



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction

[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools

[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

Print/export

[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)

Languages


[Deutsch](#)  
[Italiano](#)  
[עברית](#)  
[Русский](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [More](#) ▾



# Goertzel algorithm

From Wikipedia, the free encyclopedia

The **Goertzel algorithm** is a [Digital Signal Processing](#) (DSP) technique that provides a means for efficient evaluation of individual terms of the [Discrete Fourier Transform](#) (DFT), thus making it useful in certain practical applications, such as recognition of [DTMF](#) tones produced by the buttons pushed on a telephone keypad. The algorithm was first described by [Gerald Goertzel](#) in 1958.<sup>[1]</sup>

Like the DFT, the Goertzel algorithm analyses one selectable frequency component from a [discrete signal](#).<sup>[2][3][4]</sup> Unlike direct DFT calculations, the Goertzel algorithm applies a single real-valued coefficient at each iteration, using real-valued arithmetic for real-valued input sequences. For covering a full spectrum, the Goertzel algorithm has a [higher order of complexity](#) than [Fast Fourier Transform](#) (FFT) algorithms; but for computing a small number of selected frequency components, it is more numerically efficient. The simple structure of the Goertzel algorithm makes it well suited to small processors and embedded applications, though not limited to these.

The Goertzel algorithm can also be used "in reverse" as a sinusoid synthesis function, which requires only 1 multiplication and 1 subtraction per generated sample.<sup>[5]</sup>

## Contents [hide]

- 1 The algorithm
- 2 DFT computations
- 3 Applications
  - 3.1 Power spectrum terms
  - 3.2 Single DFT term with real-valued arithmetic
  - 3.3 Phase detection
  - 3.4 Complex signals in real arithmetic
- 4 Computational complexity
- 5 See also
- 6 References
- 7 Further reading
- 8 External links

## The algorithm [\[edit\]](#)



This section **may require [cleanup](#) to meet Wikipedia's [quality standards](#)**. The specific problem is: **inconsistent styles between equations and labels**. Please help [improve this section](#) if you can.  
*(February 2014)*

The main calculation in the Goertzel algorithm has the form of a [digital filter](#), and for this reason the algorithm is often called a *Goertzel filter*. The filter operates on an input sequence,  $x(n)$ , in a cascade of two stages with a parameter,  $f$ , giving the frequency to be analysed, normalised to cycles per sample.

The first stage calculates an intermediate sequence,  $s(n)$ :

$$(1) s(n) = x(n) + 2 \cos(2\pi f) s(n-1) - s(n-2)$$

The second stage applies the following filter to  $s(n)$ , producing output sequence  $y(n)$ .

$$(2) y(n) = s(n) - e^{-2\pi i f} s(n-1)$$

The first filter stage can be observed to be a second-order [IIR filter](#) with a [direct form](#) structure. This particular structure has the property that its internal state variables equal the past output values from that stage. Input values  $x(n)$  for  $n < 0$  are presumed all equal to 0. To establish the initial filter state so that evaluation can begin at sample  $x(0)$ , the filter states are assigned initial values  $s(-2) = s(-1) = 0$ . To avoid [aliasing hazards](#), frequency  $f$  is often restricted to the range 0 to 1/2 (see [Nyquist–Shannon sampling theorem](#)); using a value outside this range is not meaningless, but is equivalent to using an aliased frequency inside this range, since the exponential function is periodic with a period of 1 cycle per sample in  $f$ .

The second stage filter can be observed to be a [FIR filter](#), since its calculations do not use any of its past outputs.

[Z transform](#) methods can be applied to study the properties of the filter cascade. The Z transform of the first filter stage given in equation (1) is:

$$(3) \frac{S(z)}{X(z)} = \frac{1}{1 - 2\cos(2\pi f)z^{-1} + z^{-2}} = \frac{1}{(1 - e^{+2\pi i f} z^{-1})(1 - e^{-2\pi i f} z^{-1})}.$$

The Z transform of the second filter stage given in equation (2) is:

$$(4) \frac{Y(z)}{S(z)} = 1 - e^{-2\pi i f} z^{-1}$$

The combined transfer function of the cascade of the two filter stages is then

$$(5) \frac{S(z) Y(z)}{X(z) S(z)} = \frac{Y(z)}{X(z)} = \frac{(1 - e^{-2\pi i f} z^{-1})}{(1 - e^{+2\pi i f} z^{-1})(1 - e^{-2\pi i f} z^{-1})} = \frac{1}{1 - e^{+2\pi i f} z^{-1}}.$$

This can be transformed back to an equivalent time domain sequence, and the terms unrolled back to the first input term at index  $n = 0$ .

$$\begin{aligned} y(n) &= x(n) + e^{+2\pi i f} y(n-1) \\ (6) \quad &= \sum_{k=-\infty}^n x(k) e^{+2\pi i f(n-k)} \quad [citation needed] \\ &= e^{+2\pi i f n} \sum_{k=0}^n x(k) e^{-2\pi i f k} \end{aligned}$$

It can be observed that the [poles](#) of the filter's [Z transform](#) are located at  $e^{+2\pi i f}$  and  $e^{-2\pi i f}$ , on a circle of unit radius centered on the origin of the complex [Z transform plane](#). This property indicates that the filter process is [marginally stable](#) and vulnerable <sup>[6]</sup> to [numerical error accumulation](#) when computed using low-precision arithmetic and long input sequences.

## DFT computations [\[edit\]](#)

For the important case of computing a DFT term, the following special restrictions are applied.

- the filtering terminates at index  $n = N$  where  $N$  is the number of terms in the input sequence of the DFT
- the frequencies chosen for the Goertzel analysis are restricted to the special form

$$(7) f = \frac{K}{N}$$

- the index number  $K$  indicating the "frequency bin" of the DFT is selected from the set of index numbers
- $$(8) K \in \{0, 1, 2, \dots, N-1\}$$

Making these substitutions into equation (6), and observing that the term  $e^{+2\pi i K} = 1$ , equation (6) then takes the following form.

$$(9) y(N) = \sum_{k=0}^N x(k) e^{-2\pi i \frac{kK}{N}}$$

We can observe that the right side of equation (9) is extremely similar to the defining formula for DFT term  $X(K)$ , the DFT term for index number  $K$ , but not exactly the same. The summation shown in equation (9) requires  $N+1$  input terms, but only  $N$  input terms are available when evaluating a DFT. A simple but inelegant expedient is to extend the input sequence  $x$  with one more artificial value  $x(N) = 0$ .<sup>[7]</sup> We can see from equation (9), the mathematical effect on the final result is the same as removing term  $x(N)$  from the summation, thus delivering the intended DFT value.

However, there is a more elegant approach that avoids the extra filter pass. From equation (1), we can note that when the extended input term  $x(N) = 0$  is used in the final step,

$$(10) s(N) = 2\cos(2\pi f)s(N-1) - s(N-2)$$

Thus, the algorithm can be completed as follows:

- terminate the IIR filter after processing input term  $x(N-1)$
- apply equation (10) to construct  $s(N)$  from the prior outputs  $s(N-1)$  and  $s(N-2)$

- apply equation (2) with the calculated  $s(N)$  value, and with  $s(N - 1)$  produced by the final direct calculation of the filter.

The last two mathematical operations are simplified by combining them algebraically.

$$\begin{aligned}
 y(N) &= s(N) - e^{-2\pi i \frac{K}{N}} s(N - 1) \\
 (11) \quad &= (2 \cos(2\pi f) s(N - 1) - s(N - 2)) - e^{-2\pi i \frac{K}{N}} s(N - 1) \\
 &= e^{2\pi i \frac{K}{N}} s(N - 1) - s(N - 2)
 \end{aligned}$$

Note that stopping the filter updates at term  $N - 1$  and immediately applying equation (2) rather than equation (11) misses the final filter state updates, yielding a result with incorrect phase.<sup>[8]</sup>

The particular filtering structure chosen for the Goertzel algorithm is the key to its efficient DFT calculations. We can observe that only one output value  $y(N)$  is used for calculating the DFT, so calculations for all the other output terms are omitted. Since the FIR filter is not calculated, the IIR stage calculations  $s(0)$ ,  $s(1)$ , etc. can be discarded immediately after updating the first stage's internal state.

This seems to leave a paradox: to complete the algorithm, the FIR filter stage must be evaluated once using the final two outputs from the IIR filter stage, while for computational efficiency the IIR filter iteration discards its output values. This is where the properties of the direct-form filter structure are applied. The two internal state variables of the IIR filter provide the last two values of the IIR filter output, which are the terms required to evaluate the FIR filter stage.

## Applications [\[edit\]](#)

### Power spectrum terms [\[edit\]](#)

Examining equation (6), a final IIR filter pass to calculate term  $s(N)$  using a supplemental input value  $x(N) = 0$  applies a complex multiplier of magnitude 1.0 to the previous term  $s(N - 1)$ . Consequently,  $s(N)$  and  $s(N - 1)$  represent equivalent signal power. It is equally valid to apply equation (11) and calculate the signal power from term  $y(N)$ , or to apply equation (2) and calculate the signal power from term  $y(N - 1)$ . Both cases lead to the following expression for the signal power represented by DFT term  $X(K)$ .

$$\begin{aligned}
 X(K)X'(K) &= y(N) y'(N) = y(N - 1) y'(N - 1) \\
 (12) \quad &= s(N - 1)^2 + s(N - 2)^2 - 2 \cos(2\pi \frac{K}{N}) s(N - 1) s(N - 2)
 \end{aligned}$$

In the [pseudocode](#) below, the variables `sprev` and `sprev2` temporarily store output history from the IIR filter, while `x[n]` is an indexed element of the [array](#) `x` which stores the input.

```

Nterms defined here
Kterm selected here
ω = 2 * π * Kterm / Nterms;
cr = cos(ω);
ci = sin(ω);
coeff = 2 * cr;

sprev = 0;
sprev2 = 0;
for each index n in range 0 to Nterms-1
    s = x[n] + coeff * sprev - sprev2;
    sprev2 = sprev;
    sprev = s;
end

power = sprev2*sprev2 + sprev*sprev - coeff*sprev*sprev2 ;

```

It is possible<sup>[9]</sup> to organise the computations so that incoming samples are delivered singly to a [software object](#) that maintains the filter state between updates, with the final power result accessed after the other processing is done.

### Single DFT term with real-valued arithmetic [\[edit\]](#)

The case of real-valued input data arises frequently, especially in embedded systems where the input streams result from direct measurements of physical processes. Comparing to the illustration in the previous section,

when the input data are real-valued, the filter internal state variables  $s_{prev}$  and  $s_{prev2}$  can be observed also to be real-valued, consequently, no complex arithmetic is required in the first IIR stage. Optimizing for real-valued arithmetic typically is as simple as applying appropriate real-valued data types for the variables.

After the calculations using input term  $x(N-1)$ , and filter iterations are terminated, equation (11) must be applied to evaluate the DFT term. The final calculation uses complex-valued arithmetic, but this can be converted into real-valued arithmetic by separating real and imaginary terms.

$$(13) \quad \begin{aligned} c_r &= \cos(2\pi \frac{K}{N}) \\ c_i &= \sin(2\pi \frac{K}{N}) \\ y(N) &= c_r s(N-1) - s(N-2) + i c_i s(N-1) \end{aligned}$$

Comparing to the power spectrum application, the only difference is the calculation used to finish.

```
(Same IIR filter calculations as in the signal power implementation)
XKreal = spreve * cr - spreve2;
XKimag = spreve * ci;
```

### Phase detection [\[edit\]](#)

This application requires the same evaluation of DFT term  $X(K)$ , as discussed in the previous section, using a real-valued or complex-valued input stream. Then the signal phase can be evaluated as:

$$(14) \quad \phi = \tan^{-1} \frac{\Im(X(K))}{\Re(X(K))}$$

taking appropriate precautions for singularities, quadrant, and so forth when computing the inverse tangent function.

### Complex signals in real arithmetic [\[edit\]](#)

Since complex signals decompose linearly into real and imaginary parts, the Goertzel algorithm can be computed in real arithmetic separately over the sequence of real parts, yielding  $y_r(n)$ ; and over the sequence of imaginary parts, yielding  $y_i(n)$ . After that, the two complex-valued partial results can be recombined:

$$(15) \quad y(n) = y_r(n) + i y_i(n)$$

## Computational complexity [\[edit\]](#)

- According to [computational complexity theory](#), computing a set of  $M$  DFT terms using  $M$  applications of the Goertzel algorithm on a data set with  $N$  values with a "cost per operation" of  $K$  has [complexity](#)  $O(KNM)$ .

To compute a single DFT bin  $X(f)$  for a complex input sequence of length  $N$ , the Goertzel algorithm requires  $2N$  multiplications and  $4N$  additions/subtractions within the loop, as well as 4 multiplications and 4 final additions/subtractions, for a total of  $2N + 4$  multiplications and  $4N + 4$  additions/subtractions. This is repeated for each of the  $M$  frequencies.

- In contrast, using an FFT on a data set with  $N$  values has complexity  $O(KN \log N)$ .<sup>[[citation needed](#)]</sup> This is harder to apply directly because it depends on the FFT algorithm used, but a typical<sup>[[citation needed](#)]</sup> example is a radix-2 FFT, which requires  $2 \log_2 N$  multiplications and  $3 \log_2 N$  additions/subtractions per DFT bin, for each of the  $N$  bins.<sup>[[citation needed](#)]</sup>

In the complexity order expressions, when the number of calculated terms  $M$  is smaller than  $\log N$ , the advantage of the Goertzel algorithm is clear. But because FFT code is comparatively complex,<sup>[[citation needed](#)]</sup> the "cost per unit of work" factor  $K$  is often larger for an FFT,<sup>[[citation needed](#)]</sup> and the practical advantage favours the Goertzel algorithm even for  $M$  several times larger than  $\log_2 N$ .

As a rule-of-thumb for determining whether a radix-2 FFT or a Goertzel algorithm is more efficient, adjust the number of terms  $N$  in the data set upward to the nearest exact power of 2, calling this  $N_2$ , and the Goertzel algorithm is likely to be faster if

$$M \leq \frac{5N_2}{6N} \log_2 N_2$$
 <sup>[*citation needed*]</sup>

FFT implementations and processing platforms have a significant impact on the relative performance. Some FFT implementations<sup>[10]</sup> perform internal complex-number calculations to generate coefficients on-the-fly, significantly increasing their "cost K per unit of work." FFT and DFT algorithms can use tables of pre-computed coefficient values for better numerical efficiency, but this requires more accesses to coefficient values buffered in external memory, which can lead to increased cache contention that counters some of the numerical advantage.<sup>[*citation needed*]</sup>

Both algorithms<sup>[*citation needed*]</sup> gain approximately a factor of 2 efficiency<sup>[*citation needed*]</sup> when using real-valued rather than complex-valued input data. However, these gains are natural for the Goertzel algorithm but will not be achieved for the FFT without using certain algorithm variants<sup>[*which?*]</sup> specialised for [transforming real-valued data](#).

## See also <sup>[*edit*]</sup>

- Bluestein's FFT algorithm (chirp-Z)
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)

## References <sup>[*edit*]</sup>

- ↑ Goertzel, G. (January 1958), "An Algorithm for the Evaluation of Finite Trigonometric Series" <sup>[*?*]</sup>, *American Mathematical Monthly* **65** (1): 34–35, doi:10.2307/2310304 <sup>[*?*]</sup>
- ↑ Mock, P. (March 21, 1985), "Add DTMF Generation and Decoding to DSP-μP Designs" <sup>[*?*]</sup> (PDF), *EDN*, ISSN 0012-7515 <sup>[*?*]</sup>; also found in DSP Applications with the TMS320 Family, Vol. 1, Texas Instruments, 1989.
- ↑ Chen, Chiouguey J. (June 1996), *Modified Goertzel Algorithm in DTMF Detection Using the TMS320C80 DSP* <sup>[*?*]</sup> (PDF), Application Report, SPRA066, Texas Instruments
- ↑ Schmer, Gunter (May 2000), *DTMF Tone Generation and Detection: An Implementation Using the TMS320C54x* <sup>[*?*]</sup> (PDF), Application Report, SPRA096a, Texas Instruments
- ↑ http://cs-www.cs.yale.edu/c2/images/uploads/AudioProc-TR.pdf <sup>[*?*]</sup>
- ↑ Gentleman, W. M. (1 February 1969). "An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients" <sup>[*?*]</sup> (PDF). *The Computer Journal* **12** (2): 160–164. doi:10.1093/comjnl/12.2.160 <sup>[*?*]</sup>. Retrieved 28 December 2014.
- ↑ "Goertzel's Algorithm" <sup>[*?*]</sup>. Cnx.org. 2006-09-12. Retrieved 2014-02-03.
- ↑ "Electronic Engineering Times | Connecting the Global Electronics Community" <sup>[*?*]</sup>. EE Times. Retrieved 2014-02-03.
- ↑ "Efficiently detecting a frequency using a Goertzel filter". *http://netwerkt.wordpress.com/2011/08/25/goertzel-filter/* <sup>[*?*]</sup>.
- ↑ Press; Flannery; Teukolsky; Vetterling (2007), "Chapter 12", *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press





## Further reading [[edit](#)]

- Proakis, J. G.; Manolakis, D. G. (1996), *Digital Signal Processing: Principles, Algorithms, and Applications*, Upper Saddle River, NJ: Prentice Hall, pp. 480–481

## External links [[edit](#)]

- <http://netwerkt.wordpress.com/2011/08/25/goertzel-filter/> [[↗](#)] C Code implementation of iterative Goertzel algorithm
- <http://www.embedded.com/design/configurable-systems/4006427/A-DSP-algorithm-for-frequency-analysis> [[↗](#)] (Java [[↗](#)]) A DSP algorithm for frequency analysis - the Chirp-Z Transform (CZT)

Categories: [FFT algorithms](#) | [Digital signal processing](#)

This page was last modified on 15 July 2015, at 17:44.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

