Designing an Audio Spectrum Analyser

Contents

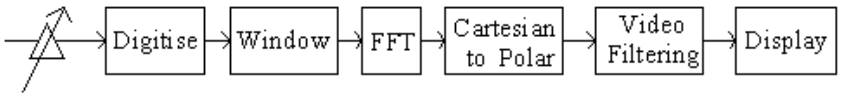
1 Introduction

The function of a spectrum analyser (SA) is to "*analyse the spectrum*" of its input signal (obviously!). This means it produces a display plot whose horizontal axis is frequency and whose vertical axis is amplitude (on a linear, power or dB scale). The idea is that it should produce a display in very-near-real-time, so that the device can be used to monitor changes in the spectrum. Unfortunately, the finer the detail we wish to see (the "resolution"), the longer it takes to make the measurement. This means that both designing a spectrum analyser and using one involves compromises between resolution and speed.

We will be looking at designing a spectrum analyser which is based around the FFT. This is not the only way to do it. Classical RF spectrum analysers have used a superheterodyne approach with a swept local oscillator. However, this is a DSP unit so we won't be discussing the classical method any further.

The basic difference between a spectrum analyser and our previous transfer-function analyser is that we could control the periodicity of the signal in a TFA but we have no such control in a SA. This means we need mechanisms for handling the fact that the signal could contain any frequencies (up to half the sampling frequency, that is) rather than simply multiples of a known fundamental frequency.

2 Structure of a Simple Audio Spectrum Analyser



This raises four obvious questions:

- What are the similarities to the TFA processing?
- What are the differences from the TFA processing?
- What are the reasons for the differences?
- What are the implications of the differences?

There are several similarities between the SA and the TFA processing:

- 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
- display

Also, the "*video filtering*" plays a similar role to the averaging in the TFA, though they are not exactly equivalent.

The crucial difference results from the differences between the signals they are analysing. The TFA is analysing a periodic signal (with some added noise) whereas the SA is analysing an arbitrary ongoing waveform. However, to use the FFT, the signal must be broken up into blocks and that has some unfortunate effects when the signal is not periodic with a period equal to the block length, as a consequence of the implicit periodic extension inherent in the use of the DFT/FFT. This introduces the requirement for windowing to ameliorate the worst of these side-effects. This is an extensive topic and we will spend quite some time on it.

2.1 Cartesian to Polar

The data which comes out of the FFT is complex: $\Re + j * \Im$ but the SA display is supposed to be 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.

2.2 Video Filtering

Video filtering does a kind of sliding average of consecutive measurements. It is used to average out the noise component of a measurement in the case when the signal component is stable with time, giving a cleaner display. It is therefore performing a similar function to that of the averaging in our TFA design.

The simplest version blends a fraction of each new measurement at a particular frequency into the previous average for that frequency:

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

Clearly this form requires one MAC (and one subtraction) per frequency displayed.

Note: The factor of $(1-\alpha)$ is included so that a stable component in the measurements leads to a stable display.

3 SA Computational Issues

The various numerical operations inside a DSP-based Spectrum Analyser will require a certain number of MACs 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 (if it can't be implemented it isn't much of a design, is it?).

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. 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:

$$\text{data-segment-length} = 79821 \text{ samples } (\backslash L")$$

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

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

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:

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

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 (simple exponentially-weighted averaging of old & new values)

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

Exercise:

- Can you work out why windowing is applied like this?

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

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.

Learning Task

Download the MATLAB program "SA_setup.m" and use it to explore the sort of numbers that come out of the program if you choose to display a much smaller part of the frequency spectrum than the full 0-20kHz. Also look into what happens with a different resolution requirement. Post a summary of your observations & thoughts to the discussion board for this topic.