Digital Signal Processing
The Short-Time Fourier Transform (STFT)

D. Richard Brown III
For signals that have frequency content that is **changing over time**, e.g., music, speech, ..., taking the DFT of the whole signal usually doesn’t provide much insight.

Example: Two second linear chirp

```matlab
N = 16000;
n = 0:N-1;
x = cos(2*pi/80000*(n+1000).^2); % linear chirp
soundsc(x,8000) % listen to sound
```
Example Continued: FFT of Whole Signal

```matlab
plot(2*n/N,20*log10(abs(fft(x)))); % FFT of whole signal
```
plot(2*[0:127]/128,20*log10(abs(fft(x(1:128))))); % FFT near beginning
plot(2*[0:127]/128,20*log10(abs(fft(x(8001:8128))))); % FFT near middle
plot(2*[0:127]/128,20*log10(abs(fft(x(15001:15128))))); % FFT near end
Short-Time Fourier Transform

Rather than analyzing the frequency content of the whole signal, we can analyze the frequency content of smaller snapshots. The STFT is defined as

\[ X[n, \lambda] = \sum_{m=-\infty}^{\infty} x[n + m]w[m]e^{-j\lambda m} \]

where \( n \in \mathbb{Z} \) is a time index and \( \lambda \in \mathbb{R} \) is a normalized frequency index.

Remarks:

1. Implicit in this definition is a window function \( w[n] \) of length \( L \).
2. The STFT results in a family of DTFTs indexed by \( n \).
3. In practice, we don’t usually compute \( X[n, \lambda] \) for all \( n = 0, 1, \ldots \) since the frequency content of the signal does not change much from sample to sample. Instead, we typically pick a “block shift” integer \( R \leq L \) and compute the STFT for values of \( n = 0, R, 2R, \ldots \).
Short-Time Fourier Transform

DSP: The Short-Time Fourier Transform (STFT)

\[ x[n] \]

\[ w[n] \]

\[ x[n]w[n] \]

\[ x[n+8]w[n] \]

\[ x[n+16]w[n] \]

\[ X[0, \lambda) \]

\[ X[8, \lambda) \]

\[ X[16, \lambda) \]

\[ \text{etc.} \]
Short-Time Fourier Transform with the DFT/FFT

We can also use the DFT/FFT to compute the STFT as

\[
X[n, k] = \sum_{m=0}^{L-1} x[n + m]w[m]e^{-j2\pi km/N}.
\]

![Diagram of STFT computation using DFT/FFT](image)
Short-Time Fourier Transform Parameters

1. Window type
   ▶ Tradeoff between side lobe amplitude $A_{SL}$ and main lobe width $\Delta_{ML}$

2. Window length $L$
   ▶ Larger $L$ gives better frequency resolution (smaller $\Delta_{ML}$)
   ▶ Smaller $L$ gives less temporal averaging

3. Temporal block shift samples $R$
   ▶ Usually $R \leq L$ and is related to $L$, e.g., $R = L$ or $R = L/2$
   ▶ Larger $R$ result in less computation
   ▶ Smaller $R$ gives “smoother” results

4. FFT length $N$
   ▶ Usually $N \geq L$
   ▶ Larger $N$ gives more frequency-domain samples of DTFT (better location and amplitude of peaks)
   ▶ Smaller $N$ results in less computation.
Matlab function `spectrogram` is useful for easily computing STFTs.

```matlab
[s,f,t] = spectrogram(x,kaiser(512,2),256,1024,8000); % x = lin chirp
image(t,f,20*log10(abs(s))); set(gca,'ydir','normal'); colorbar;
```
MATLAB Spectrogram Example (decrease $R$ to 1)

```matlab
[s,f,t] = spectrogram(x,kaiser(512,2),511,1024,8000); % x = lin chirp
image(t,f,20*log10(abs(s))); set(gca,'ydir','normal'); colorbar;
```
MATLAB Spectrogram Example (rectangular window)

```
[s,f,t] = spectrogram(x,ones(512,1),511,1024,8000); % x = lin chirp
image(t,f,20*log10(abs(s))); set(gca,'ydir','normal'); colorbar;
```
MATLAB Spectrogram Example (longer window)

```
[s,f,t] = spectrogram(x,kaiser(1024,2),1023,1024,8000);  % x = lin chirp
image(t,f,20*log10(abs(s)));  set(gca,'ydir','normal');  colorbar;
```
Speech Signals Example

ref-mic [quiet]

phys-mic [quiet]

TERC sensor [quiet]

ref-mic [high-noise]

phys-mic [high-noise]

TERC sensor [high-noise]
“Aphex Face” Spectrogram (log scale frequency axis)


From bastwood.com