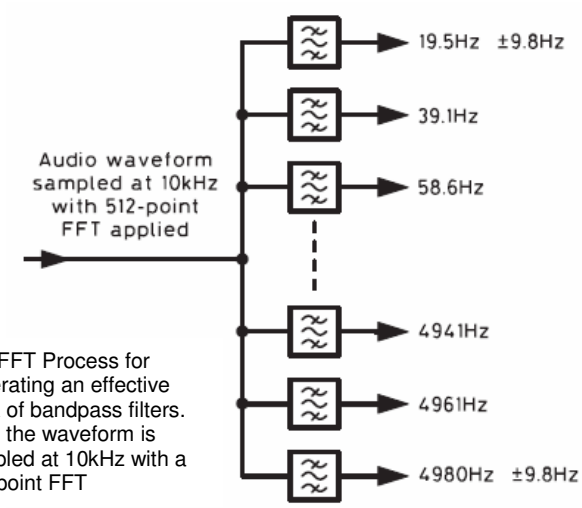
Algorithms and Techniques for Detecting Signals in Noise

Andy Talbot G4JNT

Monitoring of beacons that are normally undetectable but can suddenly come up out of the noise can be a rather time consuming and boring task. It would be a lot easier if we could automatically detect the presence of a weak carrier in noise and either sound an alarm or start some recording software. A simple threshold detector on the AGC line, or an audio filter followed by a tone detector, is one means to detect signals that are strong enough to trigger the circuitry, but for signals that are buried right down in the noise these techniques will not work.

Determine the Frequency Spectrum



The FFT Process for generating an effective bank of bandpass filters. Here the waveform is sampled at 10kHz with a 256 point FFT

A very sensitive carrier detection scheme can be devised, capable of generating a 'signal present' alarm at levels that can barely be heard by ear. The heart of the detection process uses the Fast Fourier Transform, a software that for the purposes of this description is easiest to visualise as a software generated bank of bandpass filters. When a block of successive samples of a digitised audio input waveform are passed though the FFT process, the result is an array of numbers whose values are the amplitudes of each of the spectral components from DC to one half of the sampling rate. The bandwidth of each 'bandpass filter' term is defined by the block length. So, as an example, an audio waveforms could be digitised at 12kHz and 2048 samples passed into the FFT process at a time. The output would be 1024 values of the spectrum from 0 to 6kHz, in units of 12000/2048 Hz, or approximately 6Hz.

So we have filtered the input audio waveform from an SSB bandwidth down to around 6Hz, and more significantly we still have all the information in the original 3kHz available so the need to accurately tune our signal to fit into a single 6Hz bandwidth filter has now disappeared. The rationale behind some of the numbers chosen for this example is fundamental to the FFT process, but we should note that the FFT size has to be a power of two, such as 1024, 2048, 4096 etc.

Effects of Noise

We now have the array of numbers that indicate the signal power present in the 1024 outputs of a bank of bandpass filters, each 6Hz wide, that cover our 0 to nominally 6kHz bandwidth - each one of these 1K values is termed a frequency bin, as in dust-bin! This process is repeated at the block rate, so each block of values is updated every 170ms. We now tune our receiver so the beacon frequency being monitored falls somewhere in the SSB passband. By monitoring the levels of each bin, when the signal appears the value of the appropriate FFT bin corresponding to its tone frequency rises. If this exceeds a preset threshold an alarm can be sounded and whatever action to log the result started. But now we run into problems with noise.

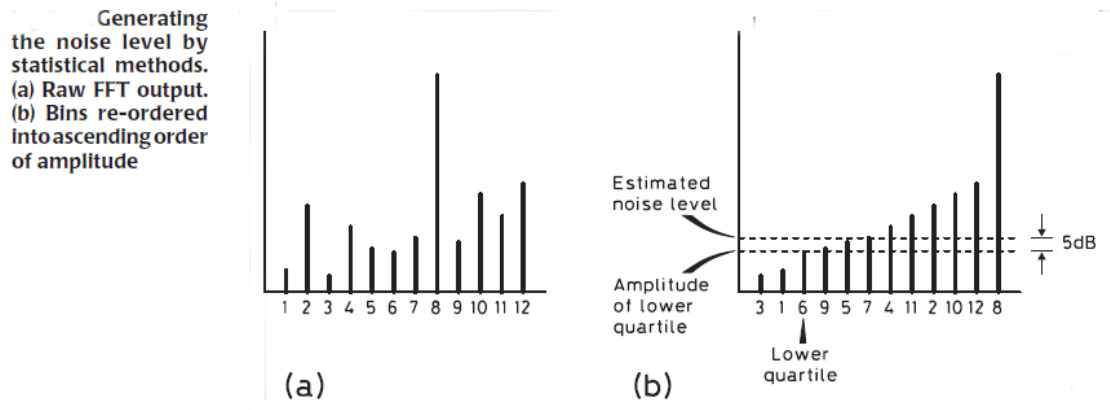
Noise is a random process and its frequency spectrum appears as many spikes at all frequencies, so in the absence of a real carrier each FFT bin will be randomly outputting values, any one of which will be an instantaneous local maximum which will vary randomly from block to block and is very likely to exceed the triggering threshold at some time. This is where statistical analysis starts to play a part. A genuine carrier falling into one of the bins, even if it is only a fraction above the noise, will give an increased value for that bin, but if measured on its own – using one block's worth of data only – it is almost certain to be swamped by noise spikes that can easily be much bigger in many of the other bins, and a simple level detection will not give reliable results. The threshold can be raised to minimise these false alarms, but setting it too high means the signal has to be quite strong before it is detected.

Noise Averaging

The next stage is to average successive blocks of data. By calculating 2048 separate moving averages for each of the bins over the last few blocks of data, we can build up a more accurate estimate of time averaged energy in each of the bins. In practice to save processing overhead and memory, we can miss out the extreme upper and lower edges. If an SSB filter is used in the receiver, only those bins covering 300Hz to 2800Hz, for example, need be considered. The averaged energy in each bin is now compared with the threshold and only if the signal in the same bin is high for several blocks will the alarm be sounded. This two dimensional process, looking over frequency and time, goes some way to smoothing out the effects of noise and begins to offer us a reliable automatic alarm when a signal appears. Visual spectral analysis programmes such as Spectran make use of time averaging in a similar way for detecting weak signal in noise, by making the human eye/brain combination look for the successive increased values in each bin. Greater reliability can be obtained by using a voting process. The signal in a bin has to exceed its threshold say three out of four times, in a successive set of readings before an alarm is generated

But what happens if the noise level changes, such as happens on HF during the day / night transition or if broadband interference appears? If the noise level rises too much while the threshold remains unchanged, there is a reasonable probability that even averaged noise could trigger the alarm – so we need some way of moving the threshold automatically. In other words we want to measure the signal to noise ratio in each bin rather than the absolute level. A way to measure S/N ratio is to sum the entire signal power across all the bins – this total gives the power in the entire input signal of signal plus noise. By making the assumption that a signal, if it appears, will only be in one or two bins at the most, by dividing the total power by the number of bins we get an approximation to the noise level in each bin. Now the time averaged power in each bin can be compared to the effective noise level and the Signal to Noise ratio for each bin calculated. The highest S/N ratio, if it exceeds a threshold, can be considered a valid signal detection. However, this summing of the total also includes any strong interference signals that may be near but not affecting the wanted signal. That would generate a false, artificially high noise level reading.

There is another way of deriving the noise background that does not require the calculation of the total power, is more immune to off signal carrier, spikes and bursts of interference and is often computationally faster that summing all the powers in each bin. The technique relies on some of the statistical properties of white (Gaussian) noise that we do not need to go into here in any detail. When the FFT process has generated its values, these are sorted into ascending power level and the value represented by the lower quartile – the point where a quarter of the bins contain lower values, for a 2048 point FFT, the 512th sorted bin is used. Statistical theory says that by adding a value of 5dB to this the result is the Signal to Noise ratio.



Generating the noise level by statistical methods. (a) Raw FFT output. (b) Bins re-ordered into ascending order of amplitude

Peak Detector

One final aspect that needs to be covered is that of signal bandwidth. We have assumed so far that the wanted signal will fall in just one bin – but this is far from being a valid case. A pure carrier could easily sit at the crossover point of two bins, and each would give a valid answer. Furthermore, if the signal is modulated, even by keying sidebands, the modulation will spread into several adjacent bins. The solution is to detect peaks rather than single bin levels.

One peak detection technique is to step across each frequency bin in turn, comparing it with the previous one. By looking for, say, three successive rises followed by a drop, then a valid peak has occurred on the previous bin. But if the modulation is so wide that it spreads over many bins this technique, which is really for detection of CW type signals, is not really valid.

The Complete signal detection process

We now have a multistage algorithm that can give very reliable detection of low Signal to Noise CW type signals in a varying environment with a low probability of false alarms. The process can be summarised:

1) Perform successive FFTs on blocks of data to generate the frequency spectrum of the input signal plus noise, store these in an array

2) Take the frequency / amplitude original data for the current block and sort it into ascending order of amplitude. Take the value represented by the lower quartile and add 5dB to give the noise level per bin. The original spectral data needs to be preserved, so a separate copy of the data needs to be generated for the sort process.

3) For each block, step across each bin in the original (unsorted) data, looking for three rising values followed by a fall to indicate a peak. If this peak occurs at a Signal to Noise ratio greater than a preset threshold, then note the value of the bin where this peak occurs and store this value in an array containing the history of all peak detections from the last few blocks worth of data.

4) Compare the bin number of each valid peak detection with the peak detections of the last few blocks worth of data using a voting system If more than, say, three out of four peaks exceed the threshold, successively, in the same bin then we have a very good assumption that a signal is present.

This is a complex algorithm to design, but does give one of the most reliable automatic signal detection routines in existence. By fine tuning each of the variables, bin size, number of bins, peak detect rising and falling values, S/N threshold and number of valid votes, it can be made to work for many of the signal types likely to be encountered. The order of some of the steps can be swapped around, eg. storing valid peaks in the history array before comparing to a threshold, or some stages can be left out – the easiest way is just to try it and see!