Application: Audio Spectrum Analyser

Digital Signal Processing

Notes

Contents

Notes

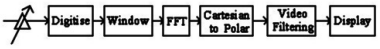
Introduction

A spectrum analyser (SA): "*analyses the spectrum*" of its input signal.

- It produces a plot whose horizontal axis is frequency and vertical axis is amplitude, power or similar.
- It should produce a display in very-near-real-time, so that the device can be used to monitor changes in the spectrum.
- Unfortunately, more fine detail takes more time to make more measurements.
- Both designing and using compromises between resolution and speed.

Notes

Simple Audio Spectrum Analyser

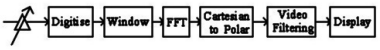


Some important properties:

- Input level control to optimise the function of the ADC
- Digitisation to get a sequence of numbers from an analogue waveform
- FFT to find a spectrum from a waveform
- Cartesian to polar conversion to find the signal power
- Video filtering
- Display

Notes

Simple Audio Spectrum Analyser



- Analyses an arbitrary ongoing waveform.
- Repeatedly apply FFT to blocks of signal that continuously changes.
- Windowing used to reduce problems of performing FFT on (potentially) non-periodic signal.

Notes

Cartesian to Polar

Data coming out of FFT is complex:

$$\mathrm{Re} + j\mathrm{Im}$$

but SA display is a graph of power (or amplitude) against frequency (on either a linear or logarithmic scale).
The simplest to implement is power, by taking the modulus-squared of the complex numbers:

$$|X(f)|^2 = Re^2(f) + Im^2(f)$$

This obviously requires two multiplies (and one add) for each frequency to be displayed, i.e. 2 MACs.

Notes

Video Filtering

- Video filtering performs sliding average of consecutive measurements.
- Averages out noise component of a measurement giving a cleaner display.
- Performs similar function to averaging in TFA design.
- Simple version blends a fraction of each new measurement at a particular frequency into the previous average for that frequency:

$$\begin{aligned}\text{display}(f,n] &= \alpha \times \text{measurement}(f,n] + (1-\alpha) \times \text{display}(f,n-1] \\ &= \text{display}(f,n-1] + \alpha \times (\text{measurement}(f,n] - \text{display}(f,n-1])\end{aligned}$$

- Requires one MAC (and one subtraction) per frequency displayed.
  $(1-\alpha)$ is included so that a stable component in the measurements leads to a stable display.

Notes

SA Computational Issues

- A certain number of MACs are required to perform Spectrum Analysis and
- Working out whether or not the DSP-processor we are using could manage them in the time available is an important aspect of the design.

For example, we could assume the following:

- Processor MAC time = 10 ns
- Resolution bandwidth = 1 Hz
- Hamming window (i.e. not rectangular therefore some MACs required) - windowing is covered later in this handout

The MATLAB program "SA_setup.m" is a demonstration of the kinds of calculations involved and of the trade-offs that might be required.
It assumes an input sampling rate of 44.1 kHz and then seeks the best resolution and display refresh rate for a given processor speed.

Notes

SA Computational Issues

As a first step, it translates the resolution into a bin-size (based on the properties of the window) using the following facts:

$$\text{bin-size} = \frac{\text{sampling-frequency}}{\text{data-segment-length}}$$

and

$$\text{resolution} = (\text{bin-size}) \times (\text{window-6dB-width})$$

from which it concludes that:

$$\text{data-segment-length} = 79821 \text{ samples } (\text{“}L\text{”})$$

The next power of 2 above this data segment length gives the shortest possible FFT:

- FFT length = 131072 samples ("N")
- Zero padding = +64%

Notes

SA Computational Issues

The user specifies the range of frequencies to be displayed, which are then converted to multiples of the spectral line spacing:

$$\text{spectral-line-spacing} = \frac{\text{sampling-frequency}}{\text{FFT-length}}$$

- Display minimum = 0 Hz
- Display maximum = 20000.3082 Hz
- Requesting 59445 points on the f-axis

Note: The program also makes the assumption that there can be no more than 65536 display points along the f-axis (assuming the DAC driving it is 16 bit accuracy).
If the user specifies "*Video averaging on*" then some MACs are going to be needed for this as well. The program deduces that:

$$\text{Total number of MACs per display} = 4714604$$

Notes

SA Computational Issues

This is the *critical calculation* and it is made up of several parts:

Operation	**MACs**
windowing	L (data segment length)
FFT	$2N.log2(N)$ $\{N$=FFT size$\}$
complex-to-power conversion	2 per display frequency
video filtering	1 per display frequency (exponentially-weighted averaging of old & new)

Notice that the window is only applied to the data, not to the whole zero-padded transform.

Notes

SA Computational Issues

Given the total number of MACs per display, the program uses the processor MAC time to deduce that:

- Minimum display update time = 0.047146 seconds

which allows the user to choose a realistic display update time (which must, of course, be more than the minimum): e.g. update time = 0.05 seconds

- Update rate = 20 Hz

Notes

SA Computational Issues

The program then works out how much consecutively processed waveform segments will overlap at this update rate as well as some other useful numbers:

- Overlap between consecutive signal segments = 97%
- FFT length = 131072 samples
- Zero padding = +64%
- Minimum resolution possible at this update rate = 0.60899 Hz

If, for some reason, the display is unsatisfactory, maybe because the flicker on such a slow display refresh rate would be unacceptable, the program allows the user to try again.

Notes

Summary

- Audio Spectrum Analyzer analyses the frequency spectrum of a signal
- Computational requirements for real time operation impose limitations
- Windowing is an important consideration to help improve frequency estimation
- Many types of windows available: all with different combinations of features resulting in a trade off, often between side lobe height and main lobe width.

Notes

Notes