis and explain clearly.)*

(b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m,\quad m = 0,1,\ldots,M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N},\quad m = 0,1,\ldots,M-1.$$

Here $M = 2^{\mu} \ll N = 2^{\nu}$

A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once* or *more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N.

Example 2:

A sequence $x = \{x[n], n = 0,1,\dots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? *(Show your analysis and explain clearly.)*

(b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m, \quad m = 0,1,\dots,M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N}, \quad m = 0,1,\dots,M-1.$$

Here $M = 2^{\mu} \ll N = 2^{\nu}$

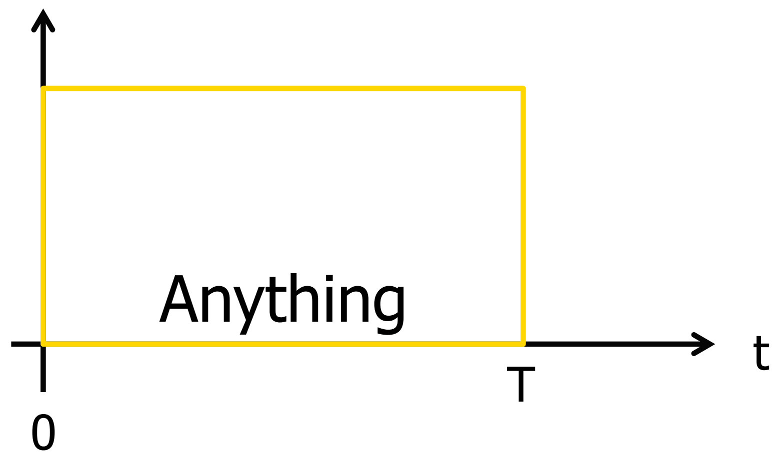
A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once* or *more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N.

Does your result change if extra multiplications outside of FFTs are *not* allowed?

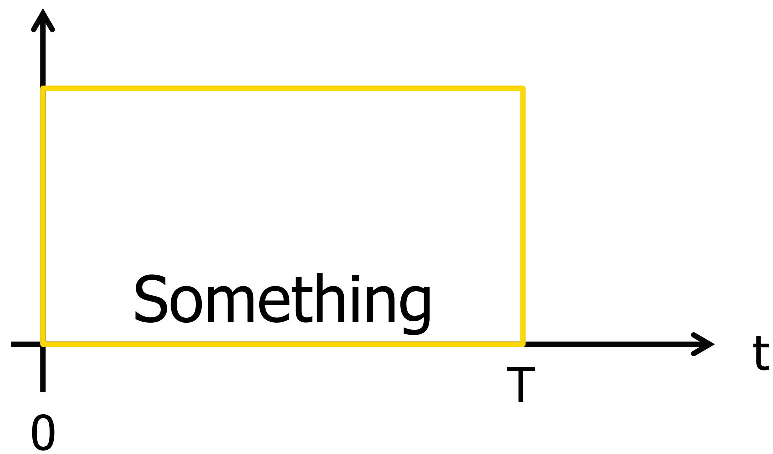
Compressive Sampling

Compressive Sampling



- What is the rate you need to sample at?
  - At least Nyquist

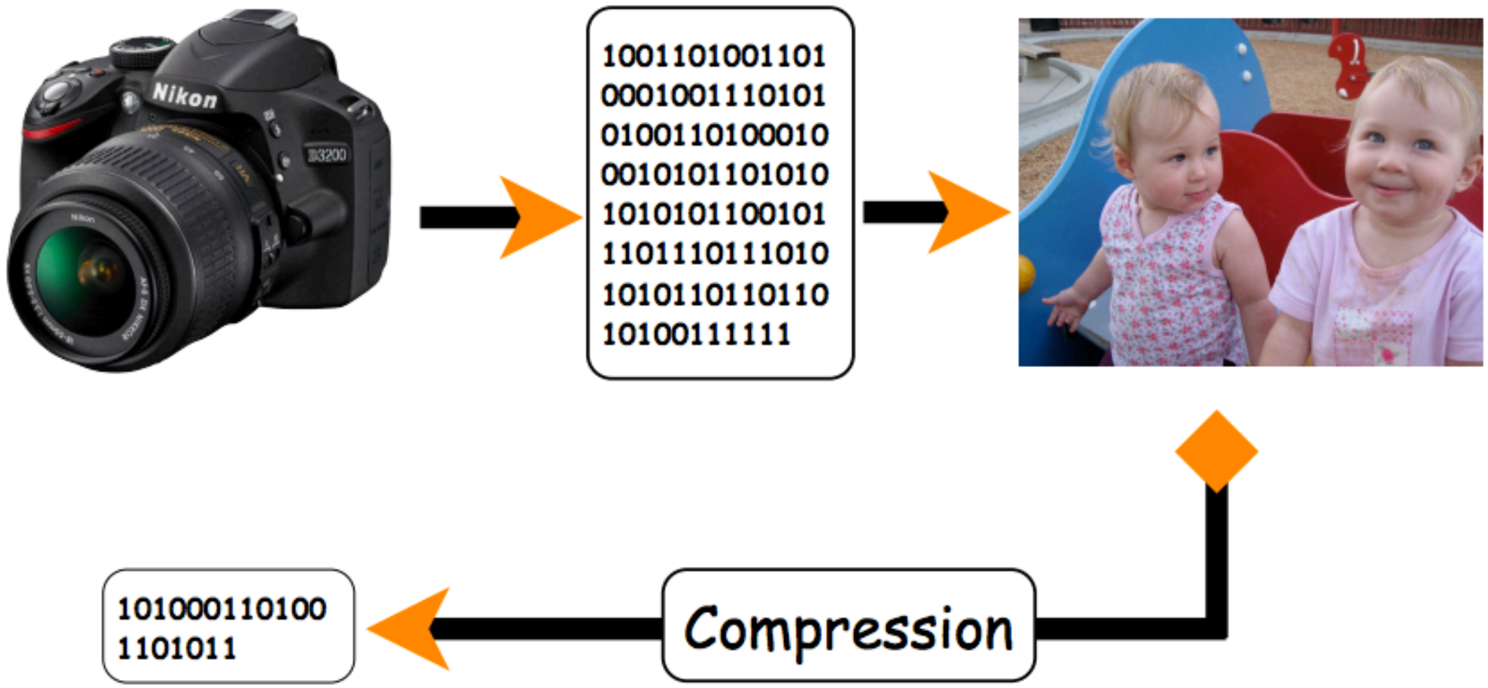
Compressive Sampling



- What is the rate you need to sample at?
  - Maybe less than Nyquist…

First: Compression

- Standard approach
  - First collect, then compress
    - Throw away unnecessary data



First: Compression

- Examples
  - Audio – 10x
    - Raw audio: 44.1kHz, 16bit, stereo = 1378 Kbit/sec
    - MP3: 44.1kHz, 16 bit, stereo = 128 Kbit/sec
  - Images – 22x
    - Raw image (RGB): 24bit/pixel
    - JPEG: 1280x960, normal = 1.09bit/pixel
  - Videos – 75x
    - Raw Video: (480x360)p/frame x 24b/p x 24frames/s + 44.1kHz x 16b x 2 = 98,578 Kbit/s
    - MPEG4: 1300 Kbit/s

First: Compression

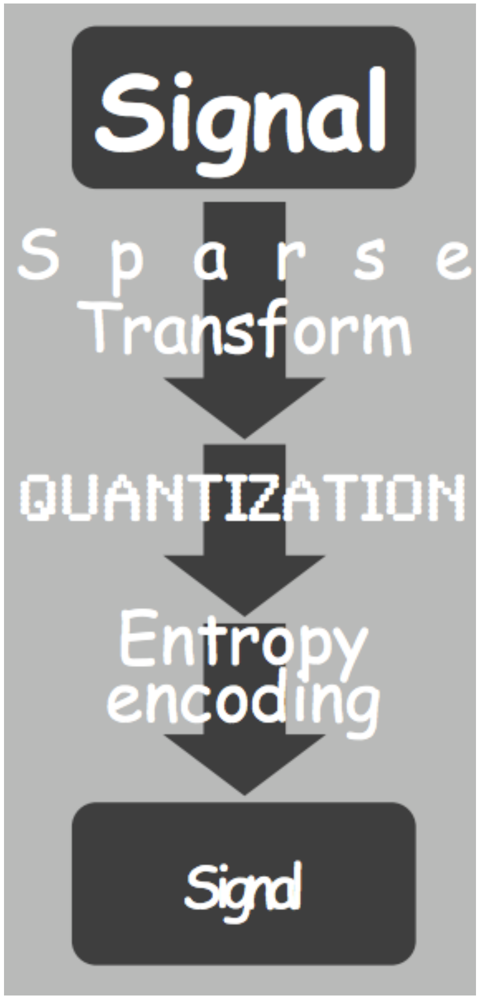
- Almost all compression algorithm use transform coding
  - mp3: DCT
  - JPEG: DCT
  - JPEG2000: Wavelet
  - MPEG: DCT & time-difference

First: Compression

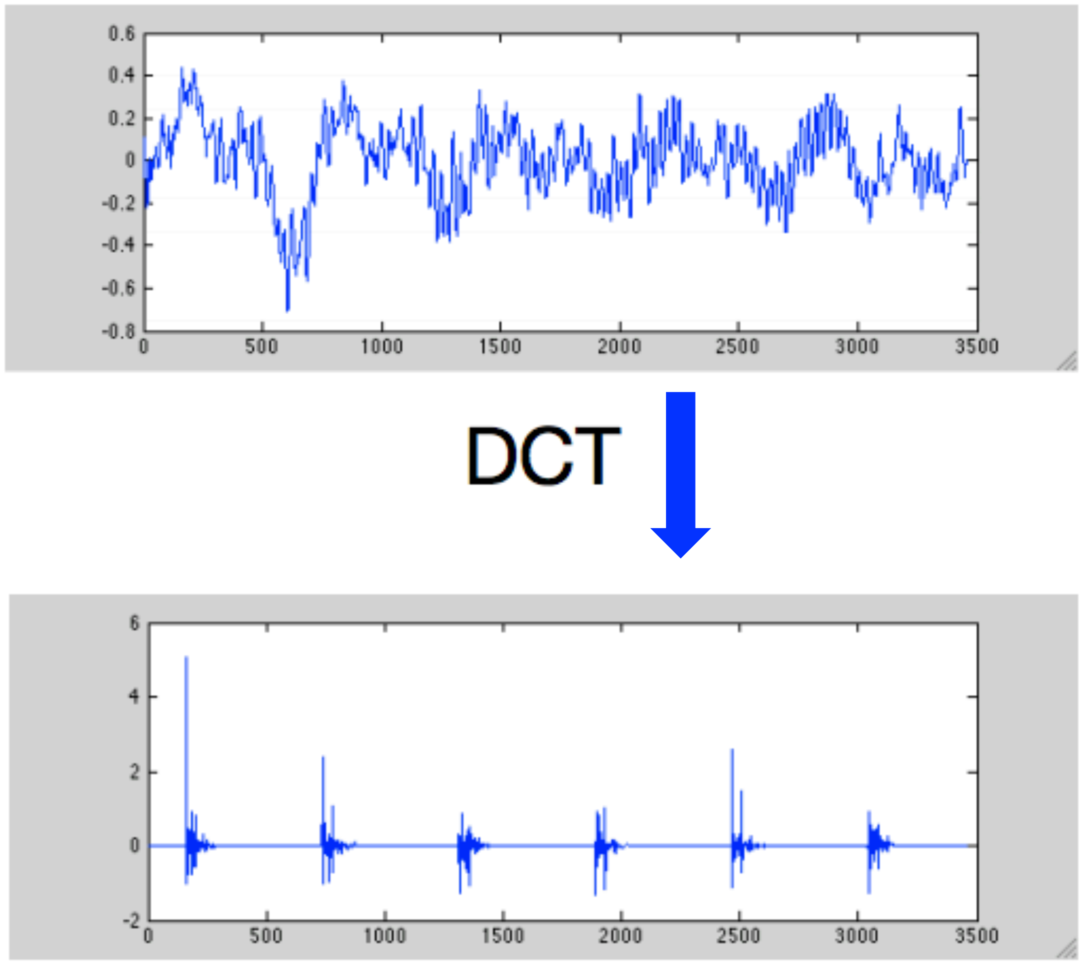
- Almost all compression algorithm use transform coding
  - mp3: DCT
  - JPEG: DCT
  - JPEG2000: Wavelet
  - MPEG: DCT & time-difference

First: Compression

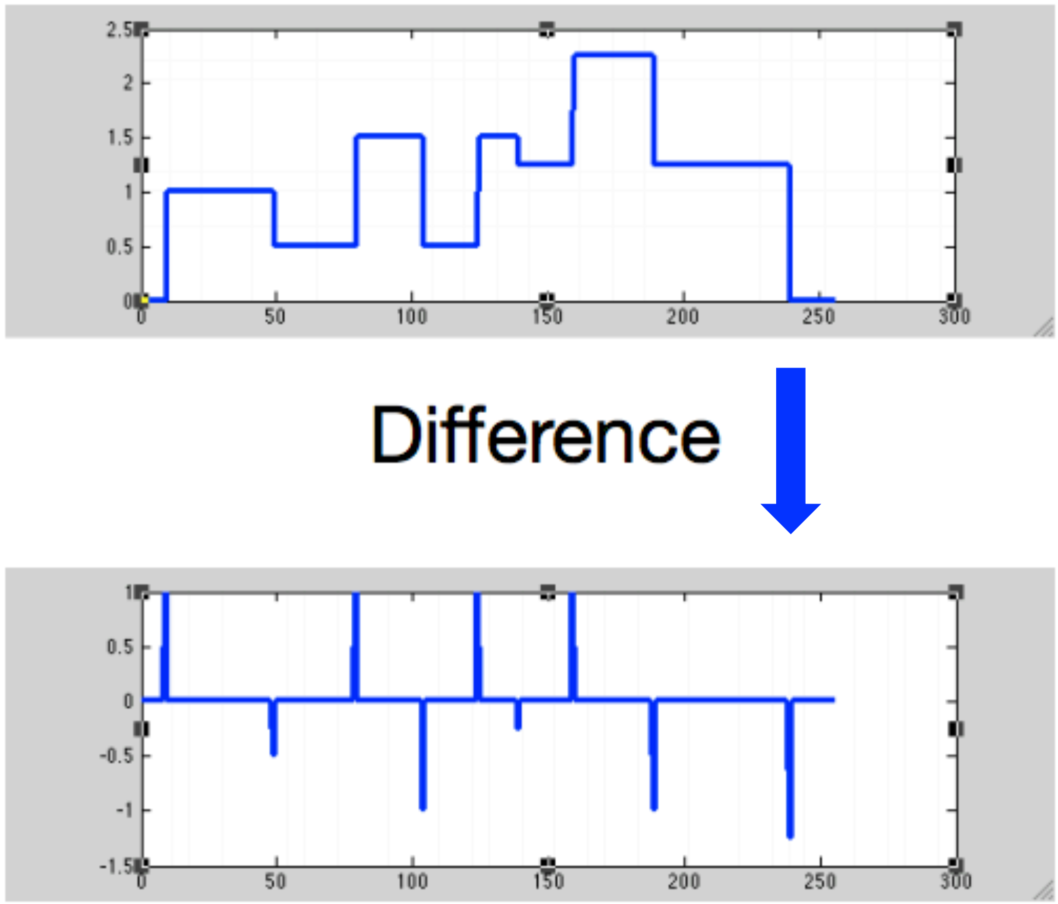
- Almost all compression algorithm use transform coding
  - mp3: DCT
  - JPEG: DCT
  - JPEG2000: Wavelet
  - MPEG: DCT & time-difference



Sparse Transform

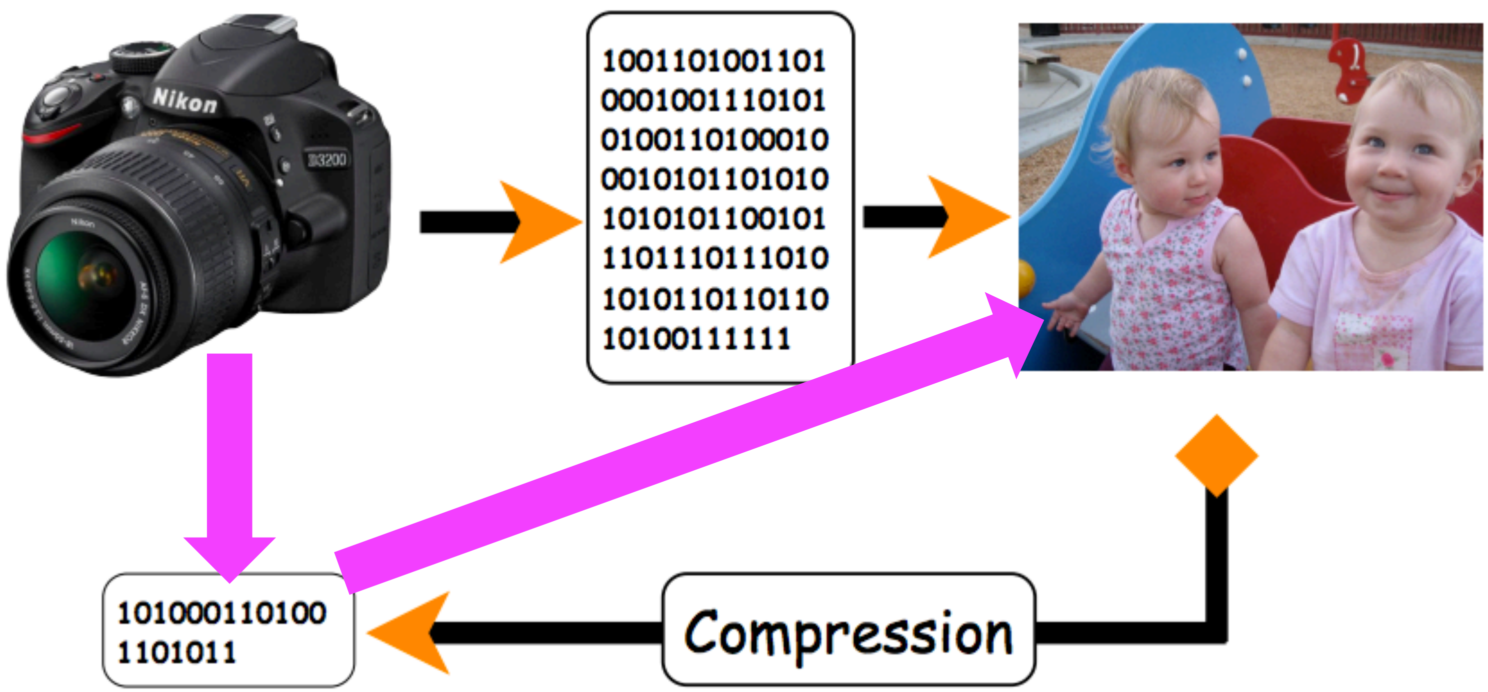


Sparse Transform



Compressive Sensing/Sampling

- Standard approach
  - First collect, then compress
    - Throw away unnecessary data



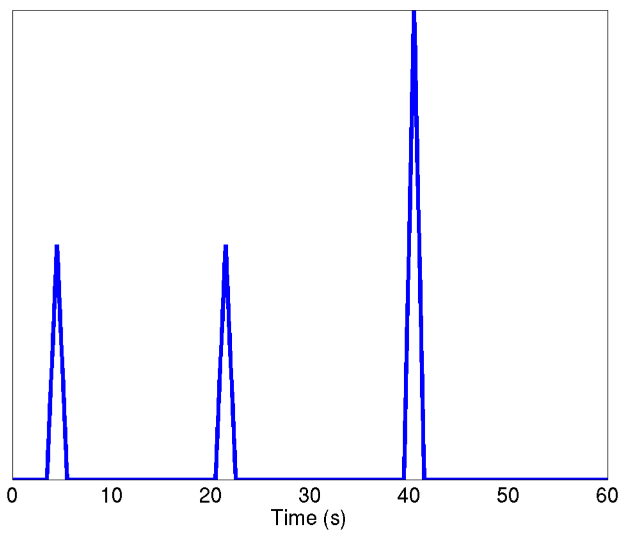
Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

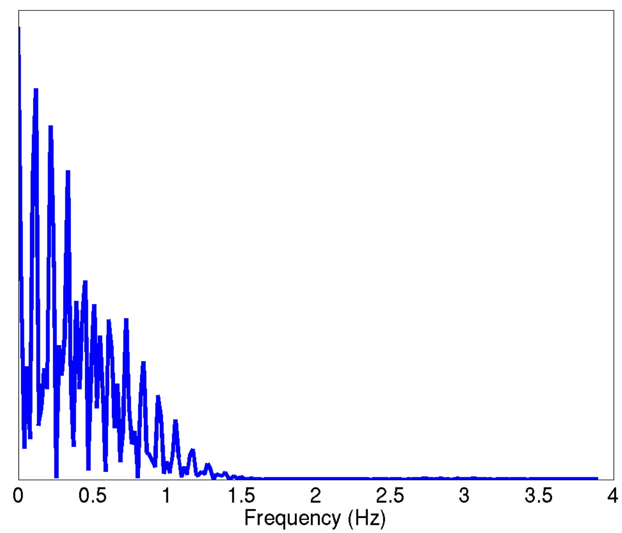
Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

Sparse signal in time



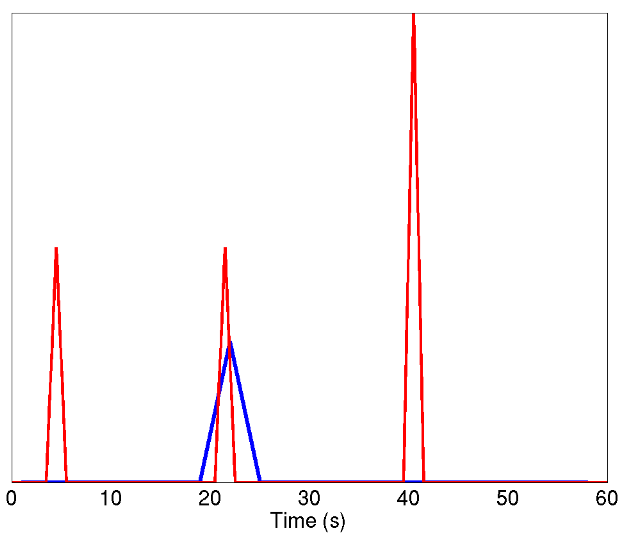
Frequency spectrum



Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

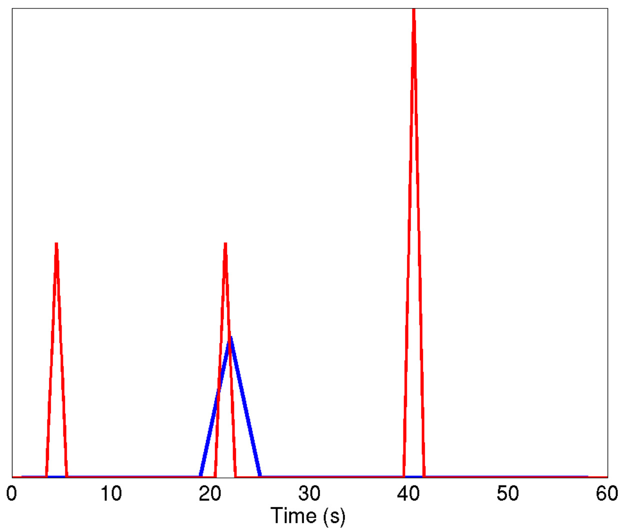
Undersampled in time

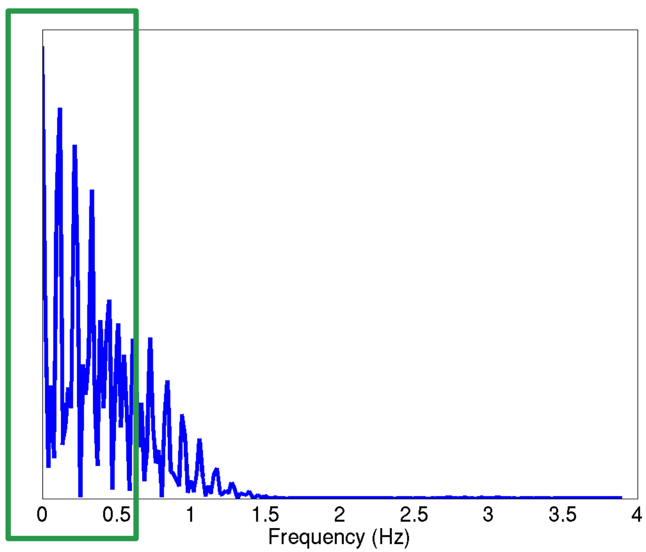


Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

Undersampled in time

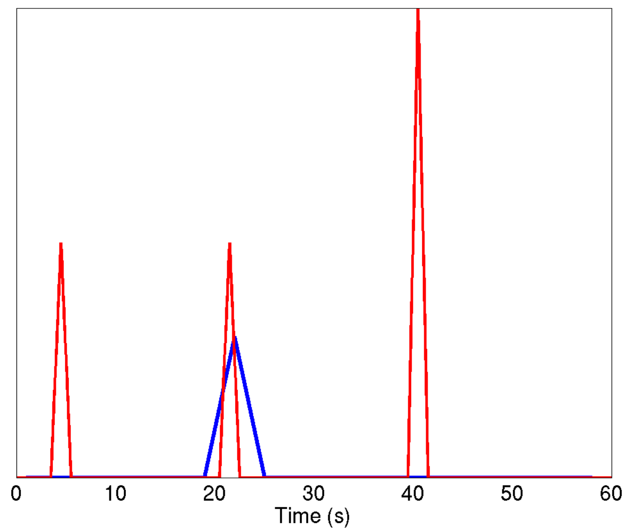




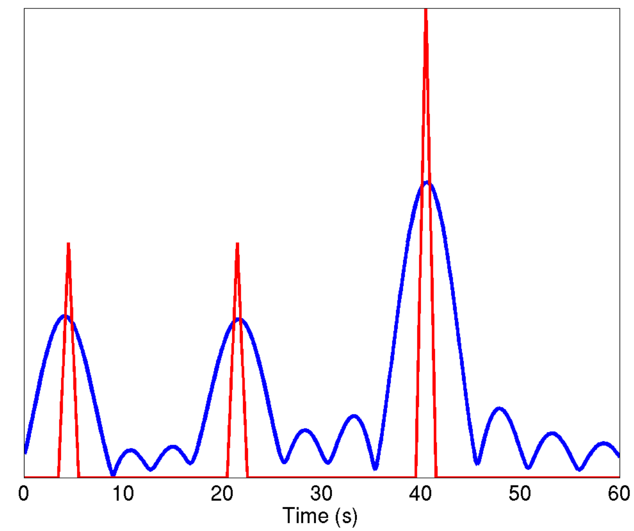
Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

Undersampled in time



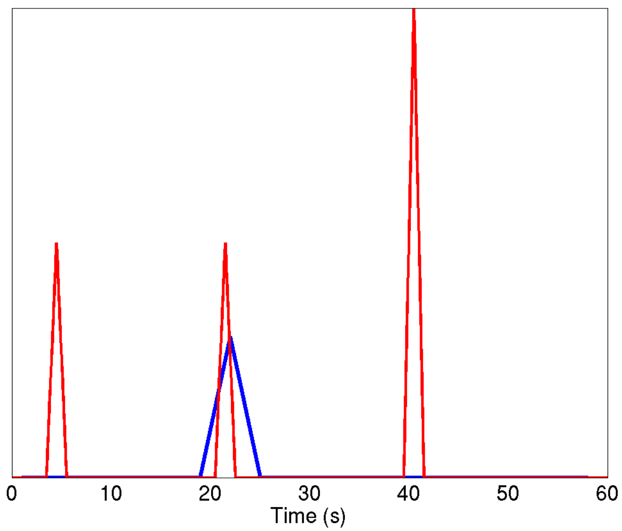
Undersampled in frequency
(reconstructed in time with IFFT)



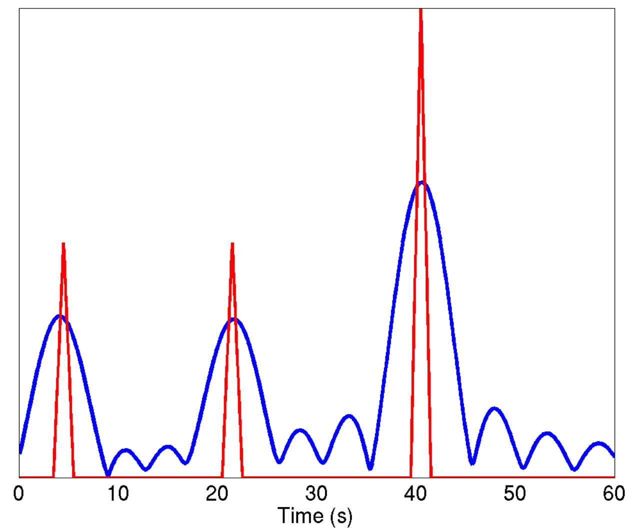
Compressive Sampling

- Sample at lower than the Nyquist rate and still accurately recover the signal, and in some cases exactly recover

Undersampled in time

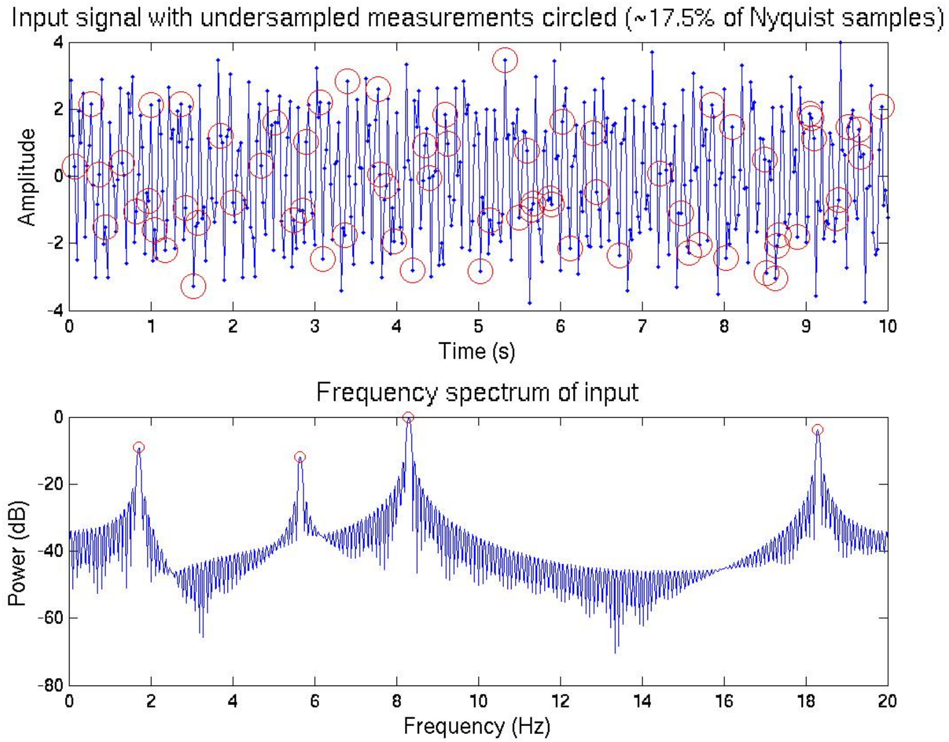


Undersampled in frequency
(reconstructed in time with IFFT)



Requires sparsity and incoherent sampling

Compressive Sampling

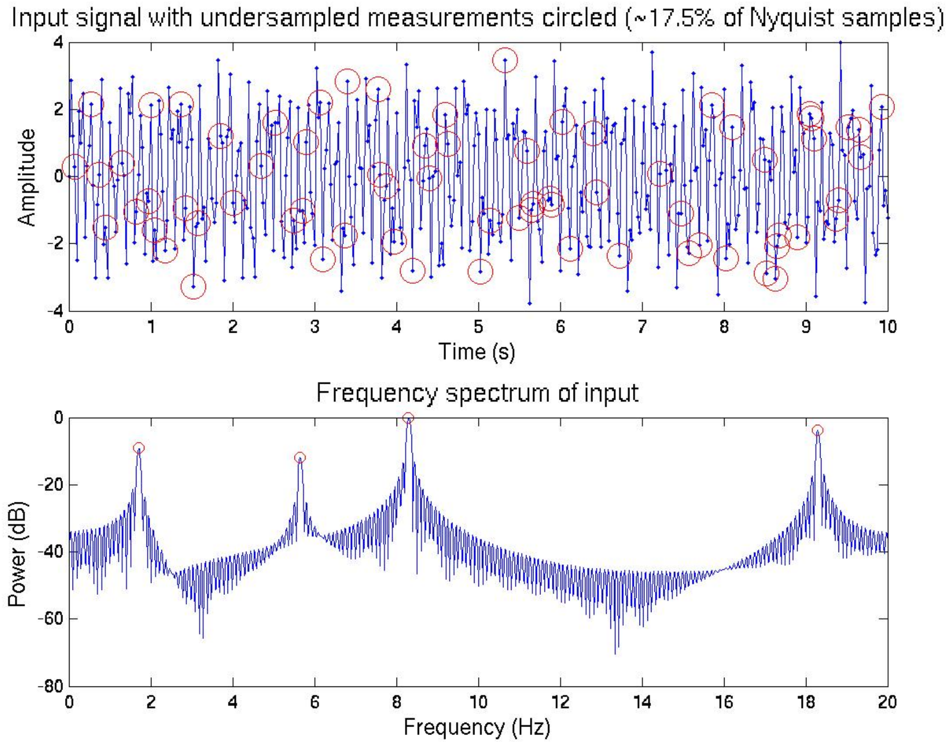


- Sense signal randomly M times
  - M > C· $\mu^2(\Phi, \Psi)$·S·log N
- Recover with linear program

$$\min \sum_{\omega} |\hat{g}(\omega)| \quad \text{subject to} \quad g(t_m) = f(t_m), \quad m = 1, \ldots, M$$

Compressive Sampling

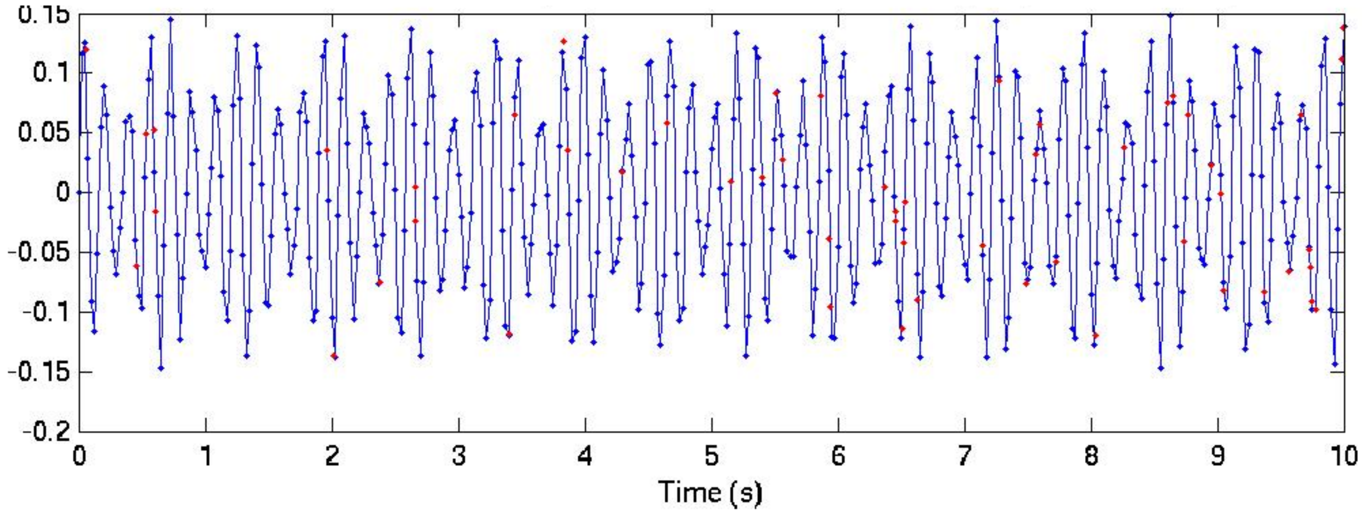
$$\hat{f}(\omega) = \sum_{i=1}^{K} \alpha_i \delta(\omega_i - \omega) \overset{\mathcal{F}}{\Leftrightarrow} f(t) = \sum_{i=1}^{K} \alpha_i e^{i\omega_i t}$$

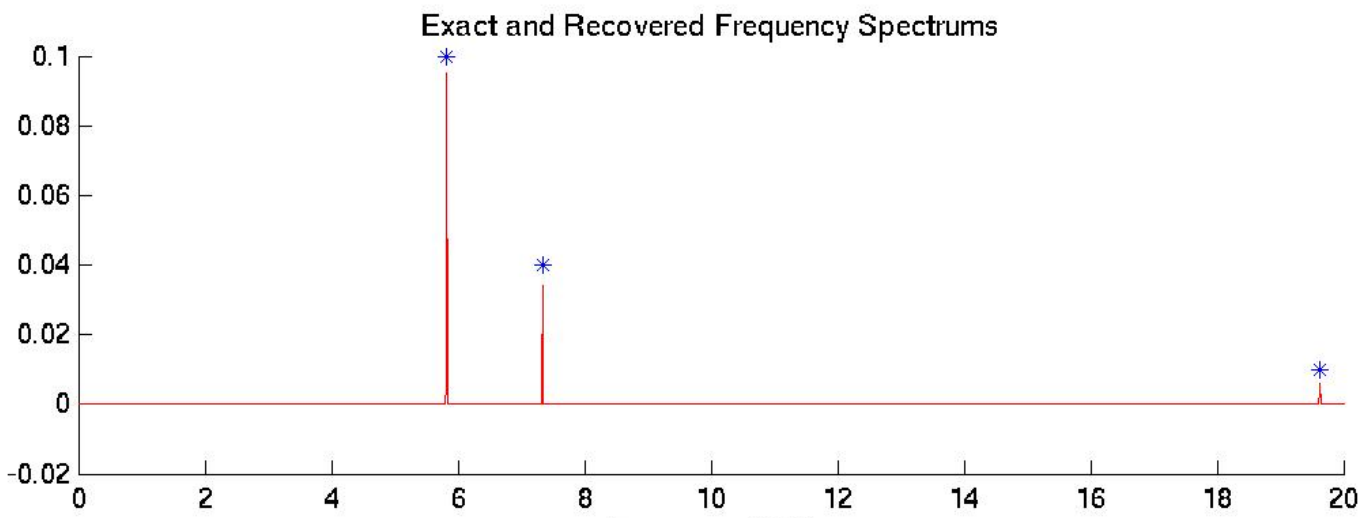


- Sense signal randomly M times
  - $M > C \cdot \mu^2(\Phi, \Psi) \cdot S \cdot \log N$
- Recover with linear program

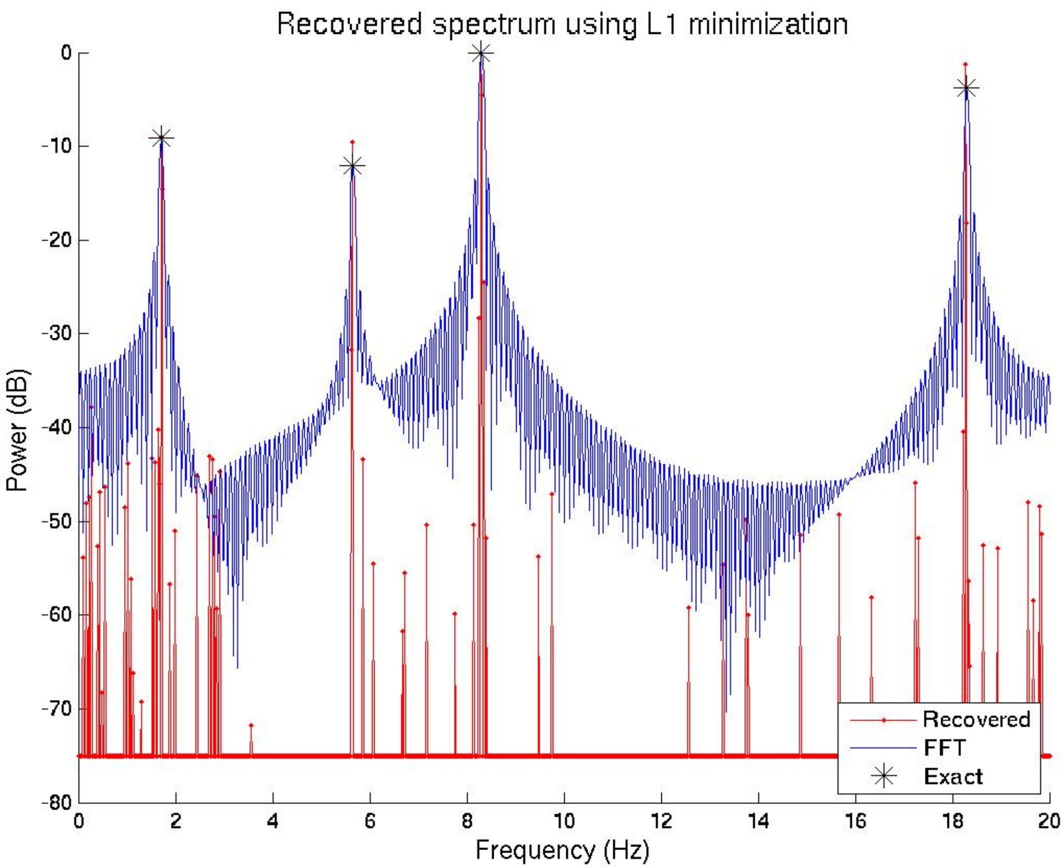
$$\min \sum_{\omega} |\hat{g}(\omega)| \quad \text{subject to} \quad g(t_m) = f(t_m), \quad m = 1, \ldots, M$$

Compressive Sampling: Simple Example



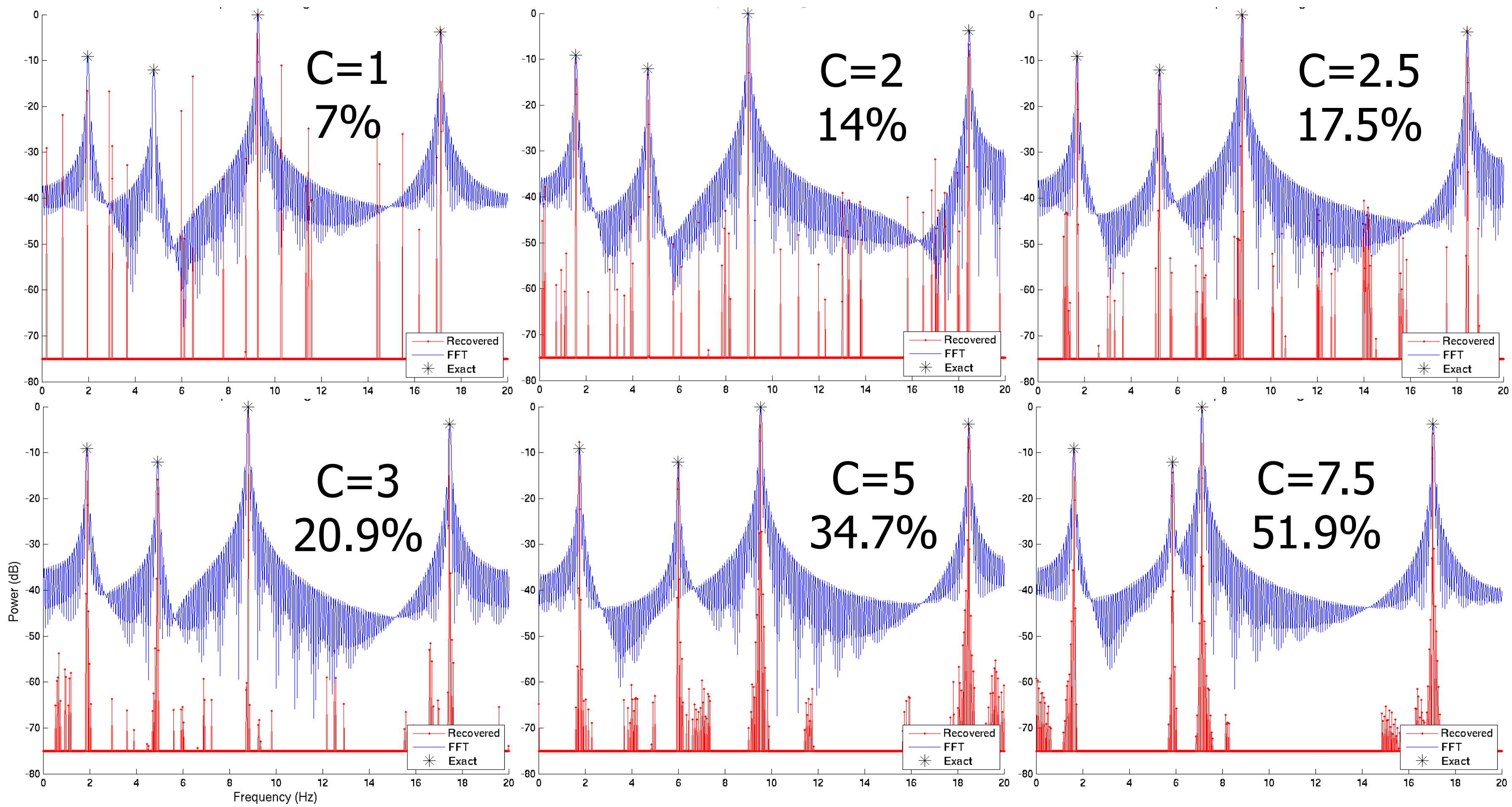


Example: Sum of Sinusoids

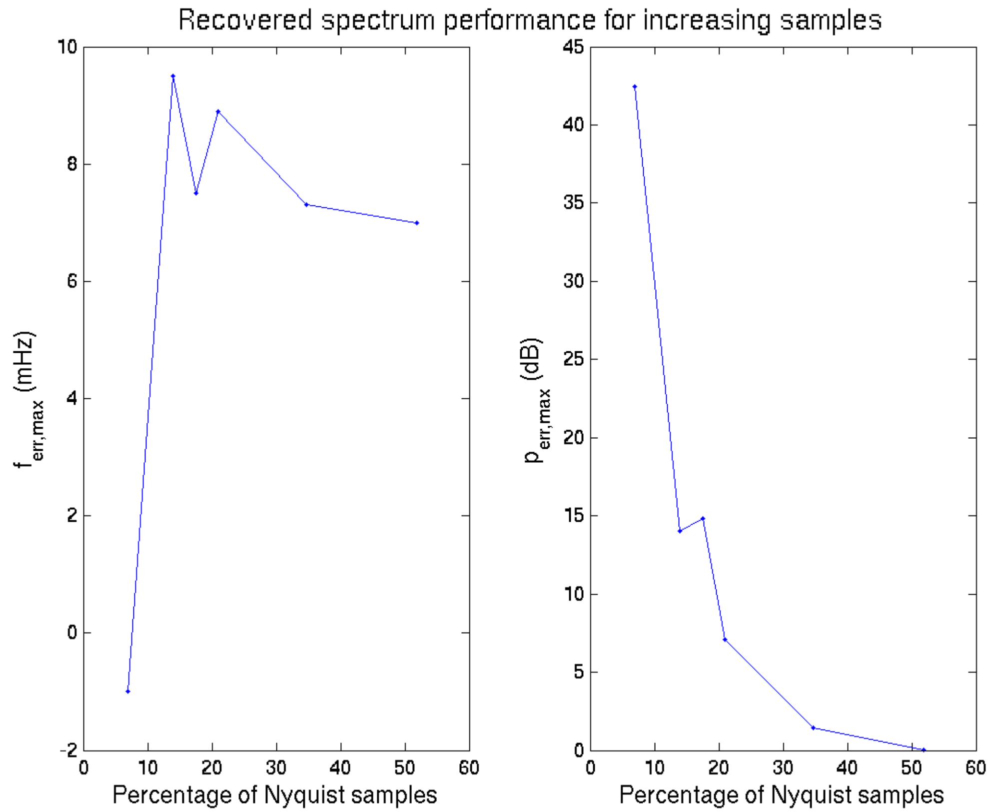


- Two relevant "knobs"
  - percentage of Nyquist samples as altered by adjusting optimization factor, *C*
  - input signal duration, *T*
    - Data block size

Example: Increasing C

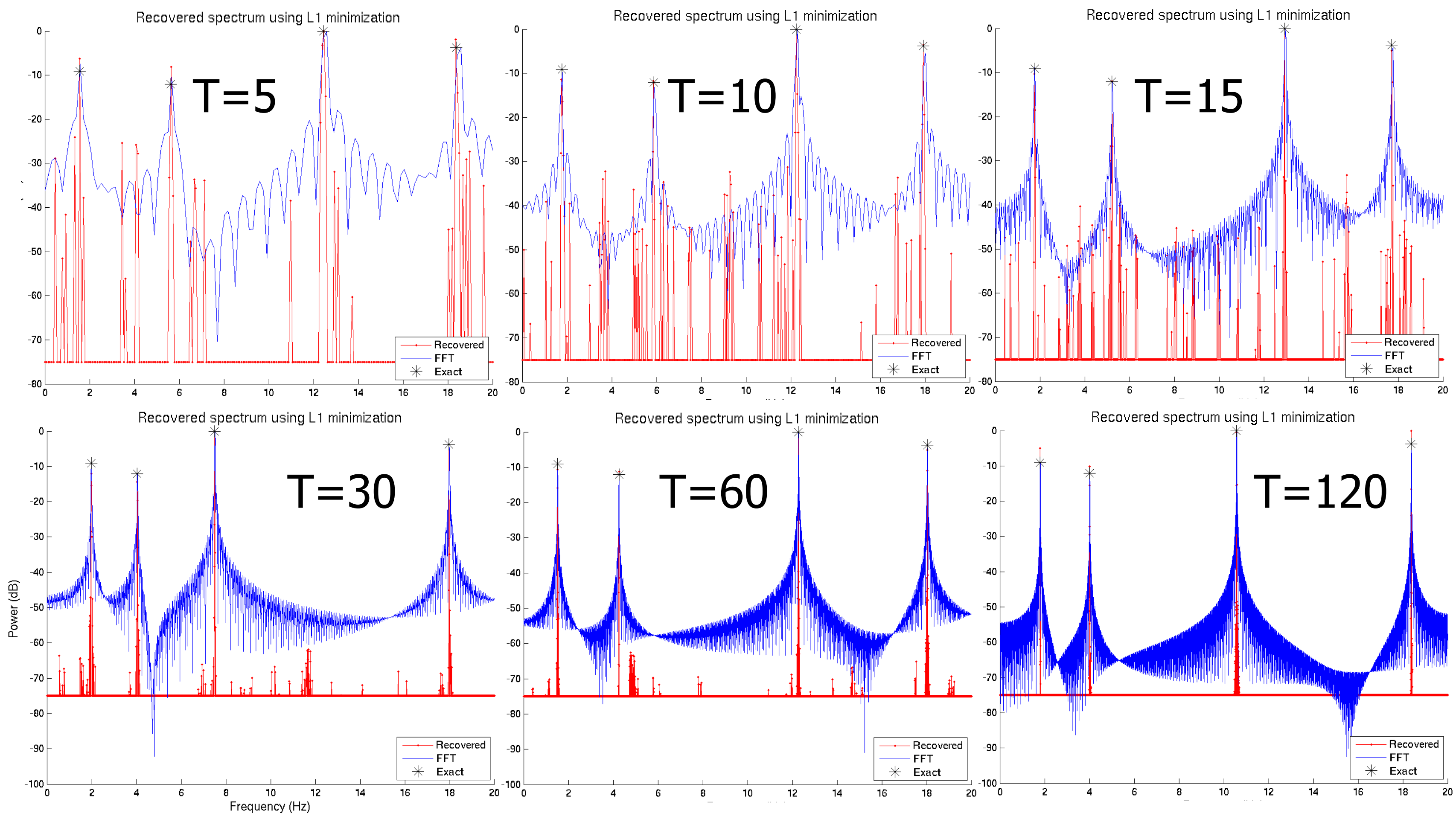


Example: Increasing C

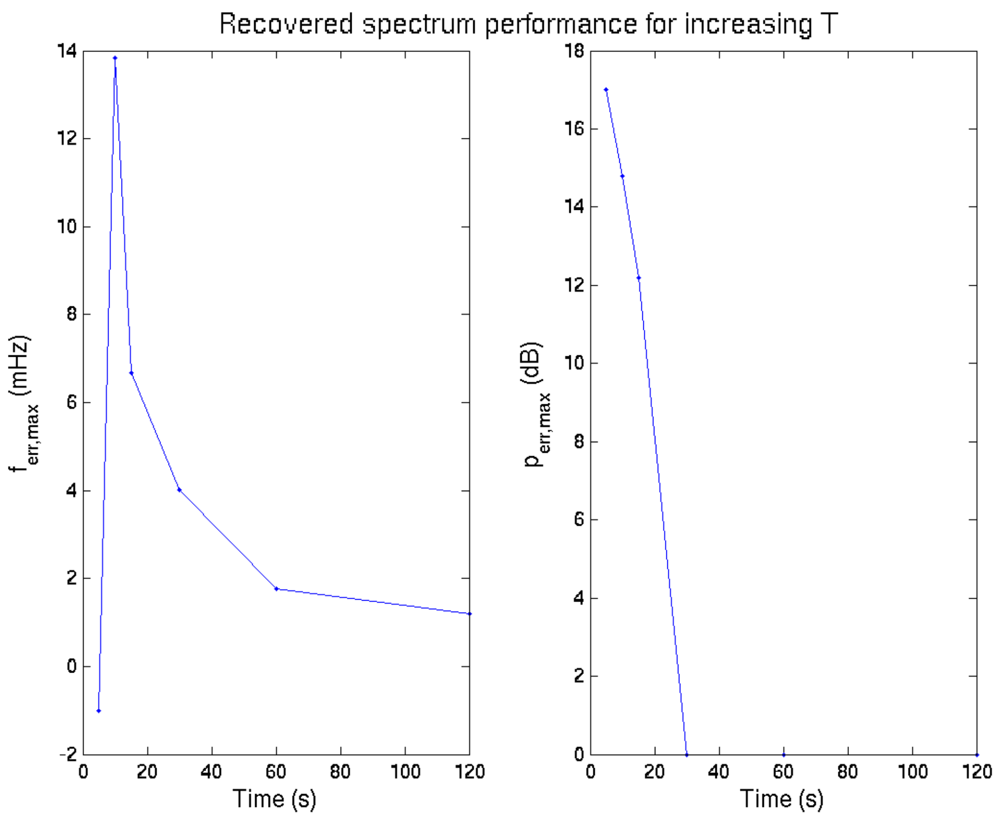


- **$f_{err,max}$ within 10 mHz**
- **$p_{err,max}$ decreasing**

Example: Increasing T

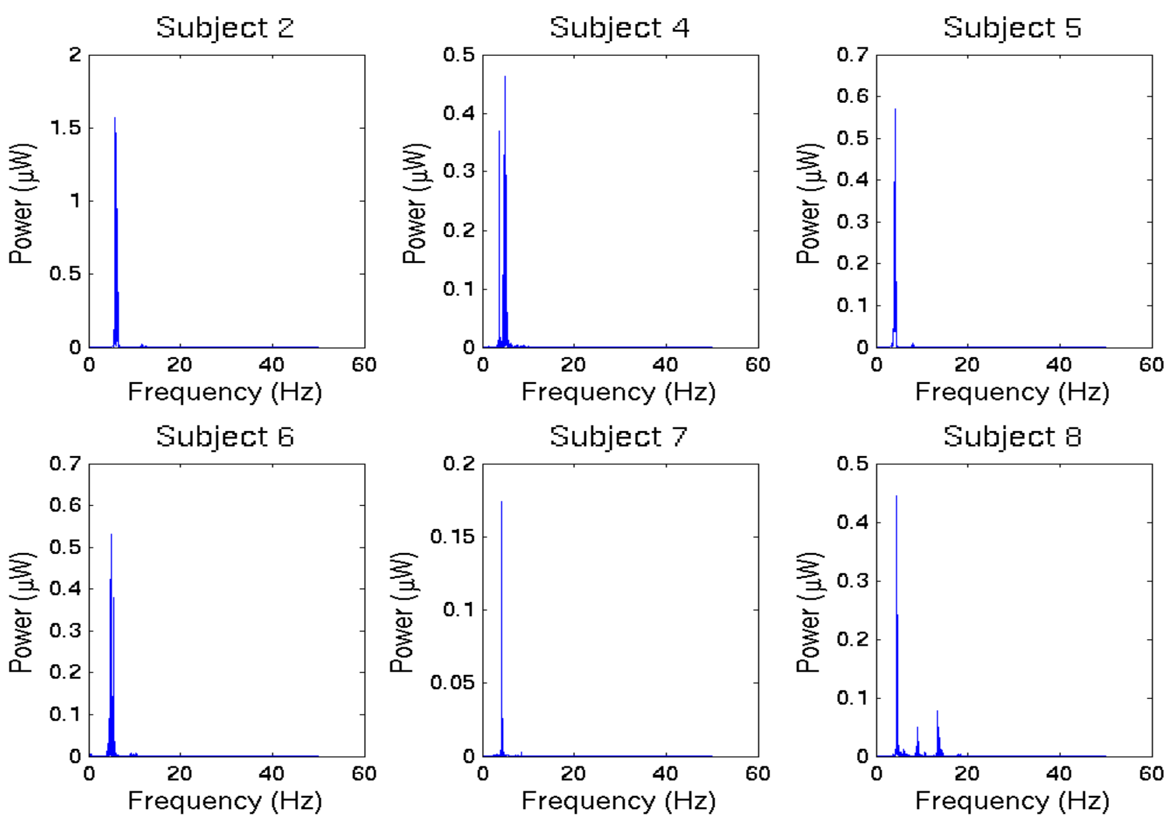


Example: Increasing T



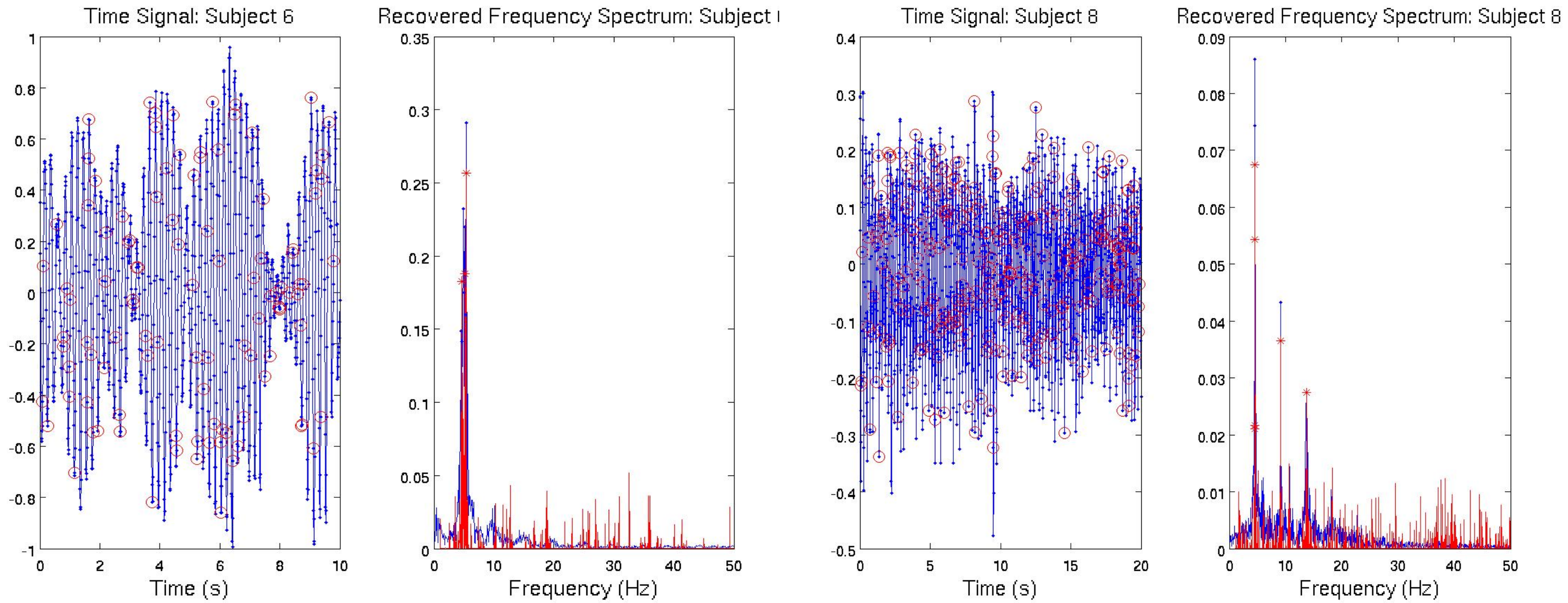
- $f_{err,max}$ **decreasing**
- $p_{err,max}$ **decreasing**

Biometric Example: Parkinson's Tremors

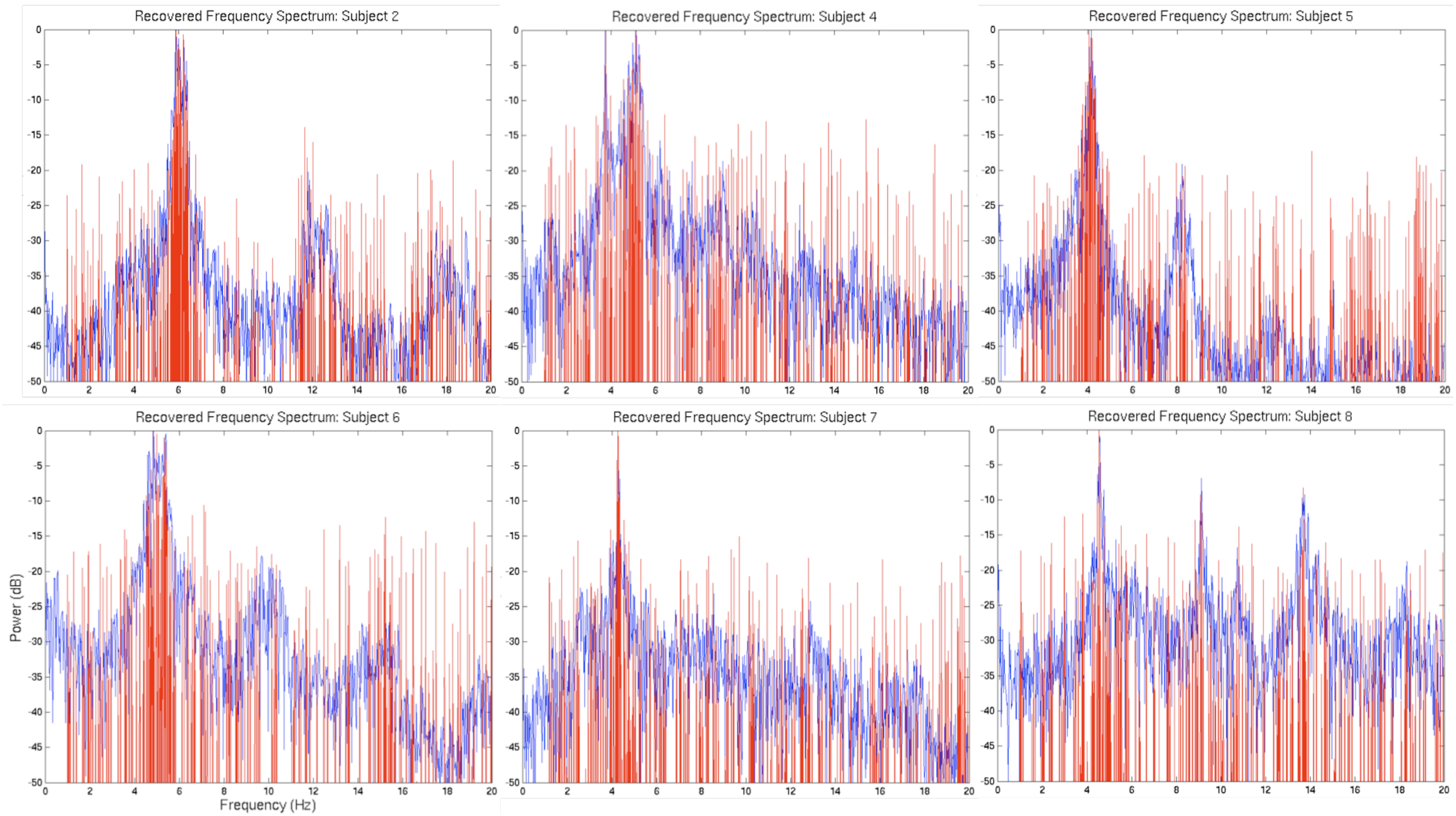


- 6 Subjects of real tremor data
  - collected using low intensity velocity-transducing laser recording aimed at reflective tape attached to the subjects' finger recording the finger velocity
  - All show Parkinson's tremor in the 4-6 Hz range.
  - Subject 8 shows activity at two higher frequencies
  - Subject 4 appears to have two tremors very close to each other in frequency

Compressive Sampling: Real Data

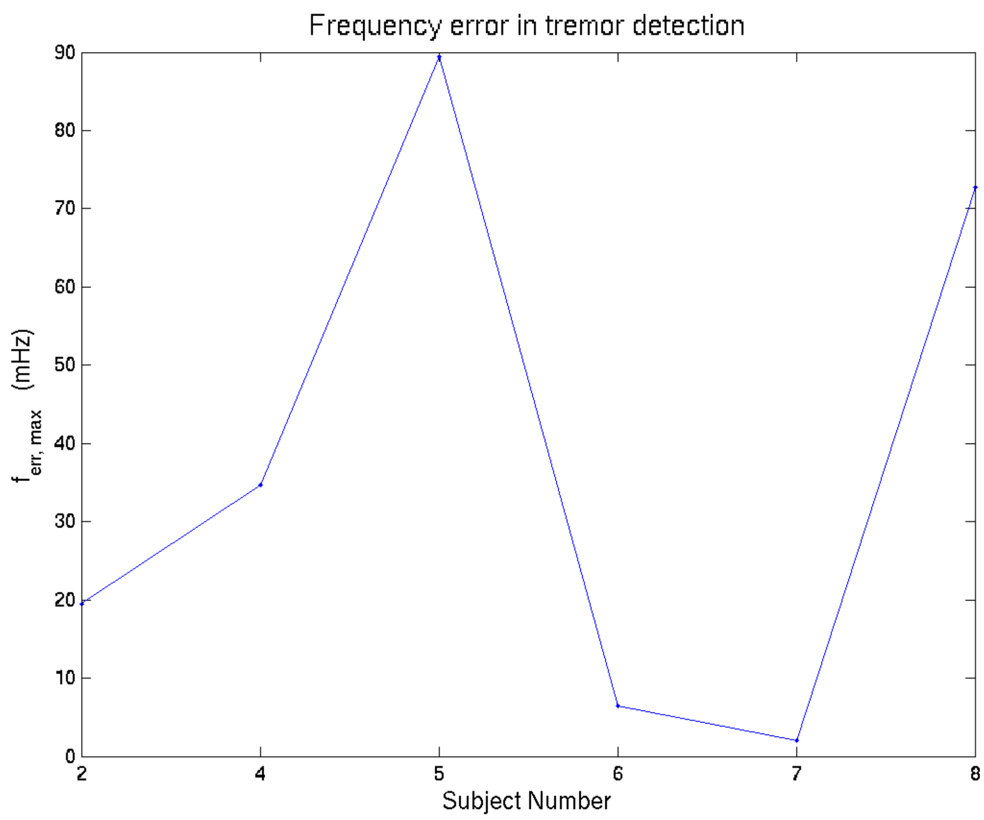


Biometric Example: Parkinson's Tremors



- **C=10.5, T=30**
  - 20% Nyquist required samples

Biometric Example: Parkinson's Tremors



- **Tremors detected within 100 mHz**

Implementing Compressive Sampling

- Devised a way to randomly sample 20% of the Nyquist required samples and still detect the tremor frequencies within 100mHz

Implementing Compressive Sampling

- Devised a way to randomly sample 20% of the Nyquist required samples and still detect the tremor frequencies within 100mHz
  - Requires post processing to randomly sample!

Implementing Compressive Sampling

- Devised a way to randomly sample 20% of the Nyquist required samples and still detect the tremor frequencies within 100mHz
  - Requires post processing to randomly sample!

- Implement hardware on chip to "choose" samples in real time
  - Only write to memory the "chosen" samples
    - Design random-like sequence generator
  - Only convert the "chosen" samples
    - Design low energy ADC

Big Ideas

- Compressive Sampling
  - Integrated sensing/sampling, compression and processing
  - Based on sparsity and incoherency

Admin

- Tania extra office hours
  - M 4-6pm
- Tuesday lecture
  - Review
  - Final exam details
    - Old exam posted
- Project
  - Due 4/25