Goertzel Algorithm for a Non-integer Frequency Index

Rick Lyons ● October 7, 2013

If you've read about the Goertzel algorithm, you know it's typically presented as an efficient way to compute an individual *k*th bin result of an *N*-point discrete Fourier transform (DFT). The integer-valued frequency index *k* is in the range of zero to *N*-1 and the standard block diagram for the Goertzel algorithm is shown in Figure 1. For example, if you want to efficiently compute just the 17th DFT bin result (output sample X_{17}) of a 64-point DFT you set integer frequency index k = 17 and N = 64, and implement the Goertzel algorithm.

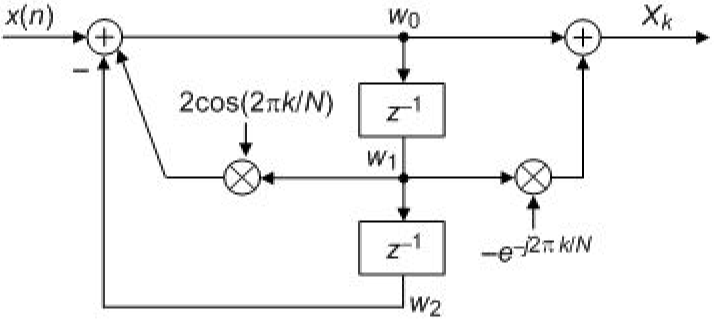


Figure 1

I won't go into any details of how the Goertzel algorithm works because there's plenty of information on the Internet, as well as in most DSP textbooks, describing that algorithm. However, this blog shows how to implement the Goertzel algorithm for any non-integer value of frequency index k in the range $0 \le k < N$.

Using a Non-integer Frequency Index

A straightforward generalized Goertzel algorithm, derived and presented by Pavel Rajmic and Petr Sysel in reference [1], gives us the opportunity to have non-integer values for the frequency index k. This yields the flexible situation where the center frequency of the spectral analysis need not be restricted to be an integer multiple of the DFT's fundamental frequency of f_s/N, where f_s is the $x(n)$ input signal's sample rate in Hz. I use the word "flexible" because, for example, Matlab's goertzel() command requires that frequency index k must be an integer [2].

My interpretation of the Rajmic and Sysel algorithm is shown in Figure 2, where the frequency index k can be any real value in the range $0 \le k < N$.

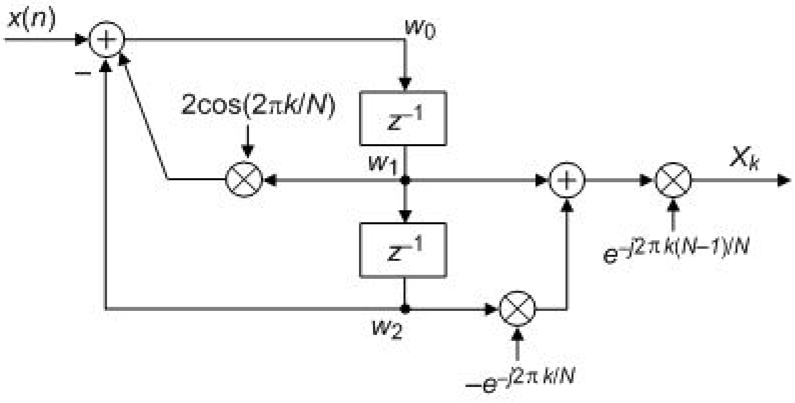
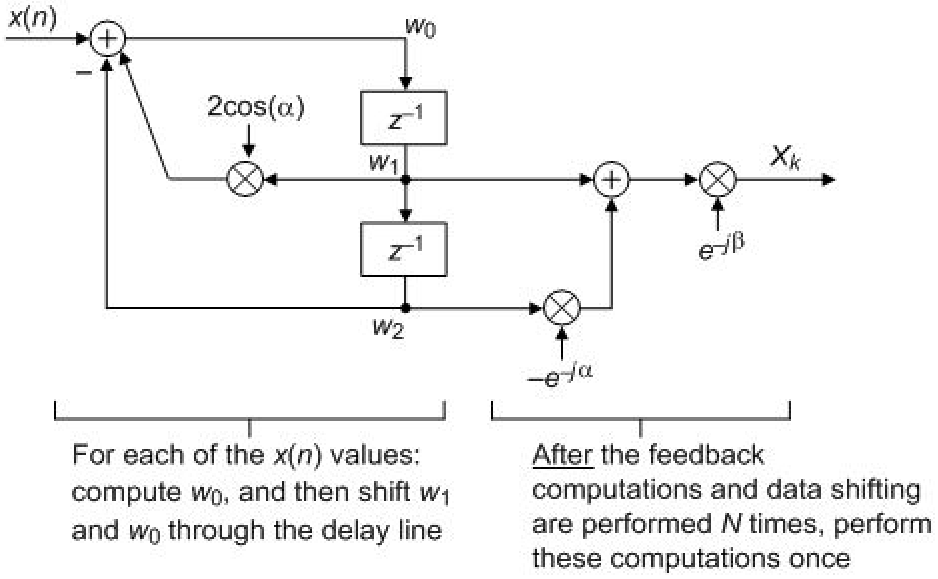


Figure 2

To implement this non-integer *k* Goertzel algorithm, we perform the processing shown in Figure 3.



For each of the *x*(*n*) values: compute *w*0, and then shift *w*1 and *w*0 through the delay line

After the feedback computations and data shifting are performed *N* times, perform these computations once

Figure 3

Angle variables α and β (both measured in radians) are defined as:

$\alpha = 2\pi k/N$, and (1)

$\beta = 2\pi k(N-1)/N$. (2)

Before the processing begins, we initialize the delay line contents by setting *w*1 and *w*2 to zero in Figure 3.

If *k* is an integer the Figure 3 processing computes the *k*th DFT bin result of an *N* point DFT. But it's worth noting that when *k* is a non-integer the processing in Figure 3 computes the discrete time Fourier transform (DTFT) of input *x*(*n*) for the frequency value of 2π*k/N* radians/sample, or equivalently kf_S/N Hz.

Reducing Computational Workload

If you "feel the need for speed" ([3]), there's a way to reduce the computational workload of the final complex-valued operations. Assuming the input sequence *x*(*n*) is real-valued and the $-e^{-j\alpha}$ and $e^{-j\beta}$ coefficients are in rectangular form, implementing the final complex-valued computations requires six real multiply and three real addition operations. I was able to reduce that computational workload by formulating the network shown in Figure 4 where *w*1 and *w*2 are nodes in the Goertzel delay line.

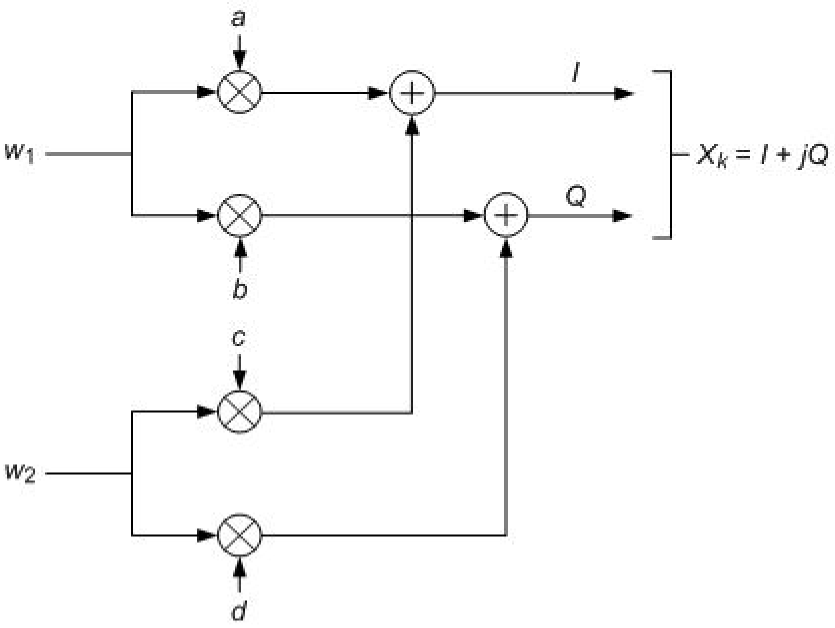


Figure 4

The coefficient values in Figure 4, precomputed and stored in memory prior to processing, are:

a = cos(β),

b = -sin(β),

c = sin(α)*sin(β) -cos(α)*cos(β), and

d = sin(2πk).

Implementing the real-valued computations in Figure 4 requires only four real multiply and two real addition operations.

Matlab code is provided below to demonstrate my interpretation of the 'non-integer *k*' Goertzel algorithm. Pavel Rajmic's Matlab code is available at: http://www.mathworks.com/matlabcentral/fileexchange/35103-generalized-goertzel-algorithm/content/goertzel_general_shortened.m

References

[1] P. Sysel and P. Rajmic, "Goertzel Algorithm Generalized to Non-integer Multiples of Fundamental Frequency", EURASIP Journal on Advances in Signal Processing, 2012, 2012:56.

[2] http://www-rohan.sdsu.edu/doc/matlab/toolbox/signal/goertzel.html

[3] A quote from Maverick (Tom Cruise) in the 1986 movie Top Gun. http://www.youtube.com/watch?v=CUpwLhZh66A

Matlab Code

```
function Xk = goertzel_non_integer_k(x,k)
%   Computes an N-point DFT coefficient for a
%   real-valued input sequence 'x' where the center
%   frequency of the DFT bin is k/N radians/sample.
%   N is the length of the input sequence 'x'.
%   Positive-valued frequency index 'k' need not be
%   an integer but must be in the range of 0 –to- N-1.

%   [Richard Lyons, Oct. 2013]

N = length(x);
Alpha = 2*pi*k/N;
Beta = 2*pi*k*(N-1)/N;

% Precompute network coefficients
Two_cos_Alpha = 2*cos(Alpha);
a = cos(Beta);
b = -sin(Beta);
c = sin(Alpha)*sin(Beta) -cos(Alpha)*cos(Beta);
d = sin(2*pi*k);

% Init. delay line contents
w1 = 0;
w2 = 0;

for n = 1:N % Start the N-sample feedback looping
    w0 = x(n) + Two_cos_Alpha*w1 -w2;
    % Delay line data shifting
      w2 = w1;
      w1 = w0;
end

Xk = w1*a + w2*c + j*(w1*b +w2*d);
```

Previous post by Rick Lyons:
☜ Is It True That *j* is Equal to the Square Root of -1 ?

Next post by Rick Lyons:
☞ Computing Translated Frequencies in Digitizing and Downsampling Analog Bandpass Signals