DFT: Computational Complexity*

Don Johnson

Abstract

A brief explanation of calculation complexity and how the complexity of the discrete Fourier transform is order N squared.

We now have a way of computing the spectrum for an arbitrary signal: The Discrete Fourier Transform (DFT)[1] computes the spectrum at N equally spaced frequencies from a length- N sequence. An issue that never arises in analog "computation," like that performed by a circuit, is how much work it takes to perform the signal processing operation such as filtering. In computation, this consideration translates to the number of basic computational steps required to perform the needed processing. The number of steps, known as the **complexity**, becomes equivalent to how long the computation takes (how long must we wait for an answer). Complexity is not so much tied to specific computers or programming languages but to how many steps are required on any computer. Thus, a procedure's stated complexity says that the time taken will be **proportional** to some function of the amount of data used in the computation and the amount demanded.

For example, consider the formula for the discrete Fourier transform. For each frequency we chose, we must multiply each signal value by a complex number and add together the results. For a real-valued signal, each real-times-complex multiplication requires two real multiplications, meaning we have $2N$ multiplications to perform. To add the results together, we must keep the real and imaginary parts separate. Adding N numbers requires $N-1$ additions. Consequently, each frequency requires $2N+2(N-1)=4N-2$ basic computational steps. As we have N frequencies, the total number of computations is $N(4N-2)$.

In complexity calculations, we only worry about what happens as the data lengths increase, and take the dominant term—here the $4N^2$ term—as reflecting how much work is involved in making the computation. As multiplicative constants don't matter since we are making a "proportional to" evaluation, we find the DFT is an $O\left(N^2\right)$ computational procedure. This notation is read "order N-squared". Thus, if we double the length of the data, we would expect that the computation time to approximately quadruple.

Exercise 1 ***(Solution on p. 2.)***

In making the complexity evaluation for the DFT, we assumed the data to be real. Three questions emerge. First of all, the spectra of such signals have conjugate symmetry, meaning that negative frequency components ($k=\left\{\frac{N}{2}+1,\ldots,N+1\right\}$ in the DFT[2]) can be computed from the corresponding positive frequency components. Does this symmetry change the DFT's complexity? Secondly, suppose the data are complex-valued; what is the DFT's complexity now? Finally, a less important but interesting question is suppose we want K frequency values instead of N; now what is the complexity?

*Version 2.11: Apr 13, 2005 9:22 am -0500

†http://creativecommons.org/licenses/by/1.0

[1] "Discrete Fourier Transform (DFT)", (1) <http://cnx.org/content/m10249/latest/#eqn1>

[2] "Discrete Fourier Transform (DFT)", (1) <http://cnx.org/content/m10249/latest/#eqn1>

Solutions to Exercises in this Module

Solution to Exercise (p. 1)

When the signal is real-valued, we may only need half the spectral values, but the complexity remains unchanged. If the data are complex-valued, which demands retaining all frequency values, the complexity is again the same. When only K frequencies are needed, the complexity is $O(KN)$.