

RESEARCH

Open Access

Goertzel algorithm generalized to non-integer multiples of fundamental frequency

Petr Sysel and Pavel Rajmic*

Abstract

The article deals with the Goertzel algorithm, used to establish the modulus and phase of harmonic components of a signal. The advantages of the Goertzel approach over the DFT and the FFT in cases of a few harmonics of interest are highlighted, with the article providing deeper and more accurate analysis than can be found in the literature, including the memory complexity. But the main emphasis is placed on the generalization of the Goertzel algorithm, which allows us to use it also for frequencies which are not integer multiples of the fundamental frequency. Such an algorithm is derived at the cost of negligibly increasing the computational and memory complexity.

Keywords: Goertzel algorithm, generalization, spectrum, DFT, DTFT, DTMF

1 Introduction

In the case of discrete-time signals, the discrete Fourier transform (DFT) is widely used for spectral analysis. The frequencies of the harmonics in the DFT always depend on the length of the transform, N , and they are integer multiples of the fundamental frequency $\Delta f = \frac{f_s}{N}$, where f_s represents the sampling frequency. Thus, Δf gives the frequency resolution of the DFT. In the case, when the transform length N is not a multiple of the signal period, the signal is a sum of harmonic components whose frequencies are not integral multiples of the fundamental frequency. Such components are not expressible in the N -point DFT spectrum by a single spectral line—this effect is called “leakage” into the neighboring DFT spectral coefficients, placed at the integer multiples of Δf [1].

Therefore, the transform length N always needs to be chosen with respect to the desired accuracy of the frequency resolution. The computational complexity of the DFT increases quadratically with the number of samples/frequencies, and thus in practice we use almost exclusively the fast Fourier transform algorithm (FFT), whose computational complexity is linearithmic (linear-logarithmic). When the task is to identify the modulus

and/or phase of a single or of just a few of the frequency components, even the FFT is of no advantage, because it always computes all the frequency components, most of which are discarded, as being of no interest. In such situations, methods specialized in computing a subset of output frequencies can be exploited with great benefit. Besides the Goertzel algorithm, which deals with single frequencies separately, it is worth mentioning the so-called pruned-FFT [2,3], which is also connected to the zoom-FFT algorithm [1], and the transform decomposition of Sorensen and Burrus [3], which efficiently combines the ideas of the split-radix FFT and the Goertzel approach. However, the essential disadvantage of rounding frequencies to the nearest integer multiples (and the inaccuracy thus introduced) remains if using any of these methods, including the classical version of the Goertzel algorithm.

In Section 2, we first show the derivation of the common Goertzel algorithm in detail. While this may seem superfluous, it will be necessary to refer back to a number of its particular steps in the later sections. Then the computational and memory complexity of the Goertzel algorithm and the FFT is compared. In Section 3, we provide the announced generalization of the Goertzel algorithm, so that it is possible to use it also for the non-integral multiples of the fundamental frequency. Such a generalization has been mentioned before, e.g.,

* Correspondence: rajmic@feec.vutbr.cz
Department of Telecommunications, Faculty of Electrical Engineering and Communication, Brno University of Technology, Purkyňova 118, 612 00 Brno, Czech Republic

in [4,1], but just for the computation of the modulus, not the phase of a harmonic component.

1.1 Example of utilization of Goertzel algorithm—DTMF

The Goertzel algorithm is typically used for frequency detection in the telephone tone dialing (dual-tone multi-frequency, DTMF), where the meaning of the signaling is determined by two out of a total of eight frequencies being simultaneously present [5]. The frequencies of each of the two groups of four signaling tones were chosen such that the frequencies of their higher harmonics or intermodulation products were sufficiently distant. The frequencies chosen for the DTFM have a big least common multiple. Hence, using a digital receiver with a sampling frequency of 8 kHz, the period of DTMF signal amounts to several tens of thousands of samples. In practice, however, the transform length N must be much smaller, so naturally the effect of spectrum leakage will appear. For example, with $N = 205$, instead of the accurate frequency 770 Hz the modulus at approximately 780.5 Hz ($= 20 \cdot 8000/205$) is computed. This situation is illustrated in Figure 1, where it is evident that the maximum occurs at the non-integer multiple of the fundamental frequency.

The value $N = 205$ is often used in practice [6], because one of the local minima of the sum of squared relative deviations of the signaling frequencies is experienced precisely for this length. In this situation, the deviation is approximately equal to 1.4%, while the transmitter frequency tolerance is 1.8%. Nevertheless, in some applications of the Goertzel algorithm the deviation from the exact frequency can exceed a prescribed tolerance, and thus both the DFT and the Goertzel algorithm would be of little use.

Using the approach presented in this article it is not necessary to round the frequencies at which detection is desired; it is possible to determine the modulus and

phase of a component at an arbitrary (even non-integer) frequency. The number of operations and memory requirements increases only negligibly with this approach.

1.2 Notation

In the following text, we assume a discrete signal x of length N , whose samples can be complex, $\{x[n]\} = \{x[0], x[1], \dots, x[N-1]\}$. Symbol k represents the number (index) of the harmonic component in the DFT, thus $k \in \mathbb{N}$. However, in the later parts of the text, we will work also with $k \in \mathbb{R}$. The unit step signal is denoted by $\{u[n]\}$, whilst $u[n] = 1$ for $n \geq 0$, $u[n] = 0$ for $n < 0$.

2 Standard Goertzel algorithm

2.1 Derivation of standard Goertzel algorithm

The algorithm invented by Goertzel [7] serves to compute the k th DFT component of the signal $\{x[n]\}$ of length N , i.e.,

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n}{N}}, \quad k = 0, \dots, N-1. \quad (1)$$

Multiplying the right side of this equation by $1 = e^{j2\pi k \frac{N}{N}}$ leads to its equivalent

$$X[k] = e^{j2\pi k \frac{N}{N}} \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n}{N}}, \quad (2)$$

which can be rearranged into

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n-N}{N}}. \quad (3)$$

The right side of (3) can be understood as a discrete linear convolution of signals $\{x[n]\}$ and $\{h_k[n]\}$, provided

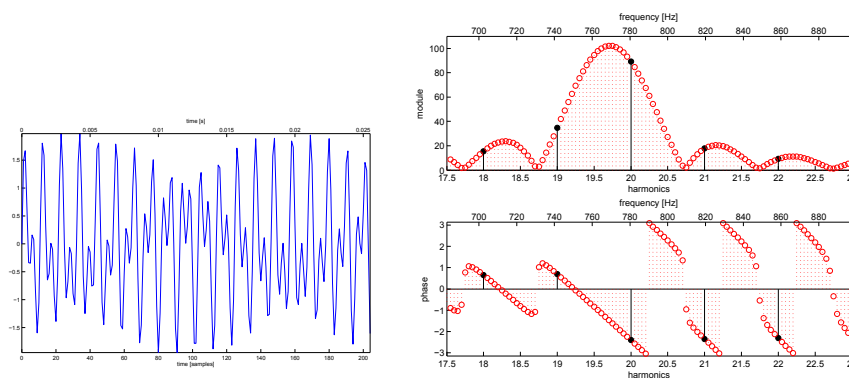


Figure 1 The picture at the top shows the sum of two harmonic signals with frequencies 770 and 1477 Hz. It is 205 samples obtained with $f_s = 8000$ Hz. The bottom picture shows the spectrum of the signal: the DFT coefficients (1) are depicted with black fill, the red ones are the values of DTFT for a non-integer mesh of frequencies (22).

that the elements of the latter signal are defined by $h_k[\ell] = e^{j2\pi k \frac{\ell}{N}} u[\ell]$. In fact, if $\{y_k[n]\}$ denotes the result of such a convolution, then it holds for its entries:

$$y_k[m] = \sum_{n=-\infty}^{\infty} x[n] h_k[m-n], \quad (4)$$

which can be rewritten as

$$y_k[m] = \sum_{n=0}^{N-1} x[n] e^{j2\pi k \frac{m-n}{N}} u[m-n] \quad (5)$$

where the compact support of the signal $\{x[n]\}$ is taken into consideration.

Comparing (3) and (5), it is clear that the desired $X[k]$ is the N th sample of the convolution, i.e.,

$$X[k] = y_k[N] \quad (6)$$

for an arbitrary but fixed $k = 0, \dots, N-1$. This means that the required value can be obtained as the output sample in time N of an IIR linear system with the impulse response $\{h_k[n]\}$.

The transfer function $H_k(z)$ of this system will now be derived; it is the \mathcal{L} -transform of its impulse response [8], thus

$$H_k(z) = \sum_{n=-\infty}^{\infty} h_k[n] z^{-n} \quad (7)$$

$$= \sum_{n=-\infty}^{\infty} e^{j2\pi k \frac{n}{N}} u[n] z^{-n} \quad (8)$$

$$= \sum_{n=0}^{\infty} e^{j2\pi k \frac{n}{N}} z^{-n} \quad (9)$$

$$= \sum_{n=0}^{\infty} \left(e^{j2\pi k \frac{1}{N}} z^{-1} \right)^n, \quad (10)$$

which can be viewed as a geometric series with the first term being equal to $e^{j2\pi k \frac{0}{N}} z^{-0} = 1$ and with the quotient $q = e^{j2\pi k \frac{1}{N}} z^{-1}$. For $|q| < 1$, i.e., $|z| > 1$, the series is convergent and its sum equals the desired transfer function:

$$H_k(z) = \frac{1}{1 - e^{j2\pi k \frac{1}{N}} z^{-1}}. \quad (11)$$

The corresponding difference equation is

$$y_k[n] = x[n] + e^{j2\pi k \frac{1}{N}} y_k[n-1], \quad \text{with } y_k[-1] = 0. \quad (12)$$

This first order difference equation contains a complex multiplication factor, which is computationally demanding. To save the computational cost, the transmission function can be extended in both the numerator and the denominator by the conjugate of $(1 - e^{j2\pi k \frac{1}{N}} z^{-1})$, which leads to

$$H_k(z) = \frac{1 - e^{-j2\pi k \frac{1}{N}} z^{-1}}{\left(1 - e^{j2\pi k \frac{1}{N}} z^{-1}\right) \left(1 - e^{-j2\pi k \frac{1}{N}} z^{-1}\right)} \quad (13)$$

$$= \frac{1 - e^{-j2\pi k \frac{1}{N}} z^{-1}}{1 - 2 \cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}. \quad (14)$$

The respective difference equation of this second order IIR system is

$$y_k[n] = x[n] - x[n-1] e^{-j2\pi k \frac{1}{N}} + 2 \cos\left(\frac{2\pi k}{N}\right) y_k[n-1] - y_k[n-2] \quad (15)$$

with $x[-1] = y[-1] = y[-2] = 0$. Such a structure can be described using the state variables:

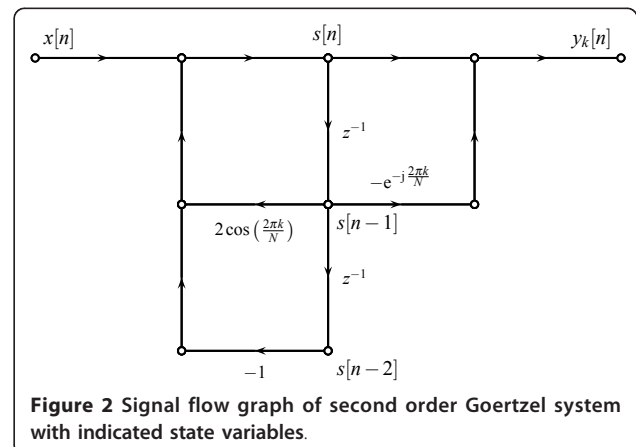
$$s[n] = x[n] + 2 \cos\left(\frac{2\pi k}{N}\right) s[n-1] - s[n-2], \quad (16)$$

while the output is given by

$$y_k[n] = s[n] - e^{-j2\pi k \frac{1}{N}} s[n-1] \quad (17)$$

and we set $s[-1] = s[-2] = 0$. The signal flow graph representing the system is depicted in Figure 2.

The state-space description is advantageous because only the output sample $y[N]$ is of interest. The algorithm iterates the real-number-only system (16) for $(N+1)$ times (beginning with the sample with the time index 0; in the last iteration the input sample $x[N]$ is put equal to



zero). Only in the last step is the output $y_k[N]$ calculated according to (17) using only a single complex multiplication. As mentioned earlier, the value in $y_k[N]$ is the desired spectral coefficient $X[k]$.

The Goertzel algorithm can hence be considered as an IIR filtering process, while only a single output sample is of interest. The algorithm is presented step by step in Figure 3.

2.2 Comparison of Goertzel algorithm and FFT

2.2.1 Properties

The Goertzel algorithm in fact performs the computation of a single DFT coefficient. Compared to the DFT, it has several advantages, because of which it is used.

- First of all, the Goertzel algorithm is advantageous in situations when only values of a few spectral components are required (as in the DTMF example in Section 1.1), not the whole spectrum. In such a case the algorithm can be significantly faster.
- The efficiency of using the FFT algorithm for the computation of DFT components is strongly determined by the signal length N . The most effective case is when N is a power of two. On the contrary, N can be arbitrary in the case of the Goertzel algorithm, and the computational complexity does not vary.
- The computation can be initiated at an arbitrary moment, even at the very time of the arrival of the very first input sample; it is not necessary to wait for the whole data block as in the case of the FFT. Thus, the Goertzel algorithm can be less demanding from the viewpoint of the memory capacity and it can perform at a very low latency. Also, the Goertzel

algorithm does not need any reordering of input or output data in the bit-reverse order [1].

- Finally, as will be shown later in the article, the modulus and phase can be established also for the non-integral spectral indexes k , raising the computational effort only negligibly. Therefore the Goertzel algorithm is convenient in cases when, for some reason, it is required to detect harmonic signals of non-integral frequencies, or, signals with a limited number of samples which causes a decrease of the DFT frequency resolution.

2.2.2 Computational and memory complexities

In the following analysis, operations which can be performed before the first data sample has been received are not considered. Specifically, the constants A , B , C in Figure 3 can be precomputed. The memory performance is handled in a minimalist scenario, i.e., such that it would not be possible to implement the algorithm with fewer storage locations.

The FFT algorithm used with N being a power of two has computational demands proportional to $N \log_2 N$, the absolute number depends on the particular implementation. Usually the number of real-number operations found in the literature is approximately $6N \log_2 N$ (taking one complex multiplication as a combination of four multiplications and two summations). When working with real signals, a number of operations can be avoided; however, it is at the cost of increased complexity of the algorithm, and, it is not true that the demands can be reduced by half, as can be read, for example in [9]. For this reason, we consider the standard “complex” FFT even for real signals.

If we analyze the number of operations of the standard Goertzel algorithm, we realize that for a real input

```

Inputs: index  $k \in \mathbb{Z}$  of the DFT spectral component; signal  $x$  of length  $N$ 
Output:  $y$ , representing  $X[k]$  according to (6)

%Precalculation of constants
 $A = 2\pi \frac{k}{N}$ 
 $B = 2 \cos A$ 
 $C = e^{-jA}$ 
%State variables
 $s_0 = 0$ 
 $s_1 = 0$ 
 $s_2 = 0$ 
%Main loop
for  $i = 0 : N - 1$     % $N$  multiplications,  $2N$  additions
     $s_0 = x[i] + B \cdot s_1 - s_2$     %corresponds to (16)
     $s_2 = s_1$ 
     $s_1 = s_0$ 
end
%Finalizing calculations
 $s_0 = B \cdot s_1 - s_2$     %corresponds to (16) with zero input; 1 multiplication and 1 addition
 $y = s_0 - s_1 \cdot C$     %corresponds to (17); 4 multiplications and 3 additions
    
```

Figure 3 Standard Goertzel algorithm.

signal, N real multiplications and $2N$ real additions are performed in the main loop. So the total number of operations is approximately $3N$ for a single frequency; we omit the small number of operations needed for pre-computing $B = 2 \cos\left(\frac{2\pi k}{N}\right)$, $C = e^{-j\frac{2\pi k}{N}}$ and the concluding complex multiplication (one for each frequency k). Thus, if N frequencies were of interest, the Goertzel algorithm would be of quadratic complexity as the DFT is.

To answer the question “for how many frequencies K is it more advantageous to exploit the Goertzel algorithm than the FFT” we compare

$$\begin{aligned} 3NK &< 6N\log_2 N \\ K &< 2\log_2 N, \end{aligned} \quad (18)$$

which represents a more accurate result than for example [[8], p. 635], where the sharper inequality $K < \log_2 N$, based solely on a comparison of the order of magnitude, is presented. Such a result, however, holds only for N being a power of two; otherwise the inequality (18) can even be more favorable for the Goertzel algorithm.

The formula (18) says that the computation should be faster than the FFT as long as the number of frequencies does not exceed $2 \log_2 N$. For example, with a signal of length $N = 32$ the Goertzel algorithm is preferable if $K \leq 9$. In the case of $N = 128$ Goertzel dominates over the FFT if $K \leq 13$.

In fact, the algorithm introduced in [3] can be even more efficient than this. It combines the good properties of both the FFT and the Goertzel algorithm, producing a DFT decomposition similar to the one used in the split-radix FFT. The dominance of the algorithm of Sorensen and Burrus over the FFT is guaranteed even for $K < N/2$. An experimental comparison of this approach with the Goertzel algorithm showed that the Goertzel algorithm performs actually better than their algorithm when $K \leq 4$ or $K \leq 5$ for a wide range of N . It should be noticed, however, that the algorithm from [3] has to work with a whole data block, and also the complexity being compared does not include rearrangement of the input data sequence.

Using the FFT algorithm requires a memory space of at least $2N$, which contains the real and imaginary parts of signal samples. Also the N values of the transformation kernel, \sin and \cos (so-called twiddle factors), are often precomputed and stored. The FFT calculation itself can be performed with no values being moved in memory (i.e., in-place), however, with regard to the impossibility of starting the computation until the last sample of a block of data is received, a buffer of at least $2N$ in size must be used. In the case of real signals, N memory locations are enough. Thus, the overall FFT memory demand is $4N$ for real signals.

For each considered frequency, the Goertzel algorithm requires: locations for saving two state variables, the real constant B , the real and imaginary parts of the precomputed C , and the real and imaginary parts of the final result. There is no need to implement input buffering, because the computation can be run as the new signal samples arrive. Similarly, the output signal can be overwritten after the last sample has arrived. In many cases it will therefore not be necessary to use buffering at the output side either. The total memory complexity of the Goertzel algorithm is thus $7K$ positions.

Combining all the above together, the Goertzel algorithm will be less memory-demanding than the FFT if

$$\begin{aligned} 7K &< 4N \\ K &< \frac{4}{7}N. \end{aligned} \quad (19)$$

A comparison of (19) and (18) leads to the conclusion that, if we look for a number K for which the Goertzel algorithm dominates over the FFT from both the memory and the computational viewpoints, then: for $N \geq 13$ formula (18) is decisive, because for these N it holds $\frac{4}{7}N > 2\log_2 N$; on the other hand, for $N \in \{2, \dots, 12\}$ (which is unusual in practice), the decisive formula is (19), because for these N it holds $\frac{4}{7}N > 2\log_2 N$; nevertheless, as the difference of the right and the left sides does not exceed 2 in this case, we can conclude, with a small loss of generality, that the comparison of the effectiveness of the two algorithms can be based just on relation (18).

3 Generalized Goertzel algorithm

Formula (2) holds for integer-valued k only. In such a case, the integer number of periods of the transformation kernel, $e^{-j2\pi k \frac{n}{N}}$, corresponds to the signal length N . In the case of $k \in \mathbb{R}$, formulas (1) and (2) are generally no longer in agreement. (The period of the transformation kernel no longer corresponds to N , hence the standard approach cannot be used.)

In Sections 3.1 and 3.2, we will generalize the algorithm such that it includes also the non-integral-valued multiples of the fundamental frequency. The complexity of the novel approach is analyzed in Section 3.3. And, as shown in Section 3.4, the non-integer case can be treated by the standard algorithm using a small trick; however, this is at the cost of increased computational effort.

3.1 Generalizing to non-integer k

In fact, when k is not integer-valued, we can no longer speak of the DFT (1), rather of the discrete-time Fourier transform (DTFT), which is defined by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \omega \in \mathbb{R}. \quad (20)$$

With the notation $\omega_k = 2\pi \frac{k}{N}$ we can write that

$$X(\omega_k) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi k \frac{n}{N}} \quad (21)$$

$$= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n}{N}}, \quad k, \omega_k \in \mathbb{R}, \quad (22)$$

where we exploited the compactness of the support of the signal $\{x[n]\}$.

The derivation of the generalized Goertzel algorithm is analog to the technique presented in Section 2. Compared to that, however, we extend formula (22) at the very beginning by unity in the form of

$$e^{j2\pi k \frac{N}{N}} \cdot e^{-j2\pi k \frac{N}{N}} = 1 \quad \text{for } k \in \mathbb{R}, \quad (23)$$

leading to

$$X(\omega_k) = e^{j2\pi k \frac{N}{N}} \cdot e^{-j2\pi k \frac{N}{N}} \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n}{N}} \quad (24)$$

$$= e^{-j2\pi k \frac{N}{N}} \sum_{n=0}^{N-1} x[n]e^{j2\pi k \frac{N-n}{N}} \quad (25)$$

$$= e^{-j2\pi k} \sum_{n=0}^{N-1} x[n]e^{j2\pi k \frac{N-n}{N}}. \quad (26)$$

Since the sum in (26) is identical to (3), the derivation of the generalized algorithm can proceed using the same steps as in Section 2.1, with one noteworthy change: the equation that characterizes the output using state variables (17) will now be of the form

$$y_k[n] = \left(s[n] - e^{-j\frac{2\pi k}{N}} s[n-1] \right) \cdot e^{-j2\pi k}. \quad (27)$$

Indeed this is so, since the “correction constant”, $e^{-j2\pi k}$, depends only on the index of the frequency component, which remains constant throughout the computation. The complex constant is equal to one for $k \in \mathbb{Z}$, which shows that this is indeed a generalization. In fact, the only variation compared to the standard Goertzel algorithm is the multiplication by this constant at the very end of the algorithm.

The constant $e^{-j2\pi k}$ affects only the phase of the result, not the module. Among other things, this means that

the interest in the modules of the components with non-integer k can be satisfied using the standard algorithm. Indeed, for example [4] uses it in this way. In cases when the phase plays a role (the delay of a signal is detected, for example), however, the use of this “correction constant” is necessary. A short remark can be found in [[1], p. 531], describing the possibility of computing the Goertzel results also for non-integer-valued k ; however, it misleads the reader in that the phase case is not distinguished at all.

3.2 Reducing number of iterations

It will be shown in this section that the last iteration of the Goertzel algorithm can be substituted by merely a single complex multiplication, instead of performing it in the usual manner.

From equation (5) we can express

$$\begin{aligned} y_k[N] &= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n-N}{N}} u[N-n] \\ &= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n-N}{N}} \end{aligned} \quad (28)$$

and also

$$\begin{aligned} y_k[N-1] &= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n-(N-1)}{N}} u[(N-1)-n] \\ &= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k \frac{n-N}{N}} e^{-j2\pi k \frac{1}{N}}. \end{aligned} \quad (29)$$

A comparison of (28) and (29) leads to the formula which characterizes the relationship between the last two samples of the convolution:

$$y_k[N] = y_k[N-1] \cdot e^{+j2\pi \frac{k}{N}}. \quad (30)$$

This means that the very last iteration of the traditional Goertzel algorithm can be replaced by a simple multiplication by $e^{j2\pi \frac{k}{N}}$. Relation (30) holds for $y_k[N]$ and $y_k[N-1]$ due to the limited support of $x[n]$. Nothing similar, however, holds for samples $y_k[N-1]$ and $y_k[N-2]$, due to the term $u[\cdot]$.

Combining $e^{j2\pi \frac{k}{N}}$ and the phase correction constant for non-integer k (see (27)) results in the overall constant

$$D = e^{-j2\pi k} \cdot e^{+j\frac{2\pi k}{N}} = e^{-j\frac{2\pi k}{N}(N-1)}. \quad (31)$$

This way the shortened generalized algorithm is obtained, as is summarized in Figure 4.

```

Inputs: frequency "index"  $k \in \mathbb{R}$ ; signal  $x$  of length  $N$ 
Output:  $y$ , representing  $X(\omega_k)$  according to eq. (20)

%Precalculation of constants
 $A = 2\pi \frac{k}{N}$ 
 $B = 2\cos A$ 
 $C = e^{-jA}$ 
 $D = e^{-j\frac{2\pi k}{N}(N-1)}$ 
%State variables
 $s_0 = 0$ 
 $s_1 = 0$ 
 $s_2 = 0$ 
%Main loop
for  $i = 0 : N - 2$  %one iteration less than traditionally
     $s_0 = x[i] + B \cdot s_1 - s_2$  % (16)
     $s_2 = s_1$ 
     $s_1 = s_0$ 
end
%Finalizing calculations
 $s_0 = x[N - 1] + B \cdot s_1 - s_2$  %corresponds to (16)
 $y = s_0 - s_1 \cdot C$ 
 $y = y \cdot D$  %constant substituting the iteration  $N - 1$ , and correcting the phase at the same
time

```

Figure 4 Generalized Goertzel algorithm with shortened iteration loop. The changes, compared to the standard Goertzel algorithm from Figure 3, are marked in color.

3.3 Computational and memory complexities

The computational complexity of the generalized Goertzel algorithm described in Section 3.1 (without the shortening in Section 3.2) grows by one complex multiplication (i.e., four real multiplications and two real additions) compared to the traditional approach. The memory requirements increase by two positions, which contain the real and imaginary parts of the correction constant $e^{-j2\pi k}$.

Although saving one iteration in the main loop according to Section 3.2 results in lowering the computational effort by two additions and one multiplication, the need for the final complex multiplication cancels such a benefit. This means: there is no advantage in shortening the main loop in case of integer-valued k ; in such a case the traditional algorithm as defined in Section 2 is the most efficient one.

However, in the case of non-integer-valued k the iteration reduction does make sense, since joining the correction constants into a single one (31) leads to the overall growth of computation complexity by three real multiplications (it would be four real multiplications and two real additions if the reduction was not exploited.) Considering the memory, such a case requires two more positions for the real and imaginary parts of (31), compared to the standard algorithm.

It is evident that the computational and memory complexities of the generalized case are only negligibly greater. The main advantage of shortening the loop according to Section 3.2 can be seen in that, for example, in continuous operation, it is not necessary to perform

the last iteration and it is possible to start processing the input sample $x[N]$ in the time spared.

3.4 Yet another approach utilizing standard Goertzel algorithm

It will be shown that, by a trick, the computation required for $k \in \mathbb{R}$ can be transformed into integer-valued problem, where the standard Goertzel algorithm can be utilized—so no modifications are needed. However, it is at the cost of raising the computational complexity, which is even greater than with the generalized Goertzel algorithm (Figure 4).

Starting from (22) again, the $k \in \mathbb{R}$ can be divided into its integer part $\lfloor k \rfloor \in \mathbb{Z}$ and the remainder $\hat{k} \in [0, 1)$, i.e., $k = \lfloor k \rfloor + \hat{k}$. This way, (22) can be rewritten as

$$X(\omega_k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \hat{k} \frac{n}{N}} e^{-j2\pi \lfloor k \rfloor \frac{n}{N}}. \quad (32)$$

If we denote the signal created by multiplying elementwise $\{x[n]\}$ and $\{e^{-j2\pi \hat{k} \frac{n}{N}}\}$ by $\{\hat{x}[n]\}$, the previous relation can be written in the form of

$$X(\omega_k) = \sum_{n=0}^{N-1} \hat{x}[n] e^{-j2\pi \lfloor k \rfloor \frac{n}{N}}, \quad (33)$$

whose right side is a usual DFT of signal \hat{x} (which is complex!) and thus can be computed by the standard Goertzel algorithm.

Regarding the memory complexity, in case of real-time processing it is of advantage to precompute and store the signal $\{e^{-j2\pi k \frac{n}{N}}\}$. It is complex and therefore requires $2N$ memory locations. Furthermore, in contrast to the traditional algorithm, we also need $2N$ positions to store \hat{x} , which is complex (instead of N in the traditional, real case).

To compute \hat{x} we need $2N$ real multiplications. In the standard Goertzel algorithm (Figure 3) the N iterations of the main loop work with real numbers only (supposing the input signal to be real). In the approach presented above, the number of real operations is doubled.

It is therefore clear that the generalized algorithm according to Figure 4 beats the above alternative approach from both the computational and the memory viewpoints.

4 Software

Two Matlab functions are available for download at URL [10].

The function named `goertzel_classic.m` realizes the standard Goertzel algorithm for $k \in \mathbb{Z}$; the generalized (and shortened) algorithm for $k \in \mathbb{R}$ is implemented in the function `goertzel_general_shortened.m`. The structure of the functions corresponds to the pseudocodes in Figures 3 and 4. Indexing the vector elements, however, starts with “1” in Matlab, which differs from our theoretical description, where it starts with “0”.

5 Conclusion

The article presented the generalization of the Goertzel algorithm. The novel approach allows us to employ also the non-integer-valued multiples of the fundamental frequency, making it possible to compute the Fourier transform in discrete-time (DTFT) this way. The main advantage consists in that in various applications where the Goertzel algorithm is utilized, it is no longer necessary to round the frequencies of desire, thus obtaining more accurate results. The article shows that this is reached at the cost of only a negligible rise in computational and memory complexities. Furthermore, it has been shown that the very last iteration of the algorithm can be substituted with a multiplication which is little more effective.

Acknowledgements

This work was supported by projects of the Czech Ministry of Education, Youth and Sports MSM0021630513, the Czech Ministry of Industry and Trade FR-TI2/220, and the Czech Science Foundation 102/09/1846.

Competing interests

The authors declare that they have no competing interests.

Received: 10 May 2011 Accepted: 6 March 2012
Published: 6 March 2012

References

1. RG Lyons, *Understanding Digital Signal Processing*, 2nd edn. (Prentice Hall PTR, NJ, 2004)
2. P Duhamel, M Vetterli, Fast Fourier transforms: A tutorial review and a state of the art. *Signal Process.* **19**, 259 (1990). doi:10.1016/0165-1684(90)90158-U
3. H Sorensen, C Burrus, Efficient computation of the DFT with only a subset of input or output points. *IEEE Transn Signal Process.* **41**(3), 1184 (1993). doi:10.1109/78.205723
4. SL Gay, J Hartung, GL Smith, Algorithms for Multi-Channel DTMF Detection for the WE DSP32 Family, in *IEEE on Proceedings of International Conference on Acoustics, Speech, and Signal Processing* Glasgow, 1134–1137 (1989)
5. Q23, *Technical Features of Push-Button Telephone Sets* (ITU-T, Geneva, 1988)
6. P Mock, Add DTMF generation and decoding to DSP-uP designs, in *Digital Signal Processing Applications with the TMS320 Family*, vol. 1. (Prentice-Hall, NJ, 1987), pp. 543–557
7. G Goertzel, An algorithm for the evaluation of finite trigonometric series. *Am. Math Monthly.* **65**(1), 34 (1958). doi:10.2307/2310304
8. AV Oppenheim, RW Schaffer, JR Buck, *Discrete-time Signal Processing*, 2nd edn. (Prentice-Hall, NJ, 1998)
9. Wikipedia contributors. in *Wikipedia: the Free Encyclopedia*, (Wikipedia Foundation, St. Petersburg, Florida, 2010), http://en.wikipedia.org/wiki/Goertzel_algorithm. 29. 6. 2005, 19. 1. 2010 [cit. 6. 4. 2010]
10. P Rajmic, Matlab codes for the generalized Goertzel algorithm (2012). http://www.mathworks.com/matlabcentral/fileexchange/35103

doi:10.1186/1687-6180-2012-56

Cite this article as: Sysel and Rajmic: Goertzel algorithm generalized to non-integer multiples of fundamental frequency. *EURASIP Journal on Advances in Signal Processing* 2012 **2012**:56.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Immediate publication on acceptance
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com