Improving FFT Frequency Resolution

For EEL 4930: Real-Time Digital Signal Processing

February 5, 2015 J. Marsar

Introduction

When taking the FFT of a signal, the size of the transform is equivalent to the number of frequency bins that will be created. Each bin represents the amount of energy that the signal has at that particular frequency. The frequency resolution is the difference in frequency between each bin, and thus sets a limit on how precise the results can be. The frequency resolution is equal to the sampling frequency divided by FFT size. For example, an FFT of size 256 of a signal sampled at 8000Hz will have a frequency resolution of 31.25Hz. If the signal is a sine wave of 110 Hz, the ideal FFT would show a sharp peak at 110Hz. Unfortunately, with the given frequency resolution, the energy will be split between bins 4 and 5 (93.75Hz and 125Hz, respectively). For some applications, it may be necessary to get a more accurate frequency measurement; this document describes how to do so.

Frequency Resolution vs. Time Resolution:

The most intuitive way to increase the frequency resolution of an FFT is to increase the size while keeping the sampling frequency constant. Doing this will increase the number of frequency bins that are created, decreasing the frequency difference between each. Unfortunately, a greater frequency resolution results in a smaller time resolution. Using the earlier example, the FFT will use 256 samples for each calculation. If 8000 samples are obtained every second, the FFT will be calculated every 256/8000 = 0.032 seconds. Not coincidentally, the inverse of the frequency resolution (31.25Hz) is the time resolution (0.032). An improvement in frequency resolution will result in a diminished time resolution. If the FFT size were increased to 2048 for a great frequency resolution, the transform would be taken on windows of 0.25 seconds. For many real-time DSP applications, an FFT needs to be computed on a much shorter duration. One way around this may be to overlap the transform windows so that samples are used in more than one FFT calculation, but that would become much more complicated while also increasing processing time and data storage. In this situation, it would be ideal to improve frequency resolution without altering the FFT size or sampling frequency.

Spectrum Interpolation:

When looking at a peak in the frequency spectrum of a signal, the relationship between the bin of highest magnitude with those surrounding it can provide valuable information. In the case of the introduction example, it could be deduced that the signal frequency lay somewhere between 93.75Hz and 125Hz because both had a large magnitude. Figure 1 shows a graph of the magnitudes of three consecutive frequency bins as the input sinusoidal signal is swept from the lower bin frequency to the higher bin frequency. In this experiment, the bins corresponded to 2930, 3047, and 3164Hz.

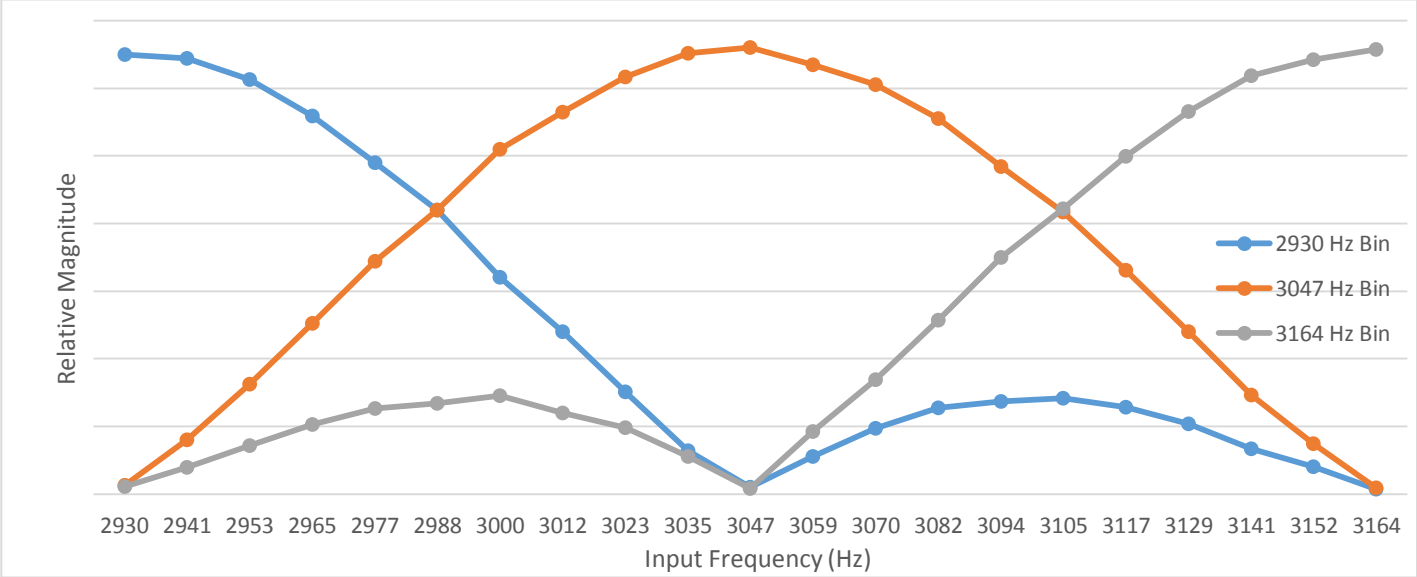

Figure 1. Magnitude of Three Consecutive FFT Bins

Clearly, there is a relationship between consecutive FFT bin magnitudes. Using this information, the exact frequency of the input sine can be approximated, even if it is not equal to one of the bin frequencies. The FFT that is computed in software is a discrete spectrum of bins[1]. For spectrum interpolation, the spectrum will be considered continuous, with the frequency of interest lying somewhere between bins. In the example from the introduction, the input frequency lies between bins 4 and 5, so on a continuous spectrum, a peak would be expected around "bin" 4.5. Interpolation involves approximating this intermittent bin based on three consecutive bins (henceforth described as the "main lobe") that create a peak in the discrete frequency spectrum.

We will denote the maximum bin of interest as k_m, the deviation of the "true" maximum from k_m as Δ_m, and the magnitude of a bin as a function M[k]. Δ_m can be approximated based on the magnitudes $M[k_m - 1]$, $M[k_m]$, and $M[k_m + 1]$. Adding that deviation to k_m will obtain the intermittent frequency bin, which can be multiplied by the frequency resolution of the FFT to determine the input signal frequency.

[1] The DC bin will be considered as bin 0

Two possible methods of frequency estimation will be discussed: parabolic interpolation (PI) and Gaussian interpolation (GI). Spectrum Interpolation involves floating point operations and division, so any hardware or performance limitations must be taken into account. While PI requires less processing, GI produces more accurate results. Either can be used depending on the specific needs of the application.

Parabolic Interpolation:

Though the shape of the main lobe may not be exactly parabolic, the assumption that it is can be used to find a fairly good approximation of the intermittent frequency with PI using the formula below:

$$\Delta_m = \frac{M[k_m + 1] - M[k_m - 1]}{2(2 * M[k_m] - M[k_m - 1] - M[k_m + 1])}$$

Gaussian Interpolation:

GI produces a better approximation of the continuous spectrum, but requires more processing power because it involves logarithmic calculations. The formula is as follows:

$$\Delta_m = \frac{\ln\left(\frac{M[k_m + 1]}{M[k_m - 1]}\right)}{2ln\left(\frac{M[k_m]^2}{M[k_m - 1] * M[k_m + 1]}\right)}$$

The input frequency is then calculated by multiplying the intermittent bin value with the frequency resolution:

$$f = \frac{F_s}{N}(k_m + \Delta_m)$$

Note that this approximation works best on input signals with frequency components that are reasonably spread out, such as a sine wave, square wave, or single instrument note. Any harmonics should be far enough away from the fundamental frequency so that they will not interfere with the main lobe of the spectrum. If the input signal contains additional frequencies between (k_m – 1) and (k_m + 1), it will be more difficult to approximate the true maximum frequency.

Windowing:

Spectrum interpolation works best when the input signal is windowed before the FFT is calculated. Hanning, Blackman, Gaussian, etc. windows will all improve the accuracy of interpolation. A Gaussian window used with Gaussian Interpolation is particularly effective. Use Matlab to generate N windowing coefficients and multiply them with the input samples of the FFT. See the document referenced below for accuracy comparisons between various windowing and interpolation combinations.

Example Code:

The following code utilizes Gaussian interpolation with a Gaussian window to determine the fundamental frequency of a musical note. The sampling window has already been adjusted by Gaussian windowing.

RFFT_SIZE = 256
magAddr: points to array of FFT magnitude results
sample_freq = 8000Hz
TI_FFT(): TI optimized function to calculate FFT
maxidx_SP_RV_2(): TI optimized function to find maximum value in array

```
TI_FFT();                                                   //perform TI FFT analysis.

int max_bin = maxidx_SP_RV_2(magAddr, RFFT_SIZE>>1); //find max bin

float freq_rez = (float)sample_freq/RFFT_SIZE;       //calculate frequency resolution

float inter_bin = max_bin + log(magAddr[max_bin+1]/magAddr[max_bin-1])*0.5
      /log(magAddr[max_bin]*magAddr[max_bin]/(magAddr[max_bin+1]*magAddr[max_bin-1]));
      //calculate the intermittent bin on continuous spectrum using GI

freq = (unsigned)(freq_rez*inter_bin);               //calculate max input frequency
```

Additional information can be found in “Improving FFT Frequency Measurement Resolution by Parabolic and Gaussian Spectrum Interpolation” at the link below:

http://cds.cern.ch/record/738182/files/ab-2004-023