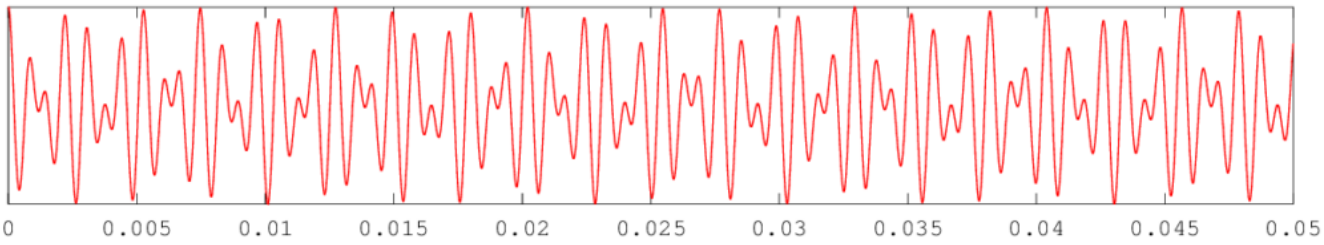
Harmonic Analysis on Embedded Systems

19 May 2016 | **Internet of Things, Internet of Things Articles, Software Engineering** | **Jörg Neulist**
Reading time: 8 minutes

Advanced Signal Processing often involves Harmonic Analysis, the decomposition of a signal into frequency components. On embedded systems, this analysis can be quite challenging. I would like to show you some possible approaches to this problem and highlight their strengths and weaknesses.

A Simple Example of Harmonic Analysis

Touch-tone dialing is a simple real-world example of Harmonic Analysis. With the advent of push-button phones in 1960s, a new dial tone signalling method was introduced. The buttons were arranged in a 4×4 matrix. Each column and each row was associated with a specific frequency and a button triggered a mixture of those two frequencies. The switching center then needs to analyse these mixtures and deduce the phone number from it.



Touch-tone signal for the button “0”, sampled at 1MHz for 50ms.

The above example displays the resulting signal when the button “0” is pressed it represents a mixture of two pure cosine signals at frequencies of 941Hz and 1336Hz. In this blog post, we will try to dissect this signal and retrieve the frequency components.

The Theory of Harmonic Analysis

Before we do that, I would like to give you a short peek into the theory. Let us first examine the space of periodic functions $R->C$ with a fixed period T. Using the convolution operation

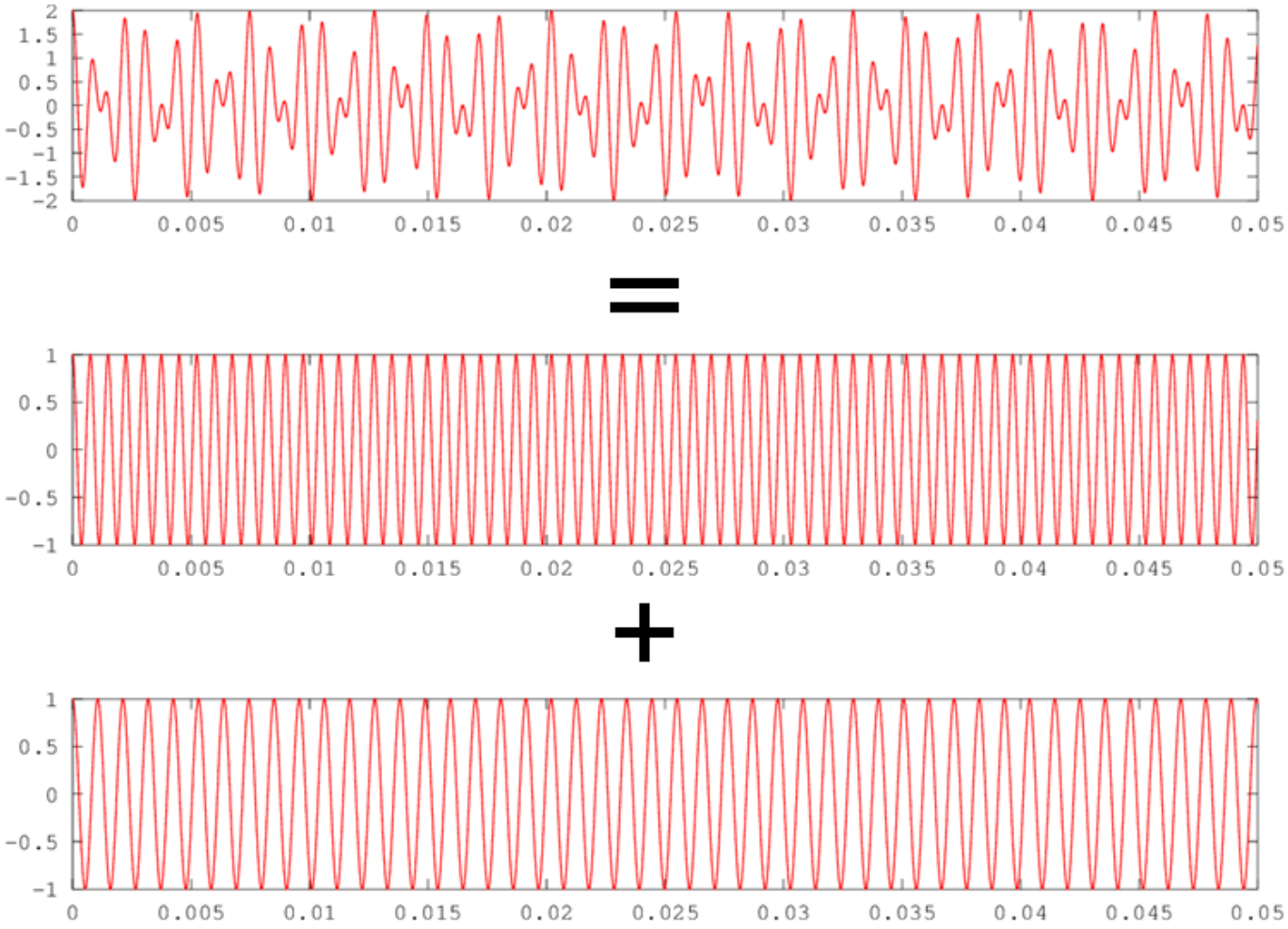
$$f \circ g := \int_0^T f(t)g(T-t)dt$$

this is a vector space. This means that we can uniquely decompose each element into a linear combination of a fixed set of so-called basis elements. One particularly interesting basis for it is the set

$$\left(b_f(t) = e^{i\omega\frac{t}{T}} = \cos\left(\omega\frac{t}{T}\right) + i\sin\left(\omega\frac{t}{T}\right)\right)_{f\in[0;\infty)}, \omega = 2\pi f$$

Fourier transform is the operation of decomposing a given function into the scalar factors for these basis elements. The decomposition can be calculated by convolving the function with each of the basis elements in turn.

Fourier analysis can be adapted for discrete functions as well – the convolution integral is then replaced by a sum. Signals are usually not periodic, but each finite segment of a signal can be treated as periodic with the full segment length.



Fourier decomposition of touch-tone example

This space of finite signals is a vector space as well, and the decomposition is therefore unique. This property is incredibly important! It means that the constituting frequency elements can be recovered from any mixture – much of our modern connected world relies on this property.

How do we go about this decomposition in practice?

DFT (Discrete Fourier Transform)

DFT is an abbreviation for Discrete Fourier Transform and it is the same basic operation noted above for continuous functions: A frequency component is recovered by convolving the signal $\big)_{n=1}$ with the basis element for the given frequency:

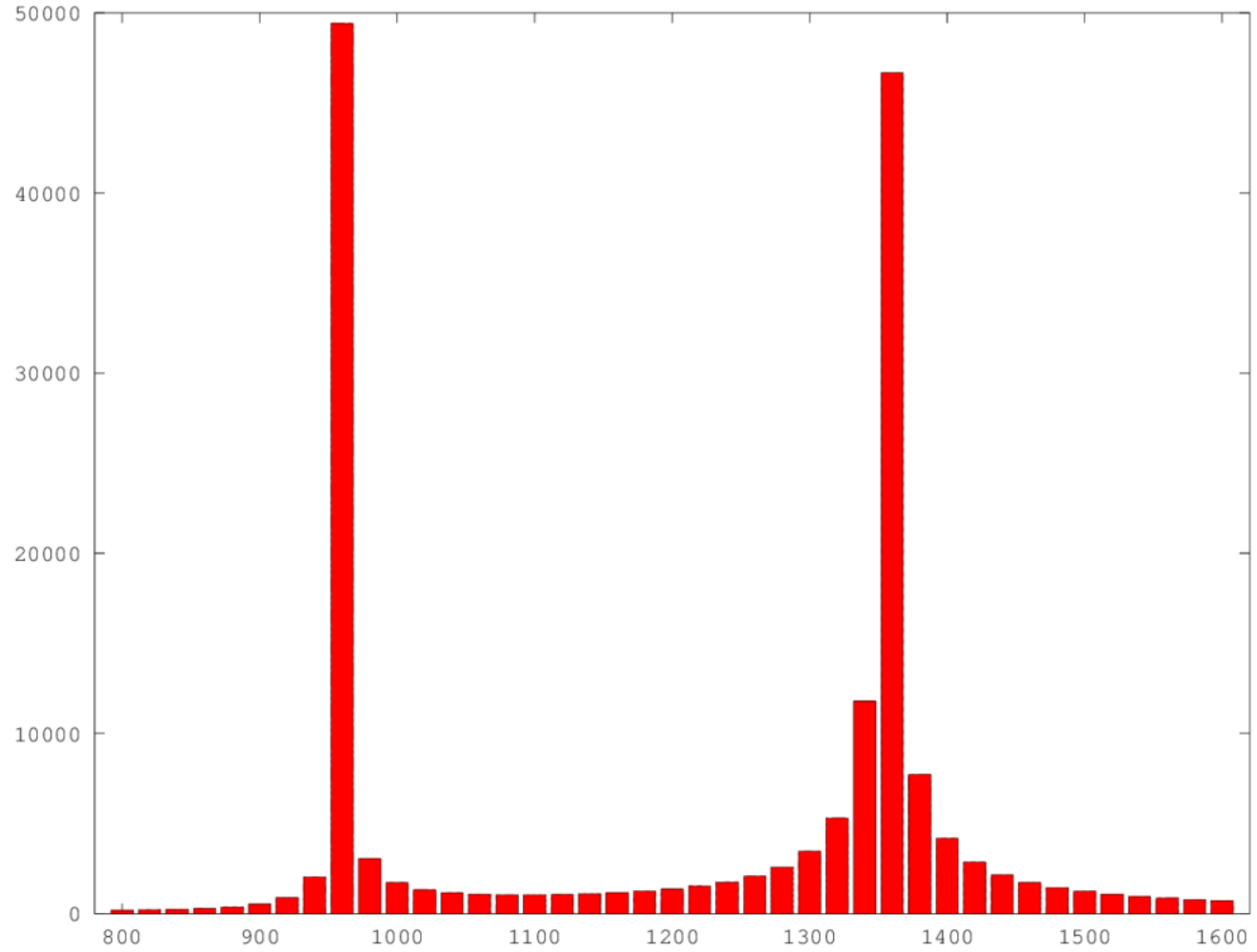
$$c_f = \frac{1}{N}\sum_{n=1}^{N} x_n e^{i\omega\frac{N-n}{N}} = \frac{1}{N}\sum_{n=1}^{N} x_n e^{-i\omega\frac{n}{N}}$$

In our embedded scenario, we want to evaluate this formula more closely: We need to evaluate the basis function in N places. This may be optimised by using look-up tables and some index manipulation magic. Sine and cosine functions can be trivially recovered from a quarter period, so we need to store at least N/4 numbers in our table [image].

We also have N complex multiplications here, and still have calculated only a single linear factor. For a full analysis up to the precision possible, we need N^2 complex multiplications and $N^2/4$ trigonometry calculations or look-ups.

FFT (Fast Fourier Transform)

This straightforward approach to DFT is clearly impractical if we want to do a full decomposition. A whole class of faster algorithms goes by the name of Fast Fourier Transform or FFT.



Excerpt of FFT magnitude from 800Hz to 1.6kHz.

Pictured above is an excerpt of the FFT for our example. For displaying purposes, the magnitude of the complex results has been calculated. The smearing effect of not exactly periodic functions is clearly visible (see the section on interpretation of results below for an explanation). The calculation yields exactly the same results as applying DFT for each frequency in turn.

The classic algorithm is the divide-and-conquer approach described by Cooley and Tukey (Cooley, Tukey, “An algorithm for the machine calculation of complex Fourier series”, Mathematics of Computation 19(1965): 297-301). The idea is to calculate DFTs for the even and odd indices separately and combine them to form the full DFT. If the number of input samples is a power of two, this decomposition can be repeated until transforms for single values are calculated. This calculation is trivial, but each combination of two DFTs takes two complex multiplications. The total number of multiplications is therefore N in the lowest recursion level, N/2 in the level above, etc. The full count of multiplications is thus

$$\sum_{n=1}^{\log_2(N)} 2^n = 2 \sum_{n=0}^{\log_2(N)-1} 2^n = 2(2^{\log_2(N)} - 1) = 2(N-1)$$

The number of trigonometry calculations is unchanged, and some overhead for the recursion is incurred.

Other approaches exist, where some additional operations are introduced in favour of a non-recursive implementation.

Goertzel Filter

FFTs are generally the fastest approaches to calculate a full DFT. However, in many cases the analysis is only needed for a small number of frequencies. A salient example is the interpretation of tone dialing. Here, each key on a telephone's keypad produces a mixture of two out of eight fixed frequencies.

The Goertzel filter was designed for this purpose. It is used to analyse a single frequency component.

The filter works as follows: At first, a series $n=-$ is calculated by seeding with $= s_0$ and running the recursion:

$$s_n = x_n + 2\cos(\omega)s_{n-1} - s_{n-2}$$

After all input data is processed, an additional input element of 0 needs to be applied:

$$s_{N+1} = 2\cos(\omega)s_N - s_{N-1}$$

The final result is obtained by:

$$c_f = s_{N+1} - e^{i\frac{\omega}{N}} s_N$$

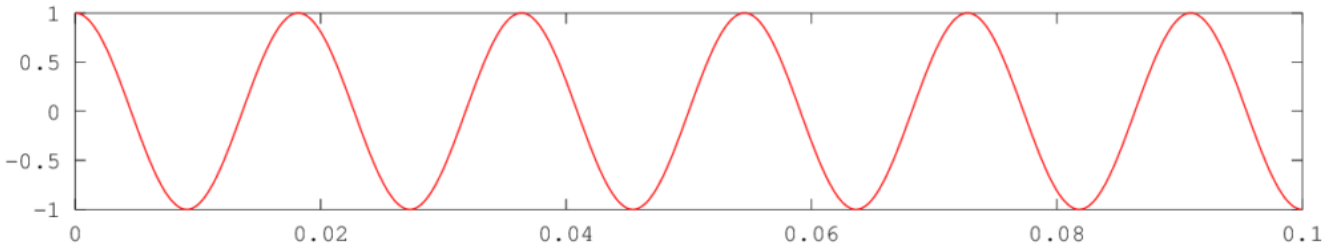
Again, application of this algorithm will yield the same results as the two previous algorithms. Only the effort differs: it takes N+1 real multiplications and only one complex multiplication. It also needs only a single trigonometry calculation.

Interpretation of Results (And Pitfalls!)

None of the algorithms above are easy, and implementation errors may be hard to

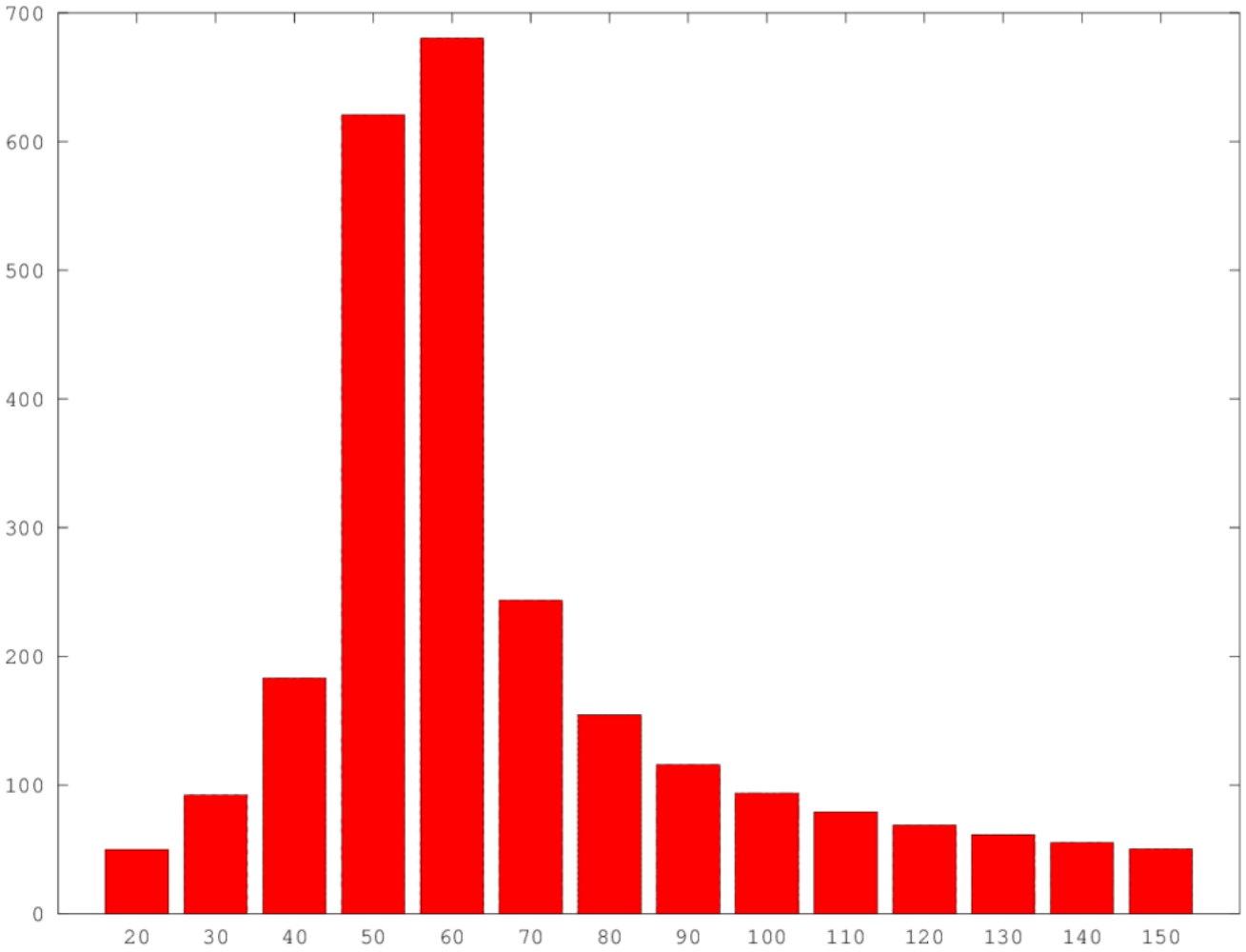
distinguish from their incorrect use or wrong interpretation of the results. Keep in mind:

- Watch the period of your signal!
  DFT can only identify harmonic functions that are periodic with the length of the sample table.
  Example: You have a table of 1024 samples obtained over the course of 100ms, for a sample rate of 10.24kHz. The harmonic analysis can only give you the frequencies of 10Hz, 20Hz, 30Hz, etc. The highest possible frequency is limited by the Shannon Sampling Theorem to half the sample rate, in this case to 5.12kHz.
  Suppose you render a sine with a frequency of 55Hz into this table.



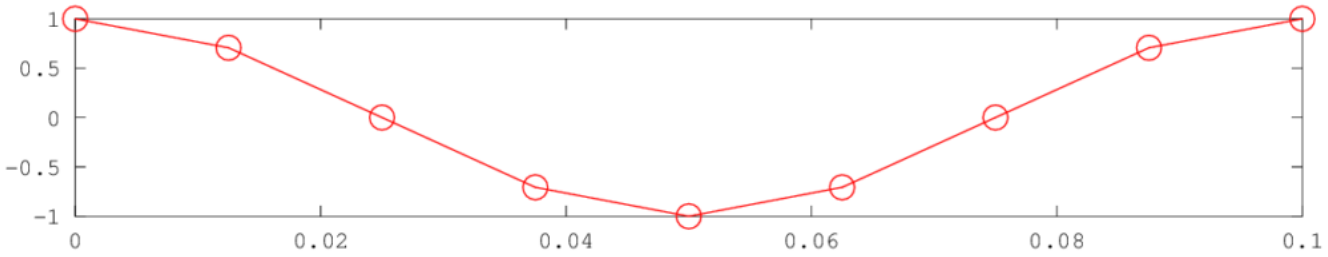
55Hz sampled at 10.24kHz over 100ms

DFT and Goertzel can be used to filter for 55Hz. The result will be unexpected, though.
This function shows five complete sine cycles plus a half cycle. This will be approximated by functions periodic with respect to the table. The result is not a single peak, but a smearing of the amplitude centered around the frequency of 55Hz.
FFT will only yield results for the strictly periodic frequencies, but in essence, the result is the same.

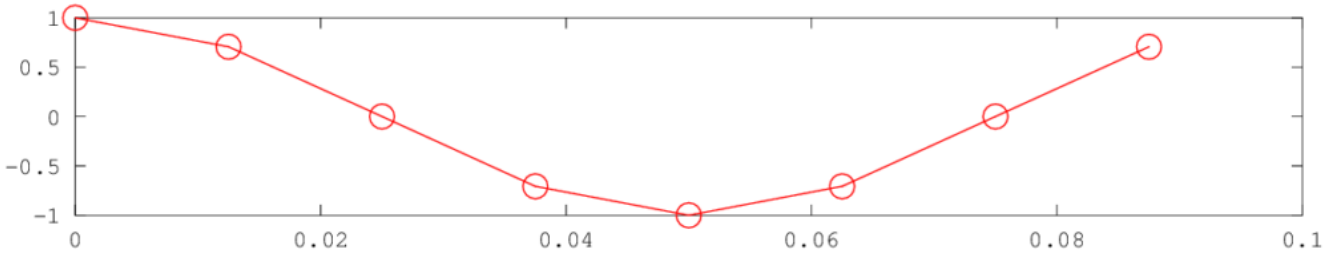


FFT excerpt from 20Hz to 160Hz

- Be precise when generating test signals!
  On a similar note, be very sure that test signals are exactly periodic. If a single generated sine curve with no phase shift has a value of zero in the last sample, it is not periodic. The zero should be the first value beyond the table.
  Any approach to Harmonic Analysis should give you the exact amplitude back in this case.



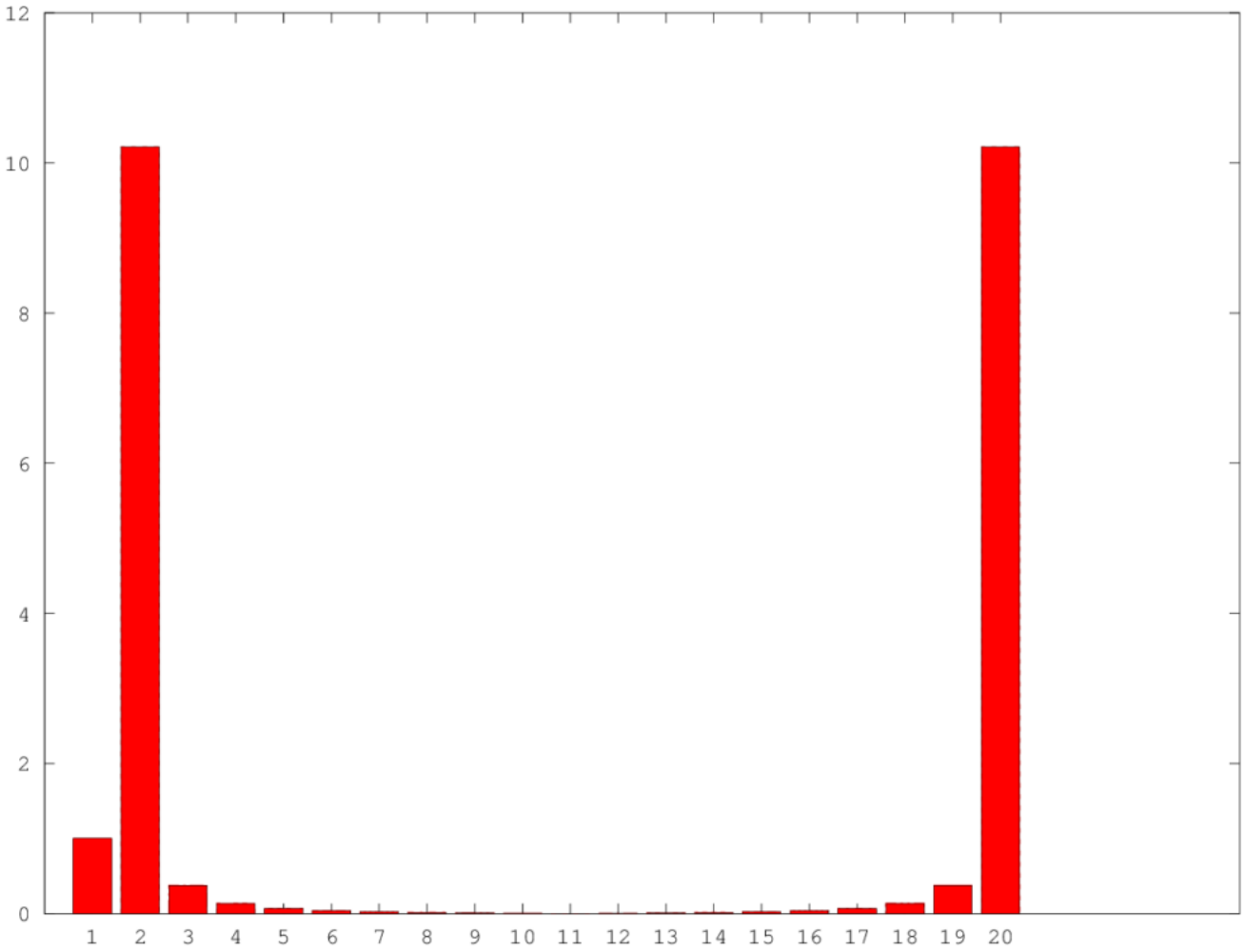
10Hz sampled at 80Hz with 9 samples (not periodic!)



10Hz sampled at 80Hz with 8 samples (periodic!)

In this case, the wrong representation yields and amplitude of 1.111, while the correct representation is correct up to the numeric precision.

- Interpret an FFT plot!
  The outputs of the Fourier Transform are complex numbers. These numbers are usually interpreted in radial representation, yielding amplitude and phase shift of cosine curves. Furthermore, the decomposition distinguishes between positive and negative frequencies. In our use case (real inputs) these are symmetric. The positive frequency elements are simply complex conjugates of the negative frequency elements. The energy is distributed between both, however, so the correct amplitude is twice of what FT calculates! To make matters interesting, this simple rule does not hold for frequency 0 (obviously, since -0=0) and the maximum frequency.



Actual FFT result

A typical FFT representation (as output by GNU octave, or MATLAB). The bar 1 shows the constant offset (i.e. the amplitude for frequency 0). 2 shows 10Hz, whereas 20 shows -10Hz. 3 and 19, 4 and 18, etc. are similarly mirrored. The bar 11, or 100Hz is special again.
The amplitude can be recovered by adding the correct frequency output and the complex conjugate of the mirrored output, and then dividing by the number of samples N. Then, an FT can also be interpreted as a correlation filter, correlating the signal with the given basis element.

Conclusion

With this short treatise I hope to give a reference for your future endeavours in harmonic analysis. I have found myself falling for each of the mentioned pitfalls and hope to have learned my lessons.

What is your own practical experience using xFTs? Share it in the comments!