View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by MURAL - Maynooth University Research Archive Library

Computing Modified Bessel functions with large Modulation Index for Sound Synthesis Applications

Joseph Timoney, *Thomas Lysaght*
Dept. of Computer Science
NUI Maynooth, Co. Kildare, Ireland.
jtimoney@cs.nuim.ie

Victor Lazzarini,
Dept. of Music,
NUI Maynooth, Co. Kildare, Ireland.
victor.lazzarini@nuim.ie

Ruiyao Gao,
Dept. of Electronic. Eng.,
ITT Tallaght, Dublin, Ireland.
rgao@ittdublin.ie

***Abstract*-Ordinary Bessel functions are a common function used when examining the spectral properties of frequency modulated signals, particularly in sound synthesis applications. Recently, it was shown that modified Bessel functions can also be used for sound synthesis. However, to limit the impact of aliasing distortion when using these functions, it is essential to set an upper limit on the frequency-dependent modulation index used when computing these functions. However, it can be impossible to do this beyond a certain threshold when using standard mathematical software tools such as Matlab, or the scientific toolbox of the Python language, because of numerical overflow issues. This short paper presents an approach to overcome this limitation using the MaxStar algorithm. Results are also presented to demonstrate the usefulness of this solution.**

Keywords: Modified Bessel functions, numerical overflow, Maxstar algorithm.

I. INTRODUCTION

Frequency Modulated (FM) signals are important in both the fields of telecommunications and sound synthesis. Ordinary Bessel functions are a key mathematical tool for the understanding of the spectral properties of these FM signals [1]. The success of FM synthesis as a sound generating technique led to the exploration of other techniques similar in concept, specifically using Modified Bessel functions [2]. However, for a long period this work was forgotten until recently when it was shown that a synthesis technique based on Modified Bessel functions was very useful for the generation of high quality, low-aliasing digital reproductions of the periodic waveforms used in analog subtractive synthesizers [3], for example, sawtooth waves. Specifically, the synthesis equation is

$$s(t) = e^{m\cos(\omega_m t)}\cos(\omega_c t) \qquad (1)$$

where *m* is the Modulation index.
This can be expressed using Modified Bessel functions as

$$I_0(m) + \sum_{n=1}^{\infty} I_n(m)\left(\cos(\omega_c t - n\omega_m t) + \cos(\omega_c t + n\omega_m t)\right) \qquad (2)$$

where

$I_n(.)$ is a modified Bessel function of order *n* and ω_c and ω_m are the carrier and modulation frequencies respectively [3].

From (2), it can be seen that equation (1) generates a harmonic signal with a spacing of frequency ω_m and magnitude scaling given by the set of Modified Bessel functions $I_n(m)$ where $n = 0,\ldots,\infty$.

In practical applications (1) should be scaled by a factor [3]

$$g(m) = e^{-m} \qquad (3)$$

which leads to

$$s(t) = e^{(m\cos(\omega_m t)-m)}\cos(\omega_c t) \qquad (4)$$

If the ratio between the carrier and modulation frequencies is one, then Equation (5) describes a unipolar pulse train. The width, and thus the smoothness of the pulse, is determined by the value of the modulation index *m*.

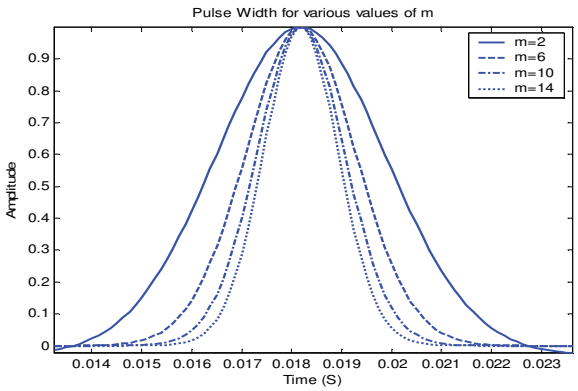

Figure 1. Plot of the pulseshape defined by (5) for various values of modulation index.

Lower values of *m* give a broader pulse shape. Figure 1 shows an example of this for values of *m* ranging from 2 to 14. For this plot the sampling rate was set to be 8 kHz and $\omega_c = \omega_m = 55$ Hz.

The spectrum of this pulse train is given by [3]

$$X(\omega)=\frac{1}{e^{m}}\sum_{n=1}^{\infty}\left(I_{n-1}(m)+I_{n+1}(m)\right)\cos(n\omega) \quad (5)$$

The harmonic amplitudes of the expression for the spectrum in (5) are determined by the modified Bessel functions $I_n(.)$ that are scaled by the factor $g(m)$. This factor gives a smooth low pass characteristic to the spectrum with the steepness of the roll-off being determined by the value of m. Figure 2 provides a spectral example of (5) for a carrier frequency of 100Hz.

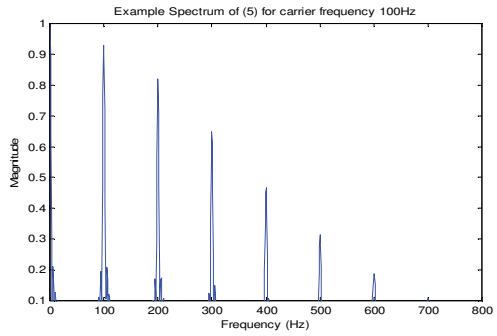

Figure 2. Plot of tan example of (5) showing the lowpass characteristic of the harmonic amplitudes.

The implications of this for the digital generation of the classic waveforms of subtractive synthesis is that by integrating this pulse train it is possible to create a signal spectrum that is an approximation to that of a sawtooth wave, whose harmonic magnitudes decrease with respect to the harmonic number. The equation for the integrated spectrum is

$$X'(\omega)=\frac{e^{-m}}{n\omega}\sum_{n=1}^{\infty}\left(I_{n-1}(m)+I_{n+1}(m)\right)\sin(n\omega) \quad (6)$$

If the integrated signal is then passed through a suitable DC blocker filter as described in [4], then the output waveshape should be close to that of a sawtooth, as illustrated in Figure 3. The example in Figure 3 was generated for carrier and modulator frequencies of 440 Hz and a sampling frequency of 44100 Hz. The modulation index was chosen to be 943 and was determined empirically. It is clear from Figure 3 that the waveshape is that of a sawtooth, validating the usefulness of this technique for the application.

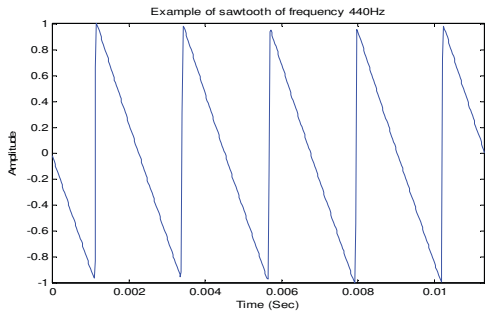

Figure 3. Sawtooth wave generated by integrating (4), followed by DC blocking, and whose spectrum is given by (6).

II. Optimising Bandlimited Signal Synthesis

What is of primary concern when creating this approximation is that the signal should be effectively bandlimited, that is, any aliased components should be of sufficiently low magnitude so as to be imperceptible. To ensure this, it is then a question of choosing a suitable value for m such that any harmonics that exist in theory beyond half the sampling frequency are sufficiently small such that their aliased version will not be heard. This can be posed as an optimsation problem, as given by ~~in~~ (7). Here, a figure of -90dB is chosen as the upper threshold on the spectral magnitude of the aliased components [3].

$$\max_{m}\left\{20\log_{10}\frac{I'_{N+1}(m)(N+1)^{-1}}{I'_{1}(m)}\right\}\leq -90 \quad (7)$$

where N is the number of harmonics in the sawtooth wave from DC to half the sampling frequency and

$$I'_{n}(m)=I_{n-1}(m)+I_{n+1}(m) \quad (8)$$

To perform this optimization it is possible to use a standard routine such as '`fmin`', for example, that is available as a routine in the Matlab software package. This routine uses a Nelder-Mead Simplex search method [5]. However, in the implementation of the optimization of (7) a problem was discovered. Specifically, when attempting to compute the magnitude of the Modified Bessel function for values of modulation index greater than 700 the algorithm used generates a numerical overflow and will return a value of infinity. A similar behaviour was observed when using the `Sci.py` module of python for the computation in [6].

A. Computing Modified Bessel Functions using Logs

To compute a Modified Bessel function of order n and modulation index m it is possible to use the formula [7]

$$I(n,m)=\sum_{k=0}^{\infty}\frac{(m/2)^{n+2k}}{k!(n+k)!} \quad (9)$$

From (9) both the numerator and denominator of the summation will grow to be infinitely large, but empirical observation found that the value of their ratio will first reach a maximum and then decrease to zero as *m* increases. In an implementation it can be surmised that the maximum number of terms in the summation can be restricted as long as it exceeds the point where the ratio reaches zero. This is a valid approach but fails when the maximum of the ratio is beyond the numerical precision of the machine. As stated already, this occurs for large values of modulation index and thus an alternative is required in such cases. An obvious choice for compressing the numerical values generated by each term of (9) is to use the logarithmic function

$$\log\left(\frac{(m/2)^{n+2k}}{k!(n+k)!}\right)=(n+2k)\log(m/2)-\log(k!(n+k)!) \tag{10}$$

and using the logarithmic property

$$\log x^{z} = z\log x \tag{11}$$

Equation (11) can then be rewritten using the multiplicative property given in (12)

$$\log(xz)=\log x+\log z \tag{12}$$

to give

$$\log\left(\frac{(m/2)^{n+2k}}{k!(n+k)!}\right)=(n+2k)\log(m/2)-\log(k!) -\log((n+k)!) \tag{13}$$

A number of possibilities exist for expressing the logarithm of a factorial. Firstly, the exact expression is [8].

$$\log(x!)=\sum_{z=1}^{x}\log z \tag{14}$$

Alternatively, a very good approximation for the logarithm of a factorial due to Ramaujan [8], for $x \neq 0$, can be written which would reduce the computational effort in evaluating the multiplications in $\log(x!)$ for each term.

$$\log(x!)\approx x\log x - x+\frac{\log(x(1+4x(1+2x)))}{6}+\frac{\log(\pi)}{2} \tag{15}$$

Furthermore, it also is more robust numerically. Particularly, in the case of Matlab [5], if the number of terms *k* selected exceeds 170, then $\log(k!)=\infty$.

Using (15), equation (13) can be rewritten to produce

$$\begin{aligned}&=(n+2k)\log(m/2)-k\log k+k-\frac{\log(k(1+4k(1+2k)))}{6}\\&-(n+k)\log(n+k)+(n+k)\\&-\frac{\log((n+k)(1+4(n+k)(1+2(n+k))))}{6}+\log\left(\frac{\pi}{2}\right)\end{aligned} \tag{16}$$

B. Applying the MaxStar algorithm

From (16) it can be seen that each term in (10) can be computed using logarithms which will significantly reduce the size of its numerical value, thus avoiding overflow problems. However, the next issue is how to add them without computing the exponential of each term. They should also be added in the log domain and then the exponential found of the overall result. To this end a very useful algorithm from the field of telecommunications is the MaxStar algorithm [9]

$$\ln\left(e^{x}+e^{z}\right)=\left\{\max(x,z)+\ln\left(1+e^{-|x-z|}\right)\right\} \tag{17}$$

where

$$\begin{aligned}\max(x,z)&=x \quad if\ x\geq z\\&=z \quad otherwise\end{aligned}$$

This expression is applied iteratively to the summation until the final term is reached [10]. An alternative approximation was recently given in [10], which was

$$\ln\left(e^{x}+e^{z}\right)=\frac{x+z}{2}+\ln\left(2\cosh\left(\frac{x-z}{2}\right)\right) \tag{18}$$

Once the sum of terms is found using (17) or (18) all that remains to compute (9) is to find the exponential of this sum. This procedure will work in Matlab [5] as long as the total value of the log of the sum of the terms in (9) is 709 or less as otherwise an infinite output will result because in this package

$$e^{710}=\infty \tag{19}$$

This could be problematic in the general case, but if all we want to do is solve the optimization problem of (7) then the logarithmic can also be rewritten as

$$\log_{10}\left(I'_{N+1}(m)(N+1)^{-1}\right)-\log_{10}(I'_{1}(m))$$

$$= \log_{10}(N+1)^{-1} + \log_{10}(I'_{N+1}(m)) - \log_{10}(I'_1(m)) \tag{20}$$

Ignoring the first term which is a constant and substituting (8) into (20) gives

$$= \log_{10}(I_N(m) + I_{N+2}(m)) - \log_{10}(I_0(m) + I_2(m)) \tag{21}$$

Both terms in (21) are structurally similar, so, for example, considering the first term only and defining the output of the MaxStar algorithm as $MS(.)$ leads to the definition

$$MS(I_n(m)) = \log\left(\sum_{k=0}^{\infty} \frac{(m/2)^{n+2k}}{k!(n+k)!}\right) \tag{22}$$

Then

$$\log_{10}(I_N(m) + I_{N+2}(m)) = \log_{10}(\exp(MS(I_N(m))) + \exp(MS(I_{N+2}(m)))) \tag{23}$$

where for clarity the function exp(x) is used to represent e^x. Applying the MaxStar algorithm to (23) produces the nested expression in (24)

$$= \log_{10}(\exp(MS(MS(I_N(m)) + MS(I_{N+2}(m))))) \tag{24}$$

It is possible then to apply the property

$$\log_y(x^z) = z \log_y x \tag{25}$$

which results in

$$= MS(MS(I_N(m)) + MS(I_{N+2}(m)))\log_{10} e \tag{26}$$

Thus, the exponential power has been removed and now is present in (26) in the form of a multiplication by a constant. In this form any issues with numerical overflow should be overcome. The optimization in (7) can finally be rewritten using the formulation in (26) as

$$\max_m \left\{ \begin{array}{l} \log_{10}(N+1)^{-1} + \\ MS(MS(I_N(m)) + MS(I_{N+2}(m)))\log_{10} e \\ -MS(MS(I_0(m)) + MS(I_2(m)))\log_{10} e \end{array} \right\} \le -\frac{90}{20} \tag{27}$$

For example, if we have a pitch frequency for the sawtooth wave of 146.8324 Hz (note D) and a sampling rate of 44100 Hz, the number of harmonics that will exist to half the sampling frequency is N=150. Setting the maximum number of terms in the summation to be 2000, the optimization routine returns a value of modulation index m=2131.7. Figure 4 is a plot of the lower portion of the spectrum of this sawtooth, after hanning windowing, showing its harmonics with no visible alias components present.

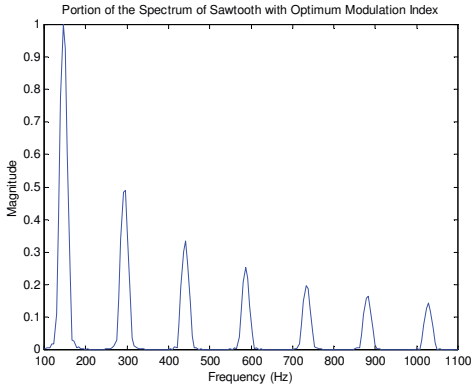

Figure 4. Lower part of spectrum of optimized sawtooth.

III. Conclusion

This short paper has presented an approach for the computation of Modified Bessel functions with high modulation indices that uses the MaxStar algorithm to overcome numerical difficulties currently experienced with mathematical software packages. Furthermore, it also has shown how an optimisation formulation can be rewritten using the MaxStar algorithm that obviates the need to explicitly compute large numbers that are raised to an exponential power.Future work will seek out other similar applications where the MaxStar algorithm would prove to be useful.

IV. Acknowledgment

Victor Lazzarini would like to acknowledge the funding support given by An Foras Feasa for this work.

References

[1] J. Chowning, and D. Bristow, *FM Theory & Applications - By musicians for musicians*, Yamaha, Tokyo, 1986.

[2] J.A. Moorer, ,"The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulas". *Journal of the Audio Engineering Society*, 24 (9), 1976.

[3] V. Lazzarini, J. Timoney and T. Lysaght, "A Modified FM synthesis approach to bandlimited signal generation", *Proc. 11th conf. Digital Audio Effects (DAFx)*, Espoo, Finland, Sept. 1-4, 2008.

[4] R. Yates and R. Lyons, 'DSP Tips & Tricks [DC Blocker Algorithms]', *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 132 – 134, March 2008.

[5] *Matlab 5.3*, The Mathworks: 1999.

[6] *Sci.py*, Python Module: 2009. http://www.scipy.org/

[7] G. N. Watson, *A Treatise on the Theory of Bessel Functions*, 2nd Ed., Cambridge Univ. Press, Cambridge, UK, 1944.

[8] *Factorial*, Wikipedia entry, 2009: http://en.wikipedia.org/wiki/Factorial.

[9] J. A. Erfanian, and S. Pasupathy, "Low-Complexity Parallel-Structure Symbol-by-Symbol Detection for IS1 Channels," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, June 1st - 2nd, 1989.

[10] Motorola Inc, Apparatus and method for calculating the logarithm of a sum of exponentials, *European Patent Office*, EP20020293271, June 2004.