

GOERTZEL'S ALGORITHM*

Douglas L. Jones

This work is produced by OpenStax-CNX and licensed under the Creative Commons Attribution License 1.0[†]

Abstract

Goertzel's algorithm reduces the cost of computing a single DFT frequency sample by almost a factor of two. It is useful for situations requiring only a few DFT frequencies.

Some applications require only a few DFT frequencies. One example is frequency-shift keying (FSK)¹ demodulation, in which typically two frequencies are used to transmit binary data; another example is DTMF², or touch-tone telephone dialing, in which a detection circuit must constantly monitor the line for two simultaneous frequencies indicating that a telephone button is depressed. *Goertzel's algorithm*[2] reduces the number of real-valued multiplications by almost a factor of two relative to direct computation via the DFT equation³. Goertzel's algorithm is thus useful for computing a **few** frequency values; if many or most DFT values are needed, FFT algorithms⁴ that compute all DFT samples in $O(N \log N)$ operations are faster. Goertzel's algorithm can be derived by converting the DFT equation⁵ into an equivalent form as a convolution, which can be efficiently implemented as a digital filter. For increased clarity, in the equations below the complex exponential is denoted as $e^{-i\frac{2\pi k n}{N}} = W_N^{-k}$. Note that because W_N^{-Nk} always equals 1, the DFT equation⁶ can be rewritten as a convolution, or filtering operation:

$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{-nk} \\
 &= \sum_{n=0}^{N-1} x(n) W_N^{-Nk} W_N^{nk} \\
 &= \sum_{n=0}^{N-1} x(n) W_N^{(N-n)(-k)} \\
 &= ((W_N^{-k} x(0) + x(1)) W_N^{-k} + x(2)) W_N^{-k} + \dots + x(N-1) W_N^{-k}
 \end{aligned} \tag{1}$$

Note that this last expression can be written in terms of a recursive difference equation⁷

$$y(n) = W_N^{-k} y(n-1) + x(n)$$

where $y(-1) = 0$. The DFT coefficient equals the output of the difference equation at time $n = N$:

$$X(k) = y(N)$$

*Version 1.5: Sep 12, 2006 1:44 pm +0000

[†]<http://creativecommons.org/licenses/by/1.0>

¹http://en.wikipedia.org/wiki/Frequency-shift_keying

²<http://en.wikipedia.org/wiki/DTMF>

³"Spectrum Analysis Using the Discrete Fourier Transform", (3) <<http://cnx.org/content/m12032/latest/#DFTequation>>

⁴"Overview of Fast Fourier Transform (FFT) Algorithms" <<http://cnx.org/content/m12026/latest/>>

⁵"DFT Definition and Properties" <<http://cnx.org/content/m12019/latest/>>

⁶"DFT Definition and Properties" <<http://cnx.org/content/m12019/latest/>>

⁷"Difference Equation" <<http://cnx.org/content/m10595/latest/>>

Expressing the difference equation as a z-transform⁸ and multiplying both numerator and denominator by $1 - W_N^k z^{-1}$ gives the transfer function

$$\frac{Y(z)}{X(z)} = H(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^k z^{-1}}{1 - ((W_N^k + W_N^{-k}) z^{-1} - z^{-2})} = \frac{1 - W_N^k z^{-1}}{1 - (2\cos(\frac{2\pi k}{N}) z^{-1} - z^{-2})}$$

This system can be realized by the structure in Figure 1

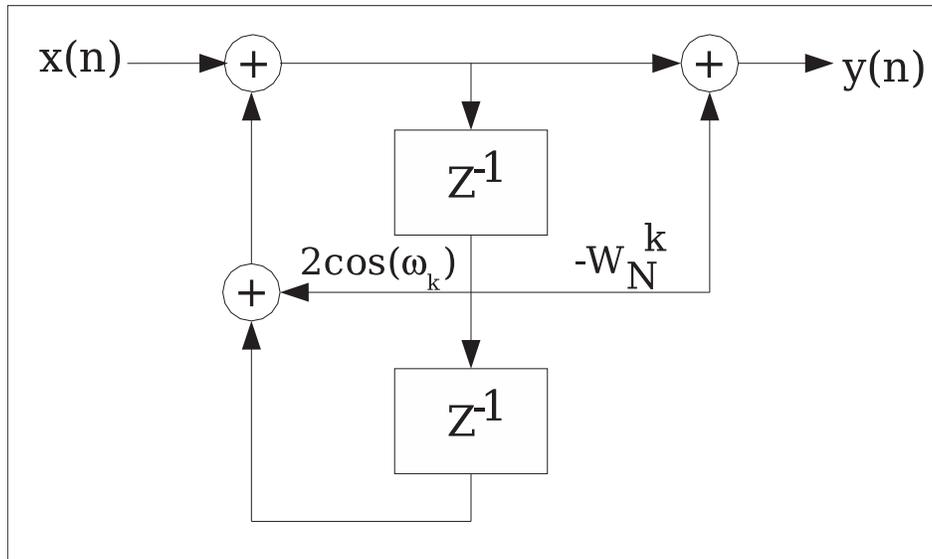


Figure 1

We want $y(n)$ not for all n , but only for $n = N$. We can thus compute only the **recursive** part, or just the left side of the flow graph in Figure 1, for $n = [0, 1, \dots, N]$, which involves only a **real/complex** product rather than a complex/complex product as in a direct DFT⁹, plus one complex multiply to get $y(N) = X(k)$.

NOTE: The input $x(N)$ at time $n = N$ must equal 0! A slightly more efficient alternate implementation¹⁰ that computes the full recursion only through $n = N - 1$ and combines the nonzero operations of the final recursion with the final complex multiply can be found here¹¹, complete with pseudocode (for real-valued data).

If the data are real-valued, only real/real multiplications and real additions are needed until the final multiply.

NOTE: The computational cost of Goertzel's algorithm is thus $2N + 2$ real multiplies and $4N - 2$ real adds, a reduction of almost a factor of two in the number of real multiplies relative to direct computation via the DFT equation. If the data are real-valued, this cost is almost halved again.

For certain frequencies, additional simplifications requiring even fewer multiplications are possible. (For example, for the DC ($k = 0$) frequency, all the multipliers equal 1 and only additions are needed.) A

⁸"Difference Equation" <<http://cnx.org/content/m10595/latest/>>

⁹"Spectrum Analysis Using the Discrete Fourier Transform", (3) <<http://cnx.org/content/m12032/latest/#DFTEquation>>

¹⁰<http://www.mstarlabs.com/dsp/goertzel/goertzel.html>

¹¹<http://www.mstarlabs.com/dsp/goertzel/goertzel.html>

correspondence by *C.G. Boncelet, Jr.*[1] describes some of these additional simplifications. Once again, Goertzel's and Boncelet's algorithms are efficient for a few DFT frequency samples; if more than $\log N$ frequencies are needed, $O(N \log N)$ FFT algorithms¹² that compute all frequencies simultaneously will be more efficient.

References

- [1] Jr. C.G. Boncelet. A rearranged dft algorithm requiring $n^2/6$ multiplications. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-34(6):1658–1659, Dec 1986.
- [2] G Goertzel. An algorithm for the evaluation of finite trigonometric series. *The American Mathematical Monthly*, 1958.

¹²"Overview of Fast Fourier Transform (FFT) Algorithms" <<http://cnx.org/content/m12026/latest/>>