Review of Fourier Transform

Real part:

\[ R(\omega) = \int_{-\infty}^{\infty} f(t) \cos \omega t \, dt \]

Imaginary part:

\[ X(\omega) = -\int_{-\infty}^{\infty} f(t) \sin \omega t \, dt \]
**Discrete Fourier Transform**

\[
R_k = \sum_{i=0}^{N-1} x_i \cos\left(\frac{2\pi ki}{N}\right)
\]

\[
X_k = -\sum_{i=0}^{N-1} x_i \sin\left(\frac{2\pi ki}{N}\right)
\]

**Computing Spectra in Nyquist**

- Representation: spectra appear as floating point arrays.
- (More detail in Week 5 slides)
What does the array mean?

Frequencies \( \in \{ K/duration \mid 0 \leq K \leq n/2 \} \)

Windows and Step Size

step size (in samples)

frame size (in samples, power of 2)
Perfect Reconstruction?

• Converting to frequency domain and back opens up many possibilities. Can we do this without loss?
  • FFT can be inverted: IFFT
  • Simple but flawed approach:
    • Square windows, no overlap

• What about windowing?

Perfect Reconstruction? (2)

• There are many overlapping windows that sum to one:

  [Diagram of overlapping windows]

• But windows are applied twice!
  • Window \(\rightarrow\) FFT \(\rightarrow\) (alter signal) \(\rightarrow\) IFFT \(\rightarrow\) Window

• Raised \(\cos^2\) with 25% overlap sums to one!
SPECTRAL PROCESSING

Using SAL to operate on spectra

From Sound to Spectra

Data flows this way

Sound → FFT iterator

Spectral frames (arrays) delivered on demand

Sound → SND-IFFT

This is the hidden object that lazily computes sound samples
OOP vs SAL

Data flows this way

CUSTOM OBJECT

FFT iterator

SND-IFFT

Some state

OOP vs SAL

Data flows this way

CUSTOM OBJECT

FFT iterator

SND-IFFT
The Object Behavior We Need

Send \texttt{:next} method, get frame

\begin{center}
CUSTOM OBJECT
\end{center}

Accept \texttt{:next} method, produce frame

How To Get Object Behavior From SAL

Send \texttt{:next} method, get frame

\begin{center}
CUSTOM OBJECT
\end{center}

\texttt{sal-function(\ldots)}

Accept \texttt{:next} method, produce frame
How To Get Object Behavior From SAL (2)

Call `sa-next()` function

CUSTOM OBJECT

Accept :next method, call (SAL) function with some state to produce frame

Template for Spectral Processing (1)

```plaintext
set sa = sa-init(input: "./rpdcello.wav",
fft-dur: 4096 / 44100.0,
skip-period: 512 / 44100.0,
window: :hann)
```
Simple analysis/synthesis examples

- See spectral-process.sal
  - Note: requires spectral-process.lsp and spectral-analysis.lsp as well.
Cross-Synthesis, Morphs, etc.

- In general, combine features from two sounds
- Common approach: separate sounds into excitation and filter parts.
- Combine filter of one signal with excitation of the other, e.g.
  - Vocal tract filter applied to noise, orchestra, etc.
  - Cello body applied to woodwind sound
- See spectral-process.sal