# Analysis of Goertzel Filter Variants for Low-Power and Memory-Constrained Frequency Analysis

Robert Fromm
*Faculty of Engineering*
*Leipzig University of Applied Sciences*
Leipzig, Germany
0000-0002-2905-0648

Olfa Kanoun
*Department of Electrical Engineering and Information Technology*
*Chemnitz University of Technology*
Chemnitz, Germany
0000-0002-7166-1266

Faouzi Derbel
*Faculty of Engineering*
*Leipzig University of Applied Sciences*
Leipzig, Germany
0000-0002-7038-8157



***Abstract*—Sensor-integrating machine elements and embedded condition monitoring systems generate large vibration data volumes, which strongly increase memory demand and energy requirements for wireless transmission. This article investigates the Goertzel filter preprocessing and compression method. The Goertzel filter computes individual frequency bins and requires only two memory cells independent of sample count. This property enables parallel evaluation of multiple frequencies without external buffering and significant compression of the data. Four Goertzel filter variants are analyzed. These include the original second-order filter, the first-order complex-valued filter, and the two Reinsch modifications. Numerical investigations quantify maximum signal amplitudes and accuracy for fixed-point and floating-point arithmetic. The first-order filter limits amplitudes significantly and provides more uniform accuracy across the frequency range at the cost of a fourfold increase in multiplications. Reinsch modifications reduce errors only for floating-point implementations. All filter variants are implemented on four microcontrollers with different architectures, including MSP430 and STM32 devices with and without hardware multipliers and floating-point units. Measured cycle counts show up to a factor 9 speedup from hardware multipliers and more than a reduction by factor 13 from floating-point units. The results provide quantitative guidelines for selecting Goertzel filter variants, arithmetic formats, and microcontroller platforms for energy-efficient vibration data preprocessing in sensor-integrating machine elements.**

***Index Terms*—Sensor-integrating machine element, condition monitoring, wireless sensor networks, microcontrollers, signal processing, numerical accuracy.**

## I. Introduction

Condition-based monitoring of critical machine elements is essential for preventing unplanned downtimes and for limiting safety and cost impacts. The continuous observation of component condition enables maintenance actions to be scheduled only when required. This approach forms the basis of predictive maintenance strategies. Such strategies reduce the likelihood of unexpected equipment failures. Published studies report that predictive maintenance can reduce unplanned downtime by up to 50%. [1]

Vibration analysis is a common technique in condition based monitoring of rotating machinery because many mechanical faults cause measurable changes in the vibration signature [1]. Gears are especially prone to damage from repeated meshing contact. Over time, surface wear or pitting develops on the gear teeth. These effects lead to characteristic changes in the vibration signal [2].

Measuring vibration close to the source of generation improves the effectiveness of fault detection. Accelerometers mounted on the machine housing capture vibrations after propagation through the whole machinery. This propagation attenuates and distorts the signal. Embedding sensors directly into machine elements enables measurements closer to the point of interest. [3]

This approach has led to the development of so called sensor-integrating machine elements (SiMEs). These are standard mechanical components equipped with integrated sensors and associated electronics. SiMEs measure parameters such as load, vibrations, and temperatures. The measured data is transmitted wirelessly. Wireless transmission is required for sensors integrated into rotating or otherwise inaccessible components. [1]

Implementing SiMEs introduces several challenges. The embedded sensor node's electronics must fit into an extremely compact form factor. The power supply is a critical aspect [4]. Long battery lifetimes on the order of months to years or the use of energy harvesting are required [5].

Multiple SiMEs generate large volumes of vibration data. For example, our measurement system used in planetary gears produces approximately 1 MB of data per recording [6]. The sensor-integrating gas foil bearing developed by Kliemank et al. [7] required a sampling rate of 2.5 MHz, resulting in very large data sets, as-well. Wireless transmission of complete data sets is highly energy demanding. Local data preprocessing,

Research funded by the German Research Foundation (DFG) under project number 466653706 as part of the project "Sensor-integrating Gear" (SIZA).

compression, and feature extraction is therefore required to reduce the amount of transmitted data. [8]

This article focuses on the well-known Goertzel filter. The feasibility of using the Goertzel filter in SiMEs for compressing measured vibration data is examined. A discrete Fourier transform converts the complete data set into the frequency domain. The input and output lengths are identical, so no compression is achieved. The Goertzel filter computes only a single frequency bin [9]. When the frequency bin is carefully preselected, for example based on multiples of gear mesh frequencies, the amount of transmitted data can be significantly reduced.

This article presents an analysis of the Goertzel filter and its use in SiMEs. Since SiMEs rely on low power microcontrollers, fixed-point implementations of the Goertzel filter are examined. The filter output response is analyzed to derive the maximum required integer range and to prevent numerical overflows. Fixed-point and floating-point implementations are compared with respect to numerical accuracy. The comparison includes the first-order Goertzel filter [10], the original second-order Goertzel filter [9], and two modified second-order filters proposed by Reinsch [11, pp. 90f]. All Goertzel filter variants are implemented in fixed-point and floating-point arithmetic on four different microcontrollers. These microcontrollers provide different combinations of hardware multipliers and floating-point units (FPUs). The required number of CPU cycles per filter iteration differs significantly between microcontrollers.

## II. State of Research

In 1958, Goertzel [9] published a second-order recursive filter for computing a single frequency bin $\omega'$ of the discrete Fourier transform . For a time-discrete input signal $x[k]$, the recursive part of the filter is evaluated with zero initial conditions and is give by:

$$y[n] = x[n] + 2y[n-1]\cos\omega' - y[n-2]. \tag{1}$$

The real part $\mathrm{Re}\{X(\omega')\}$ and the imaginary part $\mathrm{Im}\{X(\omega')\}$ of the Fourier-transformed input signal can be calculated continuously or after $N$ input samples by:

$$\mathrm{Re}\left\{X(\omega')\right\} = y[N-1]\cos\omega' - y[N-2], \tag{2}$$

$$\mathrm{Im}\left\{X(\omega')\right\} = y[N-1]\sin\omega'. \tag{3}$$

Based on these equations, the magnitude, phase, or power of the frequency bin can be calculated using standard formulas for complex numbers.

In 1969, Gentleman published a floating-point error analysis of the second-order Goertzel filter [12]. Based on this error analysis, Reinsch proposed a modification of the second-order Goertzel filter to improve accuracy in floating-point calculations [11, pp. 90f]. Reinsch proposed two different modifications that are intended for lower frequencies $\omega' < \pi/2$ and higher frequencies $\omega' > \pi/2$, respectively. The lower-frequency modification uses the difference signal $\tilde{y}[n]$ to improve numerical accuracy. With $\tilde{y}[n] = y[n] - y[n-1]$, Eq. 1 is rewritten as:

$$\tilde{y}[n] = x[n] + (2\cos\omega' - 2)y[n-1] + \tilde{y}[n-1] \tag{4}$$

For higher-frequency signals, a sum variable $\tilde{y}[n] = y[n] + y[n+1]$ is used instead. In this case, Eq. 1 yields [11]:

$$\tilde{y}[n] = x[n] + (2\cos\omega' + 2)y[n-1] - \tilde{y}[n-1]. \tag{5}$$

All Goertzel filters are marginally stable infinite-impulse-response filters. Their poles are located at $\exp(\pm\mathrm{j}\omega')$. This pole placement can cause a significant growth of the filter output for certain input signals.

Beraldin and Steenaart [10] derived formulas for the maximum output amplitudes. These limits are visualized later in our analysis. Additionally, Beraldin and Steenaart investigated the first-order Goertzel filter. This filter is based on complex-valued output signals $y[n]$. The difference equation of the first-order Goertzel filter is given by:

$$y[n] = x[n] + \exp(\mathrm{j}\omega')y[n-1]. \tag{6}$$

The disadvantage of the first-order filter is the increased number of real-valued multiplications. Four multiplications are required to evaluate the complex multiplication $\exp(\mathrm{j}\omega')y[n-1]$. However, the first-order filter provides a clear advantage for fixed-point implementations. The maximum output values are generally lower.

Several applications of the Goertzel filter have been reported across different domains. In telecommunication systems, the filter is commonly used for dual-tone multi-frequency (DTMF) signal detection and related signal processing tasks [13]–[15]. Further implementations exits, for example a software-defined radio for the DCF77, the German longwave time signal [16]. Low-power acoustic communication systems, including underwater robot communication, also rely on the second-order Goertzel filter [17]. In addition, the original Goertzel filter has been evaluated alongside other spectral analysis methods, such as the fast Fourier transform, windowed approaches, and the discrete Fourier transform, for applications including impedance spectroscopy [18].

The Goertzel filter is also used in condition monitoring applications. Wireless sensor networks for structural health monitoring employ the filter to reduce the amount of transmitted data [19]. Implementations for eddy current sensing have been realized on field-programmable gate arrays (FPGAs), although such platforms typically exhibit higher power consumption than microcontrollers with sleep modes [8], [20]. Eddy current sensors are nevertheless relevant for SiME concepts [21]. Further condition monitoring systems apply the Goertzel filter to monitor components such as aluminum-electrolyte capacitors in power converters [22].

The literature indicates that the Goertzel filter is widely used in embedded systems and in condition monitoring. Several implementations for low-power systems are available. However, no clear and systematic comparison between fixed-point and floating-point implementations was identified. In addition,

many references use the original second-order Goertzel filter. Seldome, the Reinsch modifications or the first-order filter are implemented in low-power systems. This article investigates these gaps and derives recommendations for Goertzel filter implementations on low-power microcontrollers.

## III. Numerical Investigations

### A. Methods

The first part of the investigation is based on a numerical analysis of the Goertzel filters: the first-order Goertzel filter (G1) and the second-order Goertzel filter (G2). In addition, the Reinsch modifications are examined. The modification for lower frequencies with $\omega' < \pi/2$ is abbreviated as R1 and the modification for higher frequencies is abbreviated as R2.

The numerical analyses are performed in Python. The input signals are limited to the range $[-1,1]$. For the second-order Goertzel filters, the recursive computations follow the previously defined difference equations. As a final step, the complex-valued Fourier transform is calculated using Eq. 2 and 3.

The Goertzel filters are marginally stable. The maximum output values occur for sinusoidal input signals with a frequency equal to the filter resonance frequency $\omega'$.

$$x[k] = \sin \omega' k \tag{7}$$

The maximum output signal is analyzed for all frequencies in the range $0 < \omega' < \pi$. The selected frequency $\omega'$ corresponds to a valid Fourier frequency and is an integer multiple of $2\pi/N$. The DC value with $\omega' = 0$ is deliberately excluded. For this case, the second-order Goertzel filter produces a quadratic output behavior with $y[k] \propto k^2$. This output magnitude is significantly larger. Excluding $\omega' = 0$ therefore allows the use of a smaller integer range in fixed-point implementations. For acceleration measurements, the DC component is often not relevant [23]. Alternatively, the DC offset can be obtained using other methods more effectively.

The second part of the numerical investigation focuses on the accuracy of the Goertzel filter output. An sinusoidal input signal of frequency $\omega'$ is applied to all filters. The resulting Fourier-transformed output $X(\omega')$ is calculated. The input signal has an amplitude of 1 and a phase of 0°. The error is therefore defined as $\log_2 |X(\omega') - 1|$. The logarithmic representation of the error is used to determine the effective number of valid fractional bits.

The error analysis is performed for all four filter variants and for different data sizes. For fixed-point arithmetic, the number of fractional bits is varied. The SPFPM[1] library is used for this purpose . For floating-point arithmetic, data types provided by NumPy are applied. These include 16-bit half-precision, 32-bit single-precision, and 64-bit double-precision floating-point formats.

[1] see https://github.com/rwpenney/spfpm

### B. Fixed-Point Results

Fig. 1 shows typical filter responses to a sinusoidal input signal with amplitude 1 and frequency $\omega'$. All filter responses exhibit a sinusoidal waveform with a linearly increasing envelope. The green trace in Fig. 1 represents the difference signal $\tilde{y}[k]$ of the low-frequency Reinsch modification R1. The output signals of both Reinsch modifications are identical to the output of the second-order Goertzel filter.

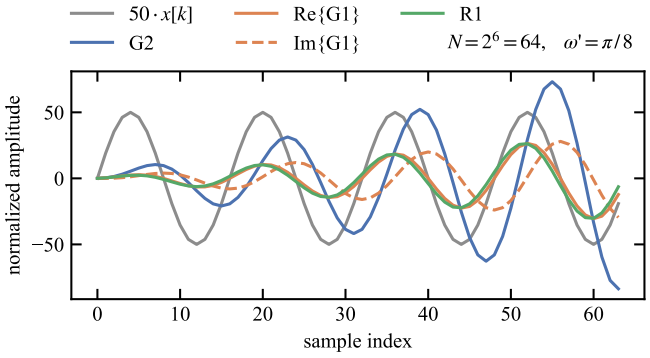


Fig. 1. Sinusoidal input signal and corresponding output signals of the first-order (G1) and second-order (G2) Goertzel filters, including the difference signal $\tilde{y}[k]$ of the low-frequency Reinsch modification

Fig. 2 shows the maximum output values of the different Goertzel filters for $N = 2^{10}$. The maximum output is shown on a logarithmic scale to indicate the number of integer bits required for a fixed-point implementation. For the Reinsch modifications, the maximum of the difference and sum signals $\tilde{y}[k]$ are displayed.

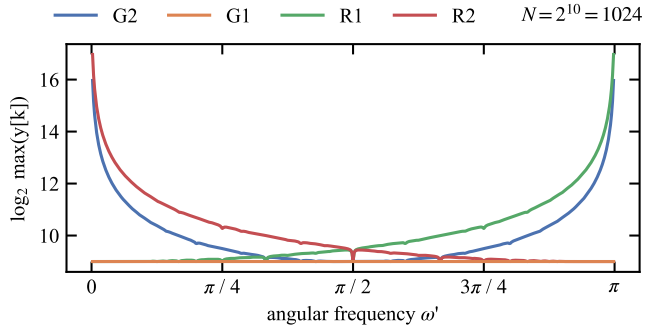


Fig. 2. Maximum output signal of the different Goertzel filters. R1 and R2 denote the difference and sum signals $\tilde{y}[k]$ of the Reinsch modifications.

The maximum output of the first-order Goertzel filter is constant for all frequencies and equals $N/2$. For the second-order Goertzel filter, the maximum output is given by $(N+2)/\sin\omega'$. This relation results in very large output values near $\omega' = 0$ and $\pi$. An increase in $N$ further amplifies the term $1/\sin\omega'$, as $\sin\omega'$ comes closer to 0. This behavior leads to extremely high output values. When the first Reinsch modification is applied only for $\omega' < \pi/2$ and the second Reinsch modification only for $\omega' > \pi/2$, the signals $\tilde{y}[k]$ remain limited. However, the output of all second-order filters are identical.

Fig. 3 shows the error distribution for the second-order Goertzel filters in the upper subplot and for the first-order Goertzel filter in the lower subplot. Different colors represent different numbers of fractional bits. For fixed-point implementations, the Reinsch modifications exhibit the same error behavior as the second-order Goertzel filter.

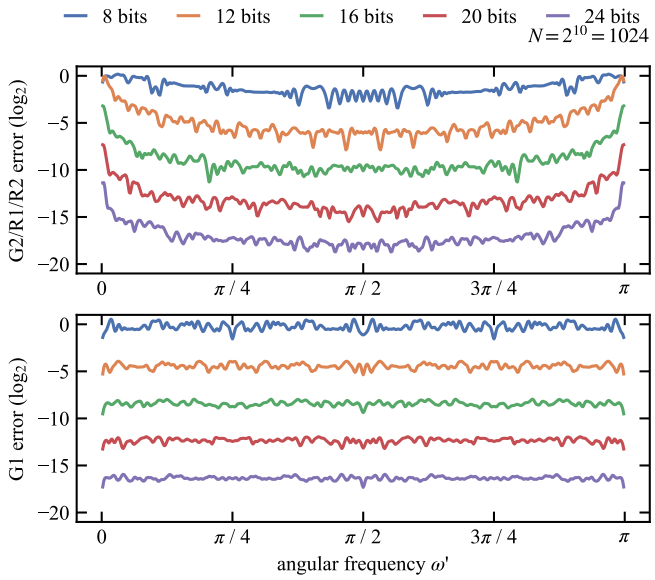


Fig. 3. Error distribution for different fractional bit counts of the fixed-point implementations. The first subplot shows the second-order Goertzel filter and both Reinsch modifications, which exhibit near identical error behavior. The second subplot shows the error distribution of the first-order filter.

With a base-2 logarithmic scale on the y-axis, the effective number of fractional bits becomes directly visible. To achieve errors below 1% at least 16 fractional bits are required for $N = 2^{10}$. For the second-order filters, frequencies close to $\sin\omega' = 0$ require at least 24 fractional bits.

Fig. 4 shows the maximum and mean error as a function of the sample count $N$. The results for all second-order filters are shown first. Only the mean error exhibits an approximately linear trend for high-precision integer representations. The error behavior of the first-order Goertzel filter is more consistently linear. The linear trend observed in the plot has a first derivative close to 1.

### C. Floating-Point Results

For the floating-point implementations, the impact of numerical precision is investigated. Fig. 5 shows the error distribution across frequency for three floating-point precisions using the original second-order Goertzel filter. For 16-bit half-precision arithmetic, numerical overflows occur at frequencies close to $\sin\omega' = 0$.

For the selected sample count of $N = 2^{12}$, the error of half-precision floating-point arithmetic is very high. In contrast, single-precision floating-point arithmetic maintains errors below 1.9% across all frequencies. The error of double-precision floating-point arithmetic is extremely small.

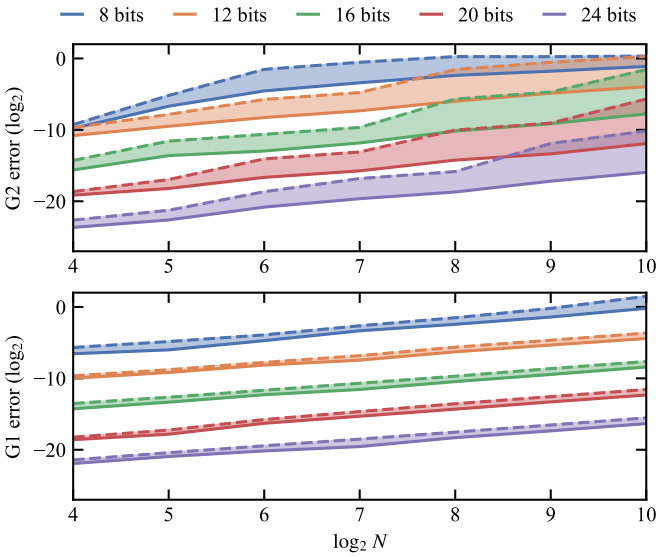


Fig. 4. Relationship between the fixed-point error and the sample count $N$. The dashed line indicates the maximum error, while the solid line shows the mean error over the entire frequency range.

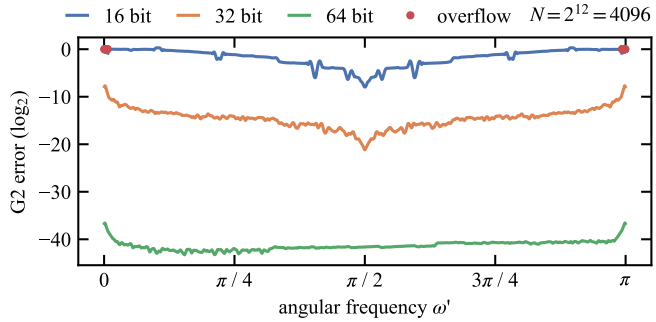


Fig. 5. Error distribution of the second-order Goertzel filter for different floating-point precisions

Fig. 6 shows the error distribution for the different filters. Only 32-bit floating-point precision is shown. The same sample count is used as in Fig. 5. The original second-order Goertzel filter exhibits the error peaks near $\sin\omega' = 0$. The first-order Goertzel filter shows an approximately constant error level. This level is comparable to the second-order filter in the range $\pi/4 < \omega' < 3\pi/4$. The Reinsch modifications effectively reduce the error within their respective frequency ranges.

A similar investigation is performed for the floating-point error as a function of increasing sample count, similar to Fig. 4. The mean error follows a linear trend with a first derivative close to 1.

## IV. Microcontroller Implementations

### A. Methods

Based on the results of the numerical analysis, the different Goertzel filters are implemented on several microcontrollers. For all implementations, the parameters are fixed to $N = 2^{10}$

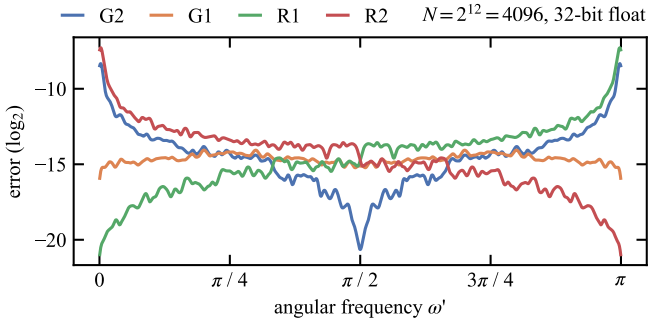


Fig. 6. Error distribution of all investigated Goertzel filters for 32-bit floating-point arithmetic

and $\omega' = \pi/8$. This frequency lies in the lower half of the spectrum. Therefore, only the first Reinsch modification is implemented.

For the fixed-point implementations, 16 fractional bits are used. As shown in Fig. 3, this configuration yields mean errors well below 0.5%. All implementations use 32-bit integers. These consist of one sign bit and 15 integer bits. As shown in Fig. 2, 15 integer bits are sufficient to avoid overflows at $\omega' = \pi/8$.

The input signal and the filter coefficients are scaled to 16 fractional bits. The Goertzel filter results calculated by the microcontrollers are verified against the Python implementation. A digital output pin is enabled during the loop iterations of each filter. The processing time is estimated by measuring the pulse width of this output. The measured time is multiplied by the CPU clock frequency and divided by $N$ to obtain the number of CPU cycles per filter iteration.

The selection of microcontrollers is based on a previous study that analyzed processing performance and sleep current for SiME applications [24]. The MSP430F2274 and MSP430FR5994 are selected due to their very low sleep current, even at elevated temperatures. Only the MSP430FR5994 provides a hardware multiplier. The STM32L053 achieves a similarly low sleep current. It also offers higher computational performance and includes a hardware multiplier. The STM32L452 is selected because it provides both a hardware multiplier and a floating-point unit (FPU). However, its sleep current increases significantly at higher temperatures. Other microcontrollers evaluated in [24] are not considered further. These devices either provide limited additional benefit for this study or exhibit very high sleep currents. An example is the STM32U575, which draws 950 µA at 125 °C.

### B. Results

Tab. I shows the number of CPU cycles required for a single iteration of the Goertzel filters.

The complex-valued multiplication required by the first-order Goertzel filter consists of four real-valued multiplications. As a result, the required cycle counts are generally increased by approximately a factor of four. The MSP430F2274 exhibits the highest cycle counts. This behavior is caused by the absence of hardware acceleration and by the 16-bit architecture used by both MSP430 processors. For the MSP430FR5994, the impact of the hardware multiplier is clearly visible. The number of required cycles is reduced by up to a factor of nine.

TABLE I
NUMBER OF CPU CYCLES PER FILTER ITERATION

| Device | G1 | G2 | R1 | G1 | G2 | R1 |
|---|---|---|---|---|---|---|
| | 32-bit fixed-point | | | 32-bit floating-point | | |
| MSP430F2274 | 4242 | 1092 | 1023 | 2059 | 961 | 942 |
| MSP430FR5994 | 676 | 121 | 125 | 2148 | 1014 | 1008 |
| STM32L053 | 452 | 135 | 138 | 983 | 527 | 526 |
| STM32L452 | 176 | 49 | 51 | 50 | 40 | 36 |

Due to the 32-bit architecture of the STM32L053, it is generally more efficient than the MSP430FR5994. Although no FPU is integrated, the cycle counts for floating-point filters are approximately halved. The STM32L452 achieves the highest performance. For fixed-point implementations, the required cycle counts are reduced by more than factor two, which is likely caused by a more efficient hardware multiplier. The STM32L452 is the only microcontroller in this study that integrates an FPU. This feature reduces the cycle counts by at least an additional factor of 13. The STM32L452 also achieves lower cycle counts for floating-point implementations than for their fixed-point counterparts.

## V. CONCLUSION

Sensor-integrating machine elements (SiMEs) that measure vibrations can generate large volumes of data. Temporary buffering in external memory may be required to prevent data loss during wireless transmission to mitigate packet loss. Wireless transmission of such data consumes a substantial share of the SiME energy budget. Local data preprocessing or compression is therefore required to mitigate these limitations.

This article investigates the Goertzel filter as a method for local data preprocessing to reduce the required data transmission. In comparison to discrete Fourier transform implementations such as the fast Fourier transform, the Goertzel filter computes only a single frequency bin rather than the complete spectrum. The fast Fourier transform requires at least the sample count $N$ as memory storage [25]. In contrast, the Goertzel filter requires only two memory cells, independent of $N$.

By operating multiple Goertzel filters in parallel, several frequency bins can be evaluated without reaching the memory requirements of Fourier transform based methods. A limitation of the Goertzel filter is that the target frequency must be known prior to computation. This requirement can be addressed by performing an angular speed estimation before applying the Goertzel filters [7]. The external memory to buffer the measurement data can be fully eliminated, when running multiple Goertzel filters in parallel to the data acquisition.

This article investigates four different implementations of the Goertzel filter: original second-order filter, first-order filter, and the two Reinsch modifications. All filter variants

are evaluated numerically using fixed-point and floating-point arithmetic. The filters are also implemented on several microcontrollers. The selected microcontrollers include devices with and without a hardware multiplier and with and without an floating-point unit (FPU).

Fixed-point implementations have the disadvantage that the integer range must be chosen carefully. Overflows can occur due to the marginally stable nature of the Goertzel filters. Based on the required CPU cycles per filter iteration shown in Tab. I, fixed-point implementations are effective only on microcontrollers that provide a hardware multiplier. For fixed-point arithmetic, the first-order Goertzel filter offers reduced maximum internal values. This property requires fewer integer bits and allows a higher number of fractional bits, which improves numerical accuracy. The accuracy is also more uniform across the frequency range compared to second-order filters. The disadvantage of the first-order Goertzel filter is the increased computational effort.

The Reinsch modifications do not provide benefits for fixed-point implementations. The filter output reaches the same values as for the original second-order filter and no improvement in numerical accuracy is observed. The second-order Goertzel filter is preferable when maximum execution speed is required. A sufficient number of integer bits must be reserved to prevent overflow, as shown in Fig. 2. Alternatively, the selected filter frequency $\omega'$ can be restricted to values sufficiently far from 0 and $\pi$ to avoid large output amplitudes.

Floating-point arithmetic reduces overflow issues due to the large numerical range of single-precision formats. The use of floating-point arithmetic should therefore be considered even when no FPU is available in the CPU. In particular, on platforms without a hardware multiplier, floating-point implementations exhibit shorter execution times. The accuracy of 32-bit floating-point arithmetic is limited, as shown in Fig. 5, and degrades with increasing sample count $N$.

For floating-point arithmetic, the Reinsch modifications are effective, as shown in Fig. 6. They reduce the numerical error within their respective frequency ranges. No additional computational effort is required for these modifications. In some cases, they are even slightly faster than the original Goertzel filter.

This study provides a visual overview of the working principles of the different Goertzel filters and their fixed-point and floating-point implementations. The presented graphs and measurement results enable the selection of an appropriate fixed-point scaling. They also support the estimation of numerical accuracy. In addition, the provided results allow the computation time of a reader's specific implementation to be assessed.

## REFERENCES


[1] E. Kirchner, T. Wallmersperger, T. Gwosch, J. D. M. Menning, J. Peters, R. Breimann, B. Kraus, P. Welzbacher, J. Küchenhof, D. Krause, E. Knoll, M. Otto, B. Muhammedi, S. Seltmann, A. Hasse, G. Schäfer, A. Lohrengel, S. Thielen, Y. Stiemcke, O. Koch, A. Ewert, T. Rosenlöcher, B. Schlecht, A. Prokopchuk, E. M. Henke, F. Herbst, S. Matthiesen, D. Riehl, F. Keil, K. Hofmann, F. Pape, D. Konopka, G. Poll, T. Steppeler, R. Ottermann, F. Dencker, M. C. Wurz, S. Puchtler, T. Baszenski, M. Winnertz, G. Jacobs, B. Lehmann, and K. Stahl, "A review on sensor-integrating machine elements," *Advanced Sensor Research*, vol. 3, no. 4, Jan. 2024.

[2] E. Knoll, M. Ochs, A. Benfer, M. Otto, R. Brederlow, B. Vogel-Heuser, and K. Stahl, "Sensor-integrating gear wheel for in-situ measurement (SIZA)," *Forschung im Ingenieurwesen*, vol. 89, no. 1, Sep. 2025.

[3] J. Peters, L. Ott, T. Gwosch, and S. Matthiesen, "Requirements for sensor integrating machine elements: A review of wear and vibration characteristics of gears," Tech. Rep., 2020.

[4] R. Chéour, F. Derbel, O. Kanoun, and M. Abid, "Wireless sensor networks with power management for low energy consumption," in *10th International Multi-Conferences on Systems, Signals & Devices 2013 (SSD13)*, 2013.

[5] J. Braun and F. Derbel, "Wireless sensor network for fire detection with network coding to improve security and reliability," 2024.

[6] F. Derbel and F. Strakosch, "Integrated sensor based smart diagnostic and online monitoring of industrial systems," in *2022 4th International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*. IEEE, Mar. 2022.

[7] M. L. Kliemank, M. Ahmadzadeh, J. Zimmer, M. D. Somba, R. Liebich, and C. Gühmann, "Sensor-integrating gas foil bearings: real-time monitoring of temperature, lift-off state, and instantaneous angular speed," *Forschung im Ingenieurwesen*, vol. 89, no. 1, Aug. 2025.

[8] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*, I. F. Akyildiz, Ed. Wiley, Jun. 2010.

[9] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *The American Mathematical Monthly*, vol. 65, no. 1, p. 34, Jan. 1958.

[10] J. A. Beraldin and W. Steenaart, "Overflow analysis of a fixed-point implementation of the goertzel algorithm," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 322–324, 1989.

[11] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, ser. Texts in applied mathematics. New York: Springer, 2008, no. 12, includes bibliographical references and index.

[12] W. M. Gentleman, "An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients," *The Computer Journal*, vol. 12, no. 2, pp. 160–164, Feb. 1969.

[13] R. Beck, A. Dempster, and I. Kale, "Finite-precision Goertzel filters used for signal tone detection," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 7, pp. 691–700, Jul. 2001.

[14] C.-P. Chu, L.-T. Shen, and S.-H. Hwang, "A new algorithm for tone detection," *AASRI Procedia*, vol. 8, pp. 118–122, 2014.

[15] P. Xu, X. Liu, W. Liu, and M. Li, "Design of DTMF signal detection method based on improved Goertzel algorithm," in *2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC)*. IEEE, Dec. 2022, pp. 470–474.

[16] M. Zaplata, "SDR implementation for DCF77," in *2013 23rd International Conference Radioelektronika (RADIOELEKTRONIKA)*. IEEE, Apr. 2013, pp. 340–345.

[17] C. Osterloh and E. Maehle, "Low-power microcontroller-based acoustic modem for underwater robot communication," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010.

[18] A. Y. Kallel, Z. Hu, and O. Kanoun, "Comparative study of ac signal analysis methods for impedance spectroscopy implementation in embedded systems," *Applied Sciences*, vol. 12, no. 2, p. 591, Jan. 2022.

[19] M. Bocca, J. Toivola, L. M. Eriksson, J. Hollmén, and H. Koivo, "Structural health monitoring in wireless sensor networks by the embedded goertzel algorithm," in *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. IEEE, Apr. 2011, pp. 206–214.

[20] M. Kekelj, N. Bulic, and V. Sucic, "An FPGA implementation of the Goertzel algorithm in a non-destructive eddy current testing," in *2017 International Conference on Signals and Systems (ICSigSys)*. IEEE, May 2017, pp. 180–184.

[21] R. Gansel, S. Zwoch, C. Heinrich, A. Lohrengel, H. J. Maier, and S. Barton, "Identification of overloads on splined shafts by means of eddy current testing technology," *Research and Review Journal of Nondestructive Testing*, vol. 1, no. 1, Aug. 2023.

[22] P. Sundararajan, M. H. M. Sathik, F. Sasongko, C. S. Tan, J. Pou, F. Blaabjerg, and A. K. Gupta, "Condition monitoring of DC-link capacitors using Goertzel algorithm for failure precursor parameter

and temperature estimation," *IEEE Transactions on Power Electronics*, vol. 35, no. 6, pp. 6386–6396, Jun. 2020.

[23] R. Thiel, R. Fromm, F. Strakosch, J. Jäkel, and F. Derbel, "Gravity-based calibration for in-situ acceleration sensors," in *2025 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2025.

[24] R. Fromm, E. Knoll, C. Wagner, K. Stahl, B. Vogel-Heuser, and F. Derbel, "The hidden cost of performance: How microcontrollers drain batteries in sensor-integrating machine elements," submitted.

[25] Z. Kaya and E. Seke, "A novel addressing algorithm of radix-2 FFT using single-bank dual-port memory," *Circuit World*, vol. 48, no. 1, pp. 64–70, Dec. 2020.