IMA Journal of Numerical Analysis (2017) **37**, 1355–1374
doi: 10.1093/imanum/drw038
Advance Access publication on August 3, 2016

On the numerical stability of the second barycentric formula for trigonometric interpolation in shifted equispaced points

Anthony P. Austin* and Kuan Xu
Mathematical Institute, University of Oxford, Woodstock Road, Oxford OX2 6GG, UK
*Corresponding author: austin@maths.ox.ac.uk kuan.xu@maths.ox.ac.uk

[Received on 20 May 2015; revised on 16 March 2016]

We consider the numerical stability of the second barycentric formula for evaluation at points in $[0, 2\pi]$ of trigonometric interpolants in an odd number of equispaced points in that interval. We show that, contrary to the prevailing view, which claims that this formula is always stable, it actually possesses a subtle instability that seems not to have been noticed before. This instability can be corrected by modifying the formula. We establish the forward stability of the resulting algorithm by using techniques that mimic those employed previously by Higham (2004, The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, **24**, 547–556) to analyse the second barycentric formula for polynomial interpolation. We show how these results can be extended to interpolation on other intervals of length-2π in many cases. Finally, we investigate the formula for an even number of points and show that, in addition to the instability that affects the odd-length formula, it possesses another instability that is more difficult to correct.

Keywords: trigonometric interpolation; Lagrange interpolation; barycentric formula; rounding error analysis; forward error; numerical stability.

1. Introduction

Let $K \geq 1$ be an odd integer, and let X be a set of K equispaced points $x_k = (k + \alpha)h$, $0 \leq k \leq K - 1$, in $[0, 2\pi]$, where $h = 2\pi/K$ is the grid spacing and $\alpha \in [0, 1]$ is a parameter that determines the grid shift (i.e., the deviation of x_0 from 0). Let $f_0, \ldots, f_{K-1}$ be arbitrary real numbers, which we take as elements of a vector f. This article begins by considering the formulae

$$t_{f,X}(x) = \frac{1}{K} \sin\left(\frac{K(x - \alpha h)}{2}\right) \sum_{k=0}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x - x_k}{2}\right)} f_k \tag{1.1}$$

and

$$t_{f,X}(x) = \frac{\sum_{k=0}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)} f_k}{\sum_{k=0}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)}} \tag{1.2}$$

for evaluating at a point $x \in [0, 2\pi]$ the unique trigonometric polynomial $t_{f,X}$ of degree $N = (K - 1)/2$ that interpolates the value f_k at the point x_k for each k. We refer to (1.1) and (1.2) as the *first* and *second barycentric formulae* for the trigonometric interpolant, respectively. We are concerned in particular with

© The authors 2016. Published by Oxford University Press on behalf of the Institute of Mathematics and its Applications. All rights reserved.

the numerical stability of the latter. The authors became interested in this subject as a result of discussions with colleagues working on extending the Chebfun software package to operate with periodic functions Wright *et al.* (2015).

The first formula (1.1) can be seen as a rewriting of the classical trigonometric interpolation formula of Gauss (1866, p. 281) when the interpolation points are equispaced. For $\alpha = 0$, it appears (along with its counterpart for an even number of interpolation points) in the work of de la Vallée Poussin (1908) and Henrici (1979). The latter gives a derivation of it in that case based on a complex change-of-variable and the relationship between trigonometric interpolation in equispaced points in an interval and interpolation in equispaced points on the unit circle using Laurent polynomials. The formula for $\alpha \neq 0$ then follows from the fact that evaluating at x the interpolant to given data on a grid with $\alpha \neq 0$ is the same as evaluating at $x - \alpha h$ the interpolant to the same data on the grid with $\alpha = 0$. The second formula (1.2) can be derived from (1.1) by observing that the latter implies

$$1 = \frac{1}{K} \sin\left(\frac{K(x-\alpha h)}{2}\right) \sum_{k=0}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)}$$

and dividing (1.1) through by this identity on both sides.

The second formula seems to have been first introduced by Salzer, who gave versions of it applicable to trigonometric interpolation in both an odd number Salzer (1948) and an even number Salzer (1960) of arbitrary points. The simplified forms that his formulae take when the points are equispaced, including (1.2), were first written down by Henrici (1979). Berrut (1984a) later provided special variants of the second formula for cases in which the interpolation data f_k possess odd or even symmetry. The primary advantage of these formulae is that they offer a way to evaluate the interpolant in just $O(nK)$ operations, where n is the number of evaluation points, as opposed to the $O(K\log K + nK)$ operations that would be required by methods based on the fast Fourier transform. These formulae are direct analogues of the more widely known barycentric formulae for polynomial interpolation that have been made popular in recent years by Berrut & Trefethen (2004).

The numerical stability of these formulae has been discussed in a few places in the literature. Henrici (1979) notes that (1.1) suffers from instability as K grows due to our inability to evaluate the factor $\sin\big(K(x-\alpha h)/2\big)$ in front of the sum to high relative accuracy for large K. Even for small K, both Henrici (1979) and Berrut (1984a) indicate that this factor causes instability when evaluating (1.1) for x close to one of the interpolation points x_k. This behavior contrasts markedly with that of the first barycentric formula for polynomial interpolation, which is backward stable even for nonoptimal interpolation grids Higham (2004).

Thus, it is necessary to use (1.2), which does not contain this factor and so cannot suffer from these issues.[1] Nevertheless, the careful numerical analyst may hesitate to assess (1.2) as stable due to the singularities at the points x_k present in the numerator and denominator. To paraphrase Henrici (1979), if x is close to x_k, cancellation errors that occur when $x - x_k$ is calculated in floating-point arithmetic may be magnified into large absolute errors in $1/\sin\big((x-x_k)/2\big)$. Typically, however, this does not seem to cause trouble, and both Henrici (1979) and Berrut (1984b) provide numerical examples illustrating the apparent stability of (1.2).

[1] Alternatively, as pointed out by an anonymous referee, one can stabilize (1.1) for evaluations near interpolation points by adapting a technique of Gautschi (2001) used to stabilize the analogous formula for sinc interpolants, but this has its own disadvantages. Moreover, the resulting algorithm still requires correction for the instabilities discussed in this article.

Henrici (1979) gives an informal argument explaining these observations. The idea is that while an error is made, the error is the *same* in both the numerator and denominator, and thus 'cancels out' in the quotient. This reasoning is equally applicable to the polynomial analogue of (1.2), and Higham (2004) has given precise arguments that justify it in that context. Working in the setting of polynomial interpolation in arbitrary points, he showed that the second polynomial formula is forward stable unless the Lebesgue constant for the interpolation grid is large. It is natural to expect that something similar holds in the trigonometric case and, in particular, that (1.2) is forward stable, since the Lebesgue constant for trigonometric interpolation in equispaced points (which are the optimal points) is modest.

It therefore comes as a surprise, at least to the authors, that this is not quite true. While (1.2) produces good results in the majority of cases, it does, in fact, possess an instability that seems to have been overlooked in the investigations of Henrici (1979) and Berrut (1984a,b). We illustrate and explain the origin of this instability in Section 2. Fortunately, it is possible to correct the instability via a rewriting of (1.2), as we show in Section 3. Combining the original and rewritten formulae, we obtain an algorithm that is forward stable, and we prove this rigorously in Section 4 by adapting the analysis of Higham (2004) to our setting. Finally, in Sections 5 and 6 we discuss interpolation on intervals other than $[0, 2\pi]$ and make a few remarks on what happens when K is even instead of odd.

2. Instability of the second formula

We can demonstrate the instability in (1.2) with a simple numerical example. Take $\alpha = 1$, $K = 3$ and $f_k = \sin(x_k)$ for each k. We evaluate (1.2) with these parameters at several points x whose distances from 0 range from 1 to 10^{-15}. We perform the evaluation twice: once in double precision and once in 256-bit (approximately 75-digit) precision using the arbitrary precision arithmetic features of the Julia programming language Bezanson *et al.* (2012), which are based on the GNU MPFR library Fousse *et al.* (2007). We take the high precision results as 'exact' and use them to measure the relative error in the results obtained in double precision.

The results are displayed in Fig. 1. The error increases steadily as the evaluation point x moves closer to 0. On the other hand, the product of the condition number $\kappa(x, X, f)$ for evaluating $t_{f,X}(x)$ (see Section 4.1) and the unit roundoff $u = 2^{-52}$ is at the level of u for all evaluation points x considered. We conclude that (1.2) is indeed unstable under these circumstances.

After a little thought, the origin of the instability can be identified. For our choice of α, $x_{K-1} = 2\pi$, so when x is near 0, we evaluate the sine function at a point close to π when computing the terms at $k = K - 1$ in the numerator and denominator of (1.2). The sine function is poorly conditioned near π, so the rounding errors incurred when forming $(x - x_{K-1})/2$ get magnified into large relative errors in the computed value of $\sin\big((x - x_{K-1})/2\big)$.

For many uses of (1.2), these errors do not cause any problems, since they 'cancel out' in the final quotient as described in Section 1. The mechanism driving the cancellation in this case is the dominance of the $k = K - 1$ terms in the numerator and denominator of (1.2): for α near 1 and x near 0, these terms will typically be much larger than the terms for $k < K - 1$, since $\sin\big((x - x_{K-1})/2\big)$ is nearly 0. Hence, any relative error in the $k = K - 1$ terms, even a large one, will divide out neatly when taking the quotient. In our example, however, f_{K-1}, which has a magnitude on the order of 10^{-16}, is much smaller than f_k for $k < K - 1$, all of which have magnitudes on the order of 1. This poor scaling of the function values relative to each other offsets the dominance of the $k = K - 1$ terms, resulting in imperfect cancellation.

Something interesting occurs if we repeat the experiment, but take $\alpha = 0$ instead of $\alpha = 1$. In this case, we anticipate instability when evaluating at points near 2π instead of 0, with the problematic terms occurring at $k = 0$ instead of $k = K - 1$. The results are displayed in Fig. 2. While the plot of the product

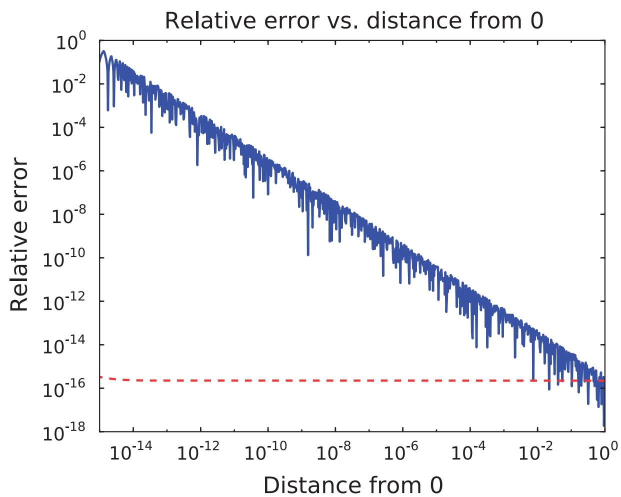

FIG. 1. Illustration of instability in (1.2). The solid line depicts the relative error in the evaluations for the example described in Section 2. The dashed line shows the product of the condition number of the evaluations (computed using the formula given by Lemma 4.1 below) and the unit roundoff $u = 2^{-52}$. As the distance between the evaluation point and 0 decreases, the relative error rises even though the evaluation remains well conditioned, indicating numerical instability.

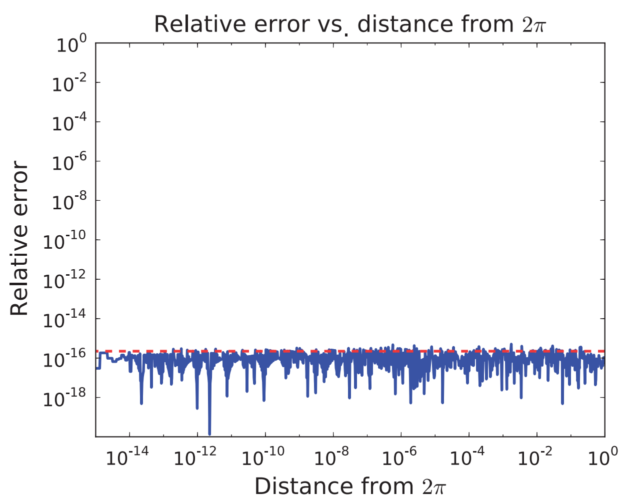

FIG. 2. Same as Fig. 1, but with $\alpha = 0$ instead of $\alpha = 1$ and with the evaluation points located near 2π instead of 0. The relative error is at the level of machine precision due to the special circumstances enjoyed by this case.

of the condition number and the unit roundoff is unaltered, surprisingly, the relative error is at the level of machine precision for all evaluation points. The reason this happens is that when $\alpha = 0$, $x_0 = 0$. Thus, $(x - x_0)/2$ is evaluated *exactly* for all $x \in [0, 2\pi]$, since subtraction of 0 and division by 2 incur no errors in standard IEEE floating-point arithmetic. There is therefore no rounding error made whose effect can be amplified by the ill conditioning of the sine function, so the instability cannot be excited in this special but very common case. If α is taken to be only near 0 (say, 10^{-15}) instead of exactly 0, the instability appears as expected.

We speculate that this behavior is one of the reasons the instability described in this section has evaded notice thus far in the literature, as the $\alpha = 0$ grid is perhaps the most frequently employed grid of equispaced points in $[0, 2\pi]$; indeed, Henrici (1979) works exclusively with this grid. The authors of this article only began to notice the instability when considering the $\alpha = 1$ grid and when working with the analogue of the $\alpha = 0$ grid on $[-\pi, \pi]$, for which the first point is $-\pi$, a number which is undistinguished in floating-point arithmetic.

3. A stable algorithm

The instability just described only arises when the interpolation data are poorly scaled in the sense that either f_0 or f_{K-1} is much smaller than the other f_k when α is near 0 or 1, respectively. If this is not the case, i.e., if $\max_{1 \le k \le K-1} |f_k/f_0|$ (for α near 0) or $\max_{0 \le k \le K-2} |f_k/f_{K-1}|$ (for α near 1) is not too large,[2] then (1.2) will be stable for all evaluation points in $[0, 2\pi]$. Interpolation data that are as wildly poorly scaled as those of the examples shown in the previous section are relatively uncommon in practice. Even when the data are poorly scaled, most evaluations of the trigonometric interpolant are done in the interior of the interval, where the sine evaluations are well conditioned, and (1.2) will be stable in this case as well. Thus, (1.2) is stable in most cases of practical interest.

Nevertheless, knowing how to fix the instability is valuable so that it can be done when needed. This can be accomplished by rewriting (1.2) to avoid evaluating the expression $\sin\big((x - x_k)/2\big)$ near points where it is poorly conditioned. There are two situations in which a bad evaluation can occur: when α is near 0 and the evaluation point x is near 2π and when α is near 1 and x is near 0.

The remedy we propose is to use periodicity to adjust the location of the interpolation point furthest from x in these cases so that the distance between it and x can never get too close to 2π. Consider the case where α is near 0. For x near 2π, the interpolation point furthest from x is x_0, so we modify (1.2) by replacing x_0 by its periodic image $x_0 + 2\pi$ and changing the signs of the $k = 0$ terms in both sums. The resulting formula, which amounts to using (1.2) to compute an interpolant in the points $x_1, \ldots, x_{K-1}, x_0 + 2\pi$ instead of $x_0, x_1, \ldots, x_{K-1}$, is exactly equal to (1.2) mathematically, but not in floating-point arithmetic. Similar comments apply to the case where α is near 1 and x is near 0, for which we replace x_{K-1} by $x_{K-1} - 2\pi$ and change the signs of the $k = K - 1$ terms. For explicit formulae, see (3.1) and (3.2), below.

We are not done yet, however, as all we have actually done is rewrite the poorly conditioned terms in (1.2) in a different way. The modified terms are still poorly conditioned, as a problem's conditioning is independent of how it is written down or represented. What has changed is the *source* of the poor conditioning. Instead of through the sine function itself, it now enters via the potential for cancellation error in the computation of the argument to the sine function. The second key idea needed to stabilize (1.2) is the realization that we can avoid these problems by computing the argument in a particular way, as we now describe.

First, we must group the terms of the argument appropriately. Consider the case where α is near 0, so that the argument to the sine function in the modified term is $(x - x_0 - 2\pi)/2$. Ignoring the division by 2, which has no potential for cancellation, if we evaluate the rest in floating-point from left to right

[2] As a rule of thumb, one can expect to lose roughly one digit of accuracy in evaluations near the 'bad' endpoint for each order of magnitude in these quantities. For instance, if α is near 0 and $\max_{1 \le k \le K-1} |f_k/f_0|$ is on the order of 10^8, then a loss of about 8 digits in evaluations near 2π would be typical.

as $(x - x_0) - \mathrm{fl}(2\pi)$, where $\mathrm{fl}(2\pi)$ is the nearest floating-point number to 2π (see Section 4.2), then the second subtraction will involve two nearby quantities whenever x is near 2π and x_0 is near 0. Even if the second subtraction is performed without rounding error, accuracy will be lost if the magnitude of the rounding error made in the first subtraction is significant compared to the magnitude of the final result.

We can fix this by grouping the terms as $\big(x - \mathrm{fl}(2\pi)\big) - x_0$ instead. While the subtraction $x - \mathrm{fl}(2\pi)$ still incurs cancellation, it is of a benign sort, as neither x nor $\mathrm{fl}(2\pi)$ has been contaminated by rounding errors from previous computations (but see the next paragraph). Moreover, since $x \leq \mathrm{fl}(2\pi)$ and $x_0 \geq 0$, the second subtraction involves two quantities of opposite sign, and hence no further cancellation can occur. The final result will therefore be a high relative accuracy approximation to the exact value (i.e., computed without rounding error) of $x - x_0 - \mathrm{fl}(2\pi)$.

This is almost what we want, but not quite: we really want a high relative accuracy approximation to $x - x_0 - 2\pi$. Evaluating $\big(x - \mathrm{fl}(2\pi)\big) - x_0$ in floating-point will not generally deliver this because of the rounding error in the approximation $\mathrm{fl}(2\pi) \approx 2\pi$. While the cancellation in $x - \mathrm{fl}(2\pi)$ is benign when this subtraction is viewed simply as a difference between two floating-point numbers, it is catastrophic from the perspective of computing an approximation to $x - 2\pi$.

The fix for this is to subtract off an additional correction term to compensate for the error in the approximation $\mathrm{fl}(2\pi) \approx 2\pi$. More precisely, let c be the nearest floating-point number to $2\pi - \mathrm{fl}(2\pi)$. Then, in exact arithmetic, $\mathrm{fl}(2\pi)+c$ is an approximation to 2π with relative error on the order of the square of the unit roundoff (see Section 4.2). We cannot form $\mathrm{fl}(2\pi) + c$ directly in floating-point arithmetic because c is insignificant compared to $\mathrm{fl}(2\pi)$ and would be rounded off; however, if adding or subtracting $\mathrm{fl}(2\pi)$ to or from something results in a quantity small enough that c is significant when compared with it, we can expect to obtain a higher accuracy result if we subsequently add or subtract c as appropriate.

In this discussion, we have considered only the case where α is close to 0 for definiteness; similar remarks apply to the modified version of (1.2) for α close to 1. Rigorous justification for all of these statements will be given in the analysis of Section 4. The value c can be easily computed using any software package that supports arbitrary precision arithmetic or even by hand with aid of a table that lists the value of π to many places. For IEEE floating-point arithmetic, $c = 2.4492935982947064 \times 10^{-16}$ in double precision and $c = -1.7484555 \times 10^{-7}$ in single precision.

The only remaining matter is to decide precisely when to use the modified formulae instead of (1.2), i.e., to give a criterion for determining when x is 'too close' to 0 or 2π. For reasons that we will justify in Section 4, we switch to the modified formulae whenever x is within $\pi|1-2\alpha|/K$ of the relevant endpoint. Note that for $\alpha = 1/2$, this quantity is zero, so (1.2) is used without modification for all $x \in [0, 2\pi]$.

To summarize, the exact procedure we propose is the following:

- If $\alpha \in [0, 1/2)$, use (1.2) for $x \in [0, 2\pi - \pi(1 - 2\alpha)/K]$. Otherwise, use

$$t_{f,X}(x) = \frac{\sum_{k=1}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)} f_k - \frac{1}{\sin\left(\frac{x-x_0-2\pi}{2}\right)} f_0}{\sum_{k=1}^{K-1} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)} - \frac{1}{\sin\left(\frac{x-x_0-2\pi}{2}\right)}}, \tag{3.1}$$

  with $x - x_0 - 2\pi$ computed as $\big(\big(x - \mathrm{fl}(2\pi)\big) - c\big) - x_0$.
- If $\alpha = 1/2$, use (1.2) for all $x \in [0, 2\pi]$.

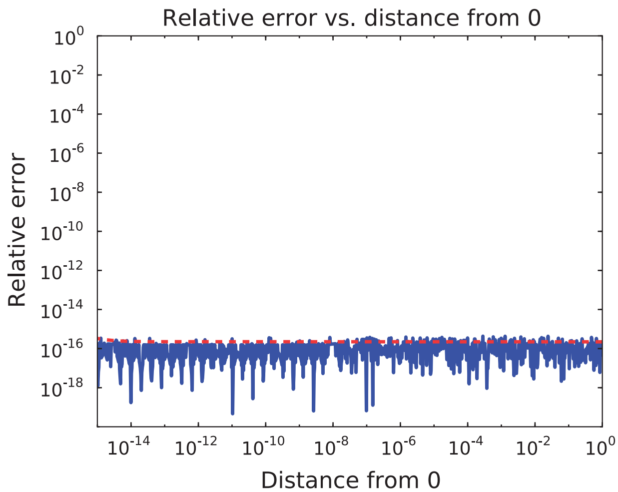

FIG. 3. Same as Fig. 1, but using the formula (3.2) (with $x - x_0 - 2\pi$ computed as prescribed in Section 3) instead of (1.2) to do the evaluations. All errors are now at the level of machine precision.

- If $\alpha \in (1/2, 1]$, use (1.2) for $x \in [\pi(2\alpha - 1)/K, 2\pi]$. Otherwise, use

$$t_{f,X}(x) = \frac{\sum_{k=0}^{K-2} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)} f_k - \frac{1}{\sin\left(\frac{x-x_{K-1}+2\pi}{2}\right)} f_{K-1}}{\sum_{k=0}^{K-2} \frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)} - \frac{1}{\sin\left(\frac{x-x_{K-1}+2\pi}{2}\right)}}, \tag{3.2}$$

with $x - x_{K-1} + 2\pi$ computed as $x - \big((x_{K-1} - \mathrm{fl}(2\pi)) - c\big)$.

This scheme has the disadvantage that an implementation must make decisions based on the location of the evaluation point. It is therefore less computationally efficient compared with (1.2) as-is; however, this is the price that must be paid to guarantee stability for all evaluation points x and all possible interpolation data f_k.

To verify that the scheme we have described works, we repeat our experiment from Section 2 with $\alpha = 1$ using this algorithm. All of the evaluation points x considered lie in $[0, \pi(2\alpha - 1)/K)$, so we use (3.2) for all of them. The relative error, depicted in Fig. 3, is now at the level of machine precision, even for evaluation points that are very close to 0.

4. Analysis of the proposed algorithm

We complete our investigation by putting the observed stability of the algorithm given in the previous section on a rigorous basis with a formal proof. The notation and framework we use for our analysis are borrowed directly from Higham (2004). To keep this article self-contained, we repeat the relevant definitions here.

4.1 *Condition number*

Our bounds will be stated in terms of the following condition number (inequalities between vectors are understood to hold componentwise):

DEFINITION 4.1 For $t_{f,X}(x) \neq 0$, the *relative condition number* of $t_{f,X}$ at x with respect to perturbations in f is

$$\kappa(x,X,f) = \lim_{\varepsilon\to 0} \sup\left\{\left|\frac{t_{f,X}(x) - t_{f+\Delta f,X}(x)}{\varepsilon t_{f,X}(x)}\right| \;:\; |\Delta f| \leq \varepsilon|f|\right\}.$$

A trivial rearranging of (1.1) yields the following 'Lagrange' form for $t_{f,X}(x)$:

$$t_{f,X}(x) = \sum_{k=0}^{K-1} \ell_k(x) f_k, \qquad \ell_k(x) = \frac{(-1)^k}{K}\frac{\sin\left(\frac{K(x-\alpha h)}{2}\right)}{\sin\left(\frac{x-x_k}{2}\right)}.$$

The following lemma gives an explicit expression for $\kappa(x,X,f)$ and shows how it can be used to bound the relative difference between $t_{f,X}(x)$ and $t_{f+\Delta f,X}(x)$ for a given perturbation Δf. It directly parallels Lemma 2.2 of Higham (2004) and can be proved identically.

LEMMA 4.1 We have

$$\kappa(x,X,f) = \frac{\sum_{k=0}^{K-1} |\ell_k(x) f_k|}{|t_{f,X}(x)|} \geq 1$$

and for any vector Δf with $|\Delta f| \leq \varepsilon|f|$,

$$\left|\frac{t_{f,X}(x) - t_{f+\Delta f,X}(x)}{t_{f,X}(x)}\right| \leq \varepsilon\kappa(x,X,f).$$

4.2 *Floating-point model*

We denote floating-point approximations to quantities by $\mathrm{fl}(\cdot)$. The standard model of floating-point arithmetic (Higham, 2002, Chapter 2) posits that whenever x and y are floating-point numbers and $\circledast$ is one of the four basic arithmetic operations $+$, $-$, $\times$ or $\div$, we have

$$\mathrm{fl}(x \circledast y) = (x \circledast y)(1+\delta)^{\pm 1} \qquad |\delta| \leq u, \tag{4.1}$$

where u is the unit roundoff. We use this model with one modification: we assume additionally that whenever x is a floating-point number

$$\mathrm{fl}\big(\sin(x)\big) = \sin(x)(1+\delta)^{\pm 1} \qquad |\delta| \leq u. \tag{4.2}$$

This assumption is not guaranteed to hold by any floating-point standard; however, it is possible to accomplish this and, similarly, for the other common transcendental functions with high-quality implementations

Muller *et al.* (2010). Moreover, the latest revision of the IEEE floating-point standard recommends (but does not mandate) that languages supporting floating-point operations also provide correctly rounded implementations for all such basic functions[3] IEEE (2008). This suggests that our additional assumption is, at the very least, reasonable. In fact, we will not require its full force: for our purposes, it is sufficient for it to hold when $x \in [-\pi, \pi]$.

The symbol $\langle n \rangle$ denotes the accumulation of n relative errors accrued during a floating-point computation:

$$\langle n \rangle = \prod_{i=1}^{n} (1 + \delta_i)^{\rho_i} \qquad \rho_i = \pm 1, \qquad |\delta_i| \leq u.$$

When necessary, we write $\langle n \rangle_k$ to indicate that the relative errors in the product depend on an index k.

Throughout our analysis, we will at times need to assume that $nu \leq 1$, where n is a small positive integer. These assumptions will hold for any floating-point system that is used in practice.

4.3 *Stability analysis*

We are now ready to carry out our analysis. For the remainder of this section, x is taken to be a fixed value in $[0, 2\pi]$, and we assume that x, x_k and f_k are all floating-point numbers.[4] We ignore all issues of overflow and underflow. Our goal is to obtain a bound on the relative error $|t_{f,X}(x) - \widehat{t}_{f,X}(x)|/|t_{f,X}(x)|$, where $\widehat{t}_{f,X}(x)$ is the approximation to $t_{f,X}(x)$ obtained by evaluating (1.2), (3.1) or (3.2) in floating-point arithmetic as prescribed in Section 3. Specifically, we will prove the following theorem:

THEOREM 4.2 The relative error in evaluating $t_{f,X}(x)$ in floating-point arithmetic using the algorithm of Section 3 satisfies

$$\left| \frac{t_{f,X}(x) - \widehat{t}_{f,X}(x)}{t_{f,X}(x)} \right| \leq (5K + 7)u\kappa(x, X, f) + (5K + 6) \left(\frac{2}{\pi} \log(K) + 2 \right) u + O(u^2) \tag{4.3}$$

for all $\alpha \in [0, 1]$.

Thus, the procedure outlined in Section 3 gives a forward stable method for evaluating trigonometric interpolants in equispaced points.

Proof. We will establish the bound for $\alpha \in [0, 1/2]$; the argument for $\alpha \in (1/2, 1]$ is similar. Our argument is identical in structure to the one given by Higham (2004) for the polynomial case.

---

[3] For instance, the implementation for sine contributed by IBM to glibc (v. 2.21 at the time of this writing) claims to do this.

[4] Of course, it is not possible that the x_k are simultaneously exactly equispaced in $[0, 2\pi]$ and also floating-point numbers. With approximately equispaced x_k, the formulae (1.2), (3.1) and (3.2) only approximate the trigonometric interpolant instead of computing it exactly. This does not matter for our investigation, however, as we are only concerned with the numerical stability of these formulae. Mascarenhas & de Camargo (2017) give an analysis of the effects of rounding errors in the interpolation points in the polynomial case.

First, we develop an expression for $\widehat{t_{f,X}}(x)$ in the case where (1.2) is used for the evaluation. By (4.1), we have, for some $\delta_{k,1}$ and $\delta_{k,2}$ with $|\delta_{k,1}| \leq u$ and $|\delta_{k,2}| \leq u$,

$$\mathrm{fl}\left(\frac{x - x_k}{2}\right) = \frac{x - x_k}{2}(1 + \delta_{k,1})(1 + \delta_{k,2}). \tag{4.4}$$

Hence, by (4.2) and the fact that $\sin\big(x(1+\varepsilon)\big) = \sin(x)\big(1 + \varepsilon x \cot(x) + O(\varepsilon^2)\big)$ for small ε, we have

$$\mathrm{fl}\left(\sin\left(\frac{x - x_k}{2}\right)\right) = \sin\left(\frac{x - x_k}{2}(1 + \delta_{k,1})(1 + \delta_{k,2})\right)\langle 1\rangle_k = \sin\left(\frac{x - x_k}{2}\right)\big(1 + \eta_k + O(u^2)\big)\langle 1\rangle_k,$$

where

$$\eta_k = (\delta_{k,1} + \delta_{k,2})\frac{x - x_k}{2}\cot\left(\frac{x - x_k}{2}\right).$$

Therefore, our floating-point approximation to the numerator of (1.2) is given by

$$\begin{aligned}
\mathrm{fl}\left(\sum_{k=0}^{K-1}\frac{(-1)^k f_k}{\sin\left(\frac{x-x_k}{2}\right)}\right) &= \sum_{k=0}^{K-1}\frac{(-1)^k f_k}{\sin\left(\frac{x-x_k}{2}\right)}\frac{\langle 2\rangle_k\langle K-1\rangle_k}{1 + \eta_k + O(u^2)} \\
&= \sum_{k=0}^{K-1}\frac{(-1)^k f_k}{\sin\left(\frac{x-x_k}{2}\right)}\langle K+1\rangle_k\big(1 - \eta_k + O(u^2)\big),
\end{aligned}$$

where we have picked up one rounding error from the division in each term and $K-1$ rounding errors from the $K-1$ additions in the sum[5], and have used the expansion $1/(1+\varepsilon) = 1 - \varepsilon + O(\varepsilon^2)$. The denominator of (1.2) may be handled similarly. Adding one more rounding error to account for the final division, we arrive at

$$\widehat{t_{f,X}}(x) = \frac{\displaystyle\sum_{k=0}^{K-1}\frac{(-1)^k f_k}{\sin\left(\frac{x-x_k}{2}\right)}\langle K+2\rangle_k\big(1 - \eta_k + O(u^2)\big)}{\displaystyle\sum_{k=0}^{K-1}\frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)}\langle K+1\rangle_k\big(1 - \eta_k + O(u^2)\big)}.$$

This expression is similar in form to the corresponding one obtained by Higham (2004) in the polynomial case, the key difference being the presence of the $1 - \eta_k + O(u^2)$ factors, which represent the error due to the conditioning of the sine evaluations.

[5] The order in which the terms are summed does not matter here (Higham, 2002, Chapter 4).

Next, just as in the proof of Theorem 4.1 of Higham (2004), we have

$$\begin{aligned}\left|\frac{t_{f,X}(x)-\widehat{t}_{f,X}(x)}{t_{f,X}(x)}\right| &\le \left(K+2+2\left(\max_{0\le k\le K-1}\left|\frac{x-x_k}{2}\cot\left(\frac{x-x_k}{2}\right)\right|\right)\right) u\,\frac{\sum_{k=0}^{K-1}\left|\frac{f_k}{\sin\left(\frac{x-x_k}{2}\right)}\right|}{\left|\sum_{k=0}^{K-1}\frac{(-1)^k f_k}{\sin\left(\frac{x-x_k}{2}\right)}\right|}\\ &\quad+\left(K+1+2\left(\max_{0\le k\le K-1}\left|\frac{x-x_k}{2}\cot\left(\frac{x-x_k}{2}\right)\right|\right)\right) u\,\frac{\sum_{k=0}^{K-1}\left|\frac{1}{\sin\left(\frac{x-x_k}{2}\right)}\right|}{\left|\sum_{k=0}^{K-1}\frac{(-1)^k}{\sin\left(\frac{x-x_k}{2}\right)}\right|}+O(u^2)\\ &=\left(K+2+2\left(\max_{0\le k\le K-1}\left|\frac{x-x_k}{2}\cot\left(\frac{x-x_k}{2}\right)\right|\right)\right) u\kappa(x,X,f)\\ &\quad+\left(K+1+2\left(\max_{0\le k\le K-1}\left|\frac{x-x_k}{2}\cot\left(\frac{x-x_k}{2}\right)\right|\right)\right) u\kappa(x,X,1)+O(u^2), \end{aligned}\tag{4.5}$$

where the second step follows from Lemma 4.1. (Here, the 1 in $\kappa(x, X, 1)$ refers to a vector of interpolation data whose entries are all 1.)

The only potential problem with this bound is in the terms involving the cotangent function, which can be large if $|x - x_k|/2$ is close to π for some k, reflecting the poor conditioning of sine near $\pm\pi$. Most dramatically, if $\alpha = 0$ then $x_0 = 0$, and $\cot\big((x - x_0)/2\big)$ becomes unbounded as x gets close to 2π. Additionally, in such cases, the error term represented by the $O(u^2)$ symbol may not be negligible, since the implied constant contains terms with $\cot\big((x - x_k)/2\big)$ as a factor for each k.

These remarks do not apply to the algorithm of Section 3, however, because its rules prevent (1.2) from being used in these problematic cases. Since we are assuming $\alpha \in [0, 1/2]$, it will only be used if $x \in [0, 2\pi - \pi(1 - 2\alpha)/K]$. The reason for this particular choice of restriction is given by Lemma A.1, presented in Appendix A, which gives a very simple bound for the cotangent terms in (4.5). Applying this result to (4.5), for $x \in [0, 2\pi - \pi(1 - 2\alpha)/K]$, we obtain

$$\left|\frac{t_{f,X}(x)-\widehat{t}_{f,X}(x)}{t_{f,X}(x)}\right| \le 5Ku\kappa(x,X,f)+(5K-1)u\kappa(x,X,1)+O(u^2). \tag{4.6}$$

On the other hand, if $x \in (2\pi - \pi(1 - 2\alpha)/K, 2\pi]$, we use (3.1) instead of (1.2). We handle the argument to sine in the modified terms as follows. Write $\mathrm{fl}(2\pi) = 2\pi(1 + \delta_{2\pi})$, where $|\delta_{2\pi}| \le u$. Then,

$$\mathrm{fl}\big(x-\mathrm{fl}(2\pi)\big)=\big(x-2\pi(1+\delta_{2\pi})\big)(1+\delta_{0,1})=\big(x-2\pi\big)\left(1-\frac{2\pi}{x-2\pi}\delta_{2\pi}\right)(1+\delta_{0,1}),$$

where $|\delta_{0,1}| \le u$. Next, we subtract the correction term c to adjust for the error in the approximation $\mathrm{fl}(2\pi) \approx 2\pi$ as explained in Section 3. As c is by definition the nearest floating-point number to $2\pi - \mathrm{fl}(2\pi) = -2\pi\delta_{2\pi}$, we have $c = -2\pi\delta_{2\pi}(1 + \delta_c)$ with $|\delta_c| \le u$. Therefore,

$$\begin{aligned}\mathrm{fl}\big(\big(x-\mathrm{fl}(2\pi)\big)-c\big)&=\big(\mathrm{fl}\left(x-\mathrm{fl}(2\pi)\right)-c\big)(1+\delta_{0,2})\\ &=\left(\big(x-2\pi\big)\left(1-\frac{2\pi}{x-2\pi}\delta_{2\pi}\right)(1+\delta_{0,1})+2\pi\delta_{2\pi}+2\pi\delta_{2\pi}\delta_c\right)(1+\delta_{0,2})\end{aligned}$$

$$= (x - 2\pi)\left(1 + \frac{2\pi\delta_{2\pi}(\delta_c - \delta_{0,1})}{x - 2\pi} + \delta_{0,1}\right)(1 + \delta_{0,2}),$$

where $|\delta_{0,2}| \leq u$. Since x is a floating point number, and since $\mathrm{fl}(2\pi)$ is the nearest floating-point number to 2π, we have $|x - 2\pi| \geq |\,\mathrm{fl}(2\pi) - 2\pi| = 2\pi|\delta_{2\pi}|$. Thus,

$$\left|\frac{2\pi\delta_{2\pi}(\delta_c - \delta_{0,1})}{x - 2\pi}\right| \leq |\delta_c| + |\delta_{0,1}| \leq 2u,$$

and so we may write $\mathrm{fl}\big(\big(x - \mathrm{fl}(2\pi)\big) - c\big) = (x - 2\pi)(1 + \widehat{\xi}_{0,1})(1 + \delta_{0,2})$, where $|\widehat{\xi}_{0,1}| \leq 3u$. Multiplying out the error terms and making the reasonable assumption that $|\widehat{\xi}_{0,1}\delta_{0,2}| \leq u$, which will hold if $3u \leq 1$, we can simplify this to $\mathrm{fl}\big(\big(x - \mathrm{fl}(2\pi)\big) - c\big) = (x - 2\pi)(1 + \tilde{\xi}_{0,1})$, where $|\tilde{\xi}_{0,1}| \leq 5u$.

These developments allow us to write, in analogy to (4.4),

$$\mathrm{fl}\left(\frac{\big(\big(x - \mathrm{fl}(2\pi)\big) - c\big) - x_0}{2}\right) = \frac{\big((x - 2\pi)(1 + \tilde{\xi}_{0,1}) - x_0\big)(1 + \delta_{0,3})}{2}(1 + \delta_{0,4})$$
$$= \frac{x - x_0 - 2\pi}{2}(1 + \xi_{0,1})(1 + \delta_{0,3})(1 + \delta_{0,4}), \tag{4.7}$$

where $\tilde{\xi}_{0,1}$ is as above, $|\delta_{0,3}|$ and $|\delta_{0,4}|$ are both at most u, and $\xi_{0,1} = \tilde{\xi}_{0,1}\big((x - 2\pi)/(x - x_0 - 2\pi)\big)$. Since $x - 2\pi$ and $-x_0$ have the same sign, $|(x - 2\pi)/(x - 2\pi - x_0)| \leq 1$, and so $|\xi_{0,1}| \leq |\tilde{\xi}_{0,1}| \leq 5u$. Note that this is a consequence of our having grouped the terms as prescribed in Section 3. From here, we work similarly to before and arrive at

$$\left|\frac{t_{f,X}(x) - \widehat{t}_{f,X}(x)}{t_{f,X}(x)}\right| \leq (K + 2 + C)u\kappa(x, X, f) + (K + 1 + C)u\kappa(x, X, 1) + O(u^2),$$

where

$$C = \max\left(2\left(\max_{1\leq k\leq K-1}\left|\frac{x - x_k}{2}\cot\left(\frac{x - x_k}{2}\right)\right|\right), 7\left|\frac{x - x_0 - 2\pi}{2}\cot\left(\frac{x - x_0 - 2\pi}{2}\right)\right|\right).$$

The factor of 7 in the second argument to the outer instance of max comes from the fact that there are three rounding error terms in (4.7) that add up to at most $7u$ compared with the two in (4.4) that add up to at most $2u$. To bound C, we use Lemma A.2, which is given in Appendix A. Combining this with Lemma A.1, we have

$$C \leq \max(4K - 2, 7) \leq 4K + 5,$$

and so

$$\left|\frac{t_{f,X}(x) - \widehat{t}_{f,X}(x)}{t_{f,X}(x)}\right| \leq (5K + 7)u\kappa(x, X, f) + (5K + 6)u\kappa(x, X, 1) + O(u^2) \tag{4.8}$$

for $x \in (2\pi - \pi(1 - 2\alpha)/K, 2\pi]$. In fact, noting that (4.8) is slightly weaker than (4.6), we see that (4.8) actually holds for $x \in [0, 2\pi]$.

To finish, we note, again following Higham (2004), that $\kappa(x, X, 1)$ is bounded above by the Lebesgue constant for the interpolation problem, which is at most $(2/\pi)\log(K)+2$ in our setting Cheney & Rivlin (1976). Combining this with (4.8) yields (4.3). This completes the proof of Theorem 4.2. □

Note carefully that this analysis depends strongly on the evaluation point x and the interpolation points x_k being in $[0, 2\pi]$. In particular, for the $\alpha < 1/2$ case that we described in detail, it does *not* apply to evaluations at $x = \mathrm{fl}(2\pi)$ if $\mathrm{fl}(2\pi) > 2\pi$.[6] The reason is that, under these circumstances, $x - 2\pi$ and x_0 may have the same sign, and so the factor multiplying $\tilde{\xi}_{0,1}$ to define $\xi_{0,1}$ in (4.7) can be large if x_0 is chosen carefully. In IEEE double-precision arithmetic, $\mathrm{fl}(2\pi) < 2\pi$, so this is not a problem; however, in IEEE single-precision arithmetic, $\mathrm{fl}(2\pi) > 2\pi$, and it is not difficult to construct a numerical example in which the algorithm of Section 3 is unstable for $x = \mathrm{fl}(2\pi)$.

If $\mathrm{fl}(2\pi) > 2\pi$ and a stable evaluation at $x = \mathrm{fl}(2\pi)$ is desired, it can be accomplished with the aid of a second correction term. To see this, note first that since $x = \mathrm{fl}(2\pi)$, when we compute $x - x_0 - 2\pi$ as prescribed in Section 3, the subtraction $x - \mathrm{fl}(2\pi)$ evaluates exactly to zero. Thus, we are left to evaluate

$$\mathrm{fl}(-c - x_0) = (2\pi\delta_{2\pi} + 2\pi\delta_{2\pi}\delta_c - x_0)(1+\delta_1) = (x - x_0 - 2\pi)\left(1 + \frac{2\pi\delta_{2\pi}\delta_c}{x - x_0 - 2\pi}\right)(1+\delta_1), \quad (4.9)$$

where $|\delta_1| \le u$ and we have used the fact that $x = \mathrm{fl}(2\pi) = 2\pi(1+\delta_{2\pi})$. Since c is by definition the nearest floating-point number to $-2\pi\delta_{2\pi}$, and since $-x_0$ is a floating-point number, we have $|x - x_0 - 2\pi| = |-x_0 - (-2\pi\delta_{2\pi})| \ge |c - (-2\pi\delta_{2\pi})| = 2\pi|\delta_{2\pi}\delta_c|$. Thus, $|2\pi\delta_{2\pi}\delta_c/(x - x_0 - 2\pi)|$ could be as large as 1, and if this is the case, we will not have computed $x - x_0 - 2\pi$ accurately.

This calculation highlights that the problem is due to the fact that $|x - x_0 - 2\pi|$ can be as small as $2\pi|\delta_{2\pi}\delta_c|$, which is $O(u^2)$, while we have only corrected for the error in $\mathrm{fl}(2\pi) \approx 2\pi$ down to $O(u)$. This naturally suggests a fix of subtracting an additional term that corrects the error down to $O(u^2)$.

Indeed, let c_2 be the nearest floating-point number to $2\pi\delta_{2\pi}\delta_c$. We have $c_2 = 2\pi\delta_{2\pi}\delta_c(1+\delta_{c_2})$, where $|\delta_{c_2}| \le u$. In IEEE double precision, $c_2 = -5.989539619436679 \times 10^{-33}$, and in single precision, $c_2 = -6.860498 \times 10^{-15}$. Subtracting c_2 from the result of (4.9), we obtain

$$\begin{aligned}\mathrm{fl}\big((-c - x_0) - c_2\big) &= \left((x - x_0 - 2\pi)\left(1 + \frac{2\pi\delta_{2\pi}\delta_c}{x - x_0 - 2\pi}\right)(1+\delta_1) - 2\pi\delta_{2\pi}\delta_c - 2\pi\delta_{2\pi}\delta_c\delta_{c_2}\right)(1+\delta_2)\\ &= (x - x_0 - 2\pi)\left(1 + \delta_1 + 2\pi\delta_{2\pi}\delta_c\frac{\delta_1 - \delta_{c_2}}{x - x_0 - 2\pi}\right)(1+\delta_2),\end{aligned}$$

where $|\delta_2| \le u$. Using the lower bound on $|x - x_0 - 2\pi|$ just derived and collecting the error terms, we find that $\mathrm{fl}\big((-c - x_0) - c_2\big) = (x - x_0 - 2\pi)(1+\xi)$ with $|\xi| \le 5u$ (assuming that $3u \le 1$). This is certainly accurate enough for our purposes.

Similar remarks apply in the $\alpha > 1/2$ case if x_{K-1} is taken to be $\mathrm{fl}(2\pi)$.

5. Interpolation on other intervals

With the main result of the article now established, in the remaining two sections, we examine what happens to our discussions in some settings beyond the one we have considered up to this point.

[6] Note that $\mathrm{fl}(2\pi)$ is the only floating-point number x in $[0, \mathrm{fl}(2\pi)]$ for which one can have $x > 2\pi$, for if $2\pi < x < \mathrm{fl}(2\pi)$, then x would be a closer floating-point number to 2π than $\mathrm{fl}(2\pi)$.

Thus far, we confined our discussion to interpolation on the interval $[0, 2\pi]$. At first glance, it would seem that much of what we have said translates directly to other intervals with little additional work, since (1.2) holds as-is for x and x_k drawn from *any* given interval of length 2π, a consequence of its depending only on the values of $x - x_k$ for each k and not on the values of x and x_k individually. In fact, the issue is more subtle, as we now explain.

The instability in (1.2) that we presented in Section 2 arises when poor conditioning of the sine function amplifies rounding errors in the computation of $(x - x_k)/2$ into large relative errors in the computed value of $\sin\big((x - x_k)/2\big)$ for some k. If it happens that $(x - x_k)/2$ is computed exactly for all k for which the corresponding evaluation of sine is ill conditioned, this cannot occur, and (1.2) will perform the evaluation stably. We observed this behavior empirically in the numerical experiments of Section 2 involving the $\alpha = 0$ grid on $[0, 2\pi]$. In terms of the analysis of Section 4, this corresponds to having $\delta_{k,1} = \delta_{k,2} = 0$ in (4.4) for the relevant values of k so that the bound (4.6) for the relative error in using (1.2) to evaluate $t_{f,X}(x)$ in floating point arithmetic holds for all $x \in [0, 2\pi]$ instead of just for x in the restricted interval given there.

In IEEE floating-point arithmetic, which uses a base-2 floating-point system, multiplication and division by 2 are always exact, barring overflow and underflow. Thus, whether $(x - x_k)/2$ is computed exactly boils down to whether the subtraction $x - x_k$ is done exactly. When working on intervals other than $[0, 2\pi]$, especially those away from the origin, this can happen with a far greater frequency than one might initially expect, owing to the following theorem of Sterbenz (Higham, 2002, Chapter 2):

THEOREM 5.1 If s and t are floating-point numbers such that $t/2 \le s \le 2t$, then $\mathrm{fl}(s - t) = s - t$ in the absence of underflow.

Note that the hypotheses of the theorem imply that s and t are both non-negative; an analogous result can be stated when s and t are both negative. It is easy to check that for $a \ge 2\pi$, the condition $t/2 \le s \le 2t$ is satisfied for all $s, t \in [a, a + 2\pi]$. Hence, all of the subtractions $x - x_k$ that occur when using (1.2) to interpolate on such an interval will be done exactly, and it follows that (1.2) is stable! Similarly, (1.2) is stable for interpolation on all intervals of the form $[b - 2\pi, b]$, for $b \le -2\pi$. Thus, there is no need to modify (1.2) in these circumstances.

For other intervals, i.e., length-2π subintervals $[a, b]$ of $(-4\pi, 4\pi)$, we can interpolate stably in the vast majority of cases using a modified version of the algorithm of Section 3 under some additional assumptions that are given in the discussion below. The formulae (3.1) and (3.2) are still applicable; we just need to change how we compute the arguments to the sine function in the modified terms. If we can show that these can be computed to high relative accuracy, then the rest of the analysis in Section 4 can be applied with only very minor modifications to conclude that the resulting algorithm is stable. As before, there are two issues that must be handled:[7] how to group the terms and how to correct for the fact that 2π cannot be represented exactly in floating-point arithmetic.

For the former, the appropriate generalization in the case where $\alpha < 1/2$ is to compute $x - x_0 - 2\pi$ as $(x - b) - (x_0 - a)$, while for $\alpha > 1/2$, we compute $x - x_{K-1} + 2\pi$ as $(x - a) - (x_{K-1} - b)$. These arrangements have the same previously identified crucial property that the terms in the final subtraction have opposite signs so that the only cancellation that occurs is the benign cancellation in each of the individual subtractions $x - b$ and $x_0 - a$.

[7] We remark that similar issues—with similar resolutions—arise in the investigations of Mascarenhas & de Camargo (2017) into the effects of rounding errors in the interpolation points on the performance of the barycentric formulae for polynomial interpolation.

The latter issue is more delicate, since the approximation $\mathrm{fl}(2\pi) \approx 2\pi$ does not enter into the computation directly. Instead, what we must correct for is the deviation of $b - a$ from 2π that we get when a and b are floating-point numbers. Ideally, we would have $b - a = \mathrm{fl}(2\pi)$ so that we could correct the error using the quantity c introduced previously, but this is not guaranteed. In general, the most we can say is that $b - a = \mathrm{fl}(2\pi) + \gamma$ for some γ that is hopefully not too large.

This leads us to the two key assumptions we make for the remainder of this section. First, we assume that $|b - a - 2\pi| \le |x - x_0 - 2\pi|$ in the case where $\alpha < 1/2$; for $\alpha > 1/2$, we assume $|a - b + 2\pi| \le |x - x_{K-1} + 2\pi|$. These inequalities would hold if $b - a$ were exactly 2π, but they can fail when b and a are floating-point numbers whose difference merely approximates 2π.

Secondly, we assume that γ itself is a floating-point number. At first glance, this seems rather restrictive, but it actually holds quite often as we now explain. Suppose for the moment that $\pi \le b \le 4\pi$. Then, by Theorem 5.1, $b - \mathrm{fl}(2\pi)$ is exactly a floating-point number. If a has been chosen well, then a and $b - \mathrm{fl}(2\pi)$ will not be far from each other. If they are close enough to each other that the subtraction $\gamma = \big(b - \mathrm{fl}(2\pi)\big) - a$ can be done exactly in floating-point arithmetic, then we are done. Looking to Theorem 5.1 once again, this is guaranteed if $a/2 \le b - \mathrm{fl}(2\pi) \le 2a$. Similarly, if $-4\pi \le a \le -\pi$, then $a + \mathrm{fl}(2\pi)$ is exactly a floating-point number, and if $b/2 \le a + \mathrm{fl}(2\pi) \le 2b$, then $\gamma = b - \big(a + \mathrm{fl}(2\pi)\big)$ will be a floating-point number as well.

Since any length-2π subinterval $[a, b]$ of $[-4\pi, 4\pi]$ has either $-4\pi \le a \le -\pi$ or $\pi \le b \le 4\pi$, and since the conditions imposed by Theorem 5.1 are rather mild, γ will be exactly a floating-point number in virtually every case of practical interest. In particular, this is true for interpolation on $[-\pi, \pi]$ with $a = -\mathrm{fl}(\pi)$, $b = \mathrm{fl}(\pi)$ (in fact, $\gamma = 0$ in that case), which is arguably the most important interval for trigonometric interpolation aside from $[0, 2\pi]$. Note that the discussion of the preceding paragraph also gives a way to compute γ in floating-point arithmetic for a given a and b.

To convert γ into an approximation of $b-a-2\pi$, it remains to correct for the error in the approximation $\mathrm{fl}(2\pi) \approx 2\pi$. For reasons similar to those given in the remarks following the proof of Theorem 4.2, using c alone will not suffice. We must additionally correct for the rounding error in the approximation $c \approx -2\pi\delta_{2\pi}$ using the constant c_2 defined previously. We compute, in floating-point arithmetic,

$$\begin{aligned}\mathrm{fl}\big((\gamma - c) - c_2\big) &= \Big(\big(b - a - \mathrm{fl}(2\pi) - c\big)(1 + \delta_1) - c_2\Big)(1 + \delta_2)\\ &= \Big(\big(b - a - 2\pi + 2\pi\delta_{2\pi}\delta_c\big)(1 + \delta_1) - 2\pi\delta_{2\pi}\delta_c - 2\pi\delta_{2\pi}\delta_c\delta_{c_2}\Big)(1 + \delta_2)\\ &= (b - a - 2\pi)\left(1 + \delta_1 + 2\pi\delta_{2\pi}\delta_c\frac{\delta_1 - \delta_{c_2}}{b - a - 2\pi}\right)(1 + \delta_2),\end{aligned} \tag{5.1}$$

where $|\delta_1|$ and $|\delta_2|$ are at most u.

Our assumption that γ is a floating-point number has the consequence that we can bound $|b-a-2\pi|$ from below, for $|b-a-2\pi| = |b-a-\mathrm{fl}(2\pi)+2\pi\delta_{2\pi}| = |\gamma - (-2\pi\delta_{2\pi})|$. Since $-\gamma$ is a floating-point number, and since c is the closest floating-point number to $2\pi\delta_{2\pi}$, the right-hand side can be no smaller than $|c - (-2\pi\delta_{2\pi})| = 2\pi|\delta_{2\pi}\delta_c|$. It follows that

$$\left|2\pi\delta_{2\pi}\delta_c\frac{\delta_1 - \delta_{c_2}}{b - a - 2\pi}\right| \le |\delta_1| + |\delta_{c_2}| \le 2u,$$

and hence, multiplying out the error terms in (5.1), we find that we may write $\mathrm{fl}\big((\gamma - c) - c_2\big) = (b - a - 2\pi)(1 + \xi_\gamma)$, where $|\xi_\gamma| \le 5u$ and where we have implicitly made the reasonable assumption that $4u \le 1$. Thus, $\mathrm{fl}\big((\gamma - c) - c_2\big)$ is a high relative accuracy approximation to $b - a - 2\pi$.

At last, we can show how to compute the argument to the sine function in the modified terms of (3.1) and (3.2) to the needed accuracy. As usual, we consider the $\alpha < 1/2$ case, the $\alpha > 1/2$ case being similar. First, we compute, using the grouping of the terms prescribed earlier in this section

$$\begin{aligned}\mathrm{fl}\big((x-b)-(x_0-a)\big) &= \big((x-b)(1+\delta_1)-(x_0-a)(1+\delta_2)\big)(1+\delta_3)\\ &= \big(x-x_0-(b-a)\big)\left(1+\delta_1\frac{x-b}{(x-b)-(x_0-a)}+\delta_2\frac{x_0-a}{(x-b)-(x_0-a)}\right)\\ &\quad\times(1+\delta_3),\end{aligned}$$

where $|\delta_1|$, $|\delta_2|$ and $|\delta_3|$ are all at most u. Since $x-b$ and $-(x_0-a)$ have the same sign, the factors multiplying δ_1 and δ_2 in the second bracketed factor are at most 1 in absolute value, and it follows that we have $\mathrm{fl}\big((x-b)-(x_0-a)\big) = \big(x-x_0-(b-a)\big)(1+\xi_1)$, where $|\xi_1| \le 4u$, assuming that $2u \le 1$. Writing $c' = \mathrm{fl}\big((\gamma-c)-c_2\big)$ for brevity, we now apply the correction we just computed to obtain

$$\begin{aligned}\mathrm{fl}\Big(\big((x-b)-(x_0-a)\big)-c'\Big) &= \big((x-x_0-(b-a))(1+\xi_1)-(b-a-2\pi)(1+\xi_\gamma)\big)(1+\delta_1)\\ &= (x-x_0-2\pi)\left(1+\xi_1\frac{x-x_0-(b-a)}{x-x_0-2\pi}+\xi_\gamma\frac{b-a-2\pi}{x-x_0-2\pi}\right)(1+\delta_4),\end{aligned}$$

where $|\delta_4| \le u$. By our assumption that $|b-a-2\pi| \le |x-x_0-2\pi|$, the factors multiplying ξ_1 and ξ_γ in this equation are bounded in magnitude by 2 and 1, respectively. Thus, we have $\mathrm{fl}\Big(\big((x-b)-(x_0-a)\big)-c'\Big) = (x-x_0-2\pi)(1+\xi)$ with $|\xi| \le 15u$, assuming that $13u \le 1$, as desired.

Applying similar arguments for the case where $\alpha > 1/2$, we can summarize our findings in this section as follows:

- If $a \ge 2\pi$ or $b \le -2\pi$, we can interpolate stably using (1.2) directly.
- If $[a,b] \subset (-4\pi, 4\pi)$ with $b-a = \mathrm{fl}(2\pi)+\gamma$, we can interpolate stably if γ is exactly a floating-point number and if $|b-a-2\pi| \le |x-x_0-2\pi|$ when $\alpha \in [0,1/2)$ or $|a-b+2\pi| \le |x-x_{K-1}+2\pi|$ when $\alpha \in (1/2, 1]$. These assumptions are not always valid, but they hold in many cases.
- Under the conditions of the last item, the interpolant can be computed by first calculating γ via
  - $\big(b-\mathrm{fl}(2\pi)\big)-a$ if $\pi \le b < 4\pi$ or
  - $b-\big(\mathrm{fl}(2\pi)+a\big)$ if $-4\pi < a \le -\pi$

  and then computing the correction factor $c' = (\gamma-c)-c_2$. Then, there are three cases:
  - If $\alpha \in [0,1/2)$, use (1.2) for $x \in [a, b-\pi(1-2\alpha)/K]$. Otherwise, use (3.1) with $x-x_0-2\pi$ computed as $\big((x-b)-(x_0-a)\big)-c'$.
  - If $\alpha = 1/2$, use (1.2) for all $x \in [a,b]$.
  - If $\alpha \in (1/2, 1]$, use (1.2) for $x \in [a+\pi(2\alpha-1)/K, b]$. Otherwise, use (3.2) with $x-x_{K-1}+2\pi$ computed as $\big((x-a)-(x_{K-1}-b)\big)-c'$.

6. The case of an even number of interpolation points

Our analysis in this article has focused exclusively on the version of the second formula applicable to an odd number K of equispaced points. We close with a few words about the case of even K.

The even-K counterpart of (1.2) is

$$t_{f,X}(x) = \frac{\sum_{k=0}^{K-1} \frac{(-1)^k}{\tan\left(\frac{x-x_k}{2}\right)} f_k}{\sum_{k=0}^{K-1} \frac{(-1)^k}{\tan\left(\frac{x-x_k}{2}\right)}}, \tag{6.1}$$

that is, it is identical to (1.2) except that the sine function is replaced by the tangent Henrici (1979). The instability in (1.2) emerged from the poor conditioning of the sine function near $\pm\pi$. The tangent function is also poorly conditioned near $\pm\pi$, so one would expect (6.1) to suffer from a similar instability. Indeed, this is the case, and it can be corrected analogously.

This is not the end of the story, however, as the tangent function additionally suffers from poor conditioning near $\pm\pi/2$, suggesting the possibility of an instability in (6.1) when $|x - x_k|$ is close to π for some k, an instability that (1.2) does not possess. At first glance, it seems that this is not an issue, since, if $|x - x_k|$ is close to π, then $\tan\big((x - x_k)/2\big)$ is large. If the interpolation data in the vector f are all comparable in magnitude to one another, this means that the kth terms in the sums in (6.1) will be small relative to the rest so that, while they may have been evaluated inaccurately due to the poor conditioning, their contribution to the final result will be negligible. If the datum f_k is much larger than the others, however, it will offset the growth in $\tan\big((x - x_k)/2\big)$, the kth term in the numerator will not be relatively small and the instability will manifest itself.

We can illustrate these effects with the following numerical example. Let $\alpha = 0$ and $K = 6$, so that the grid points are $x_k = k\pi/3$, $0 \le k \le 5$, and let the interpolation data be $f_0 = f_2 = f_3 = f_4 = f_5 = 1$ and $f_1 = 10^{15}$. We evaluate the interpolant using (6.1) at several points x near $4\pi/3$ so that $|x - x_1|$ is close to π. Just as in the experiments of Sections 2 and 3, we perform the evaluation once using double-precision arithmetic and once using higher-precision arithmetic, and then compute the relative error in the former, taking the latter as ‘exact’. Since f_1 is considerably larger than the other interpolation data, we expect to see evidence of instability per the previous paragraph.

The results are displayed in Fig. 4(a), which plots the relative error as a function of the distance of x from $4\pi/3$. As predicted, the formula does indeed exhibit instability. The ‘pyramid’ shape of the error curve is due to the fact that $4\pi/3$ is a grid point, x_4. As x gets closer to $4\pi/3$, $\tan\big((x - x_4)/2\big)$ shrinks, causing the $k = 4$ term in the sum in the numerator of (6.1) to become larger. At the same time, the $k = 1$ term that suffers from the ill conditioning of the tangent function becomes smaller, as explained previously; it only retains its significance because of the large magnitude of f_1. For the value of f_1 that we have chosen, the $k = 1$ term is the dominant term in the sum until the distance from $4\pi/3$ has decreased to about 10^{-7}, after which the $k = 4$ term takes over. Since the evaluations of the tangent function in the $k = 4$ term are all well conditioned for x in the chosen range, it is computed to high relative accuracy. Hence, we expect the error in the overall evaluation to improve as it takes more and more control from the $k = 1$ term.

We can verify the correctness of this explanation by making the datum f_1 so large that the $k = 1$ term always makes a significant contribution to the sum, even when the distance between x and $4\pi/3$ is very small. In this case, we expect to see the error rise steadily as the distance shrinks. These expectations are confirmed by Fig. 4(b), which shows the results of running the same experiment with $f_1 = 10^{30}$.

Regrettably, this new instability is not as easily corrected as the one described in Section 2. The technique from Section 3 of using periodicity to change the interpolation grid is not applicable here: if $|x - x_k|$ is close to π, then $|x - (x_k + 2n\pi)|$ will be close to an odd multiple of π for any integer n, so the evaluation of $\tan\big((x - (x_k + 2n\pi))/2\big)$ associated with the adjusted point in the resulting modified

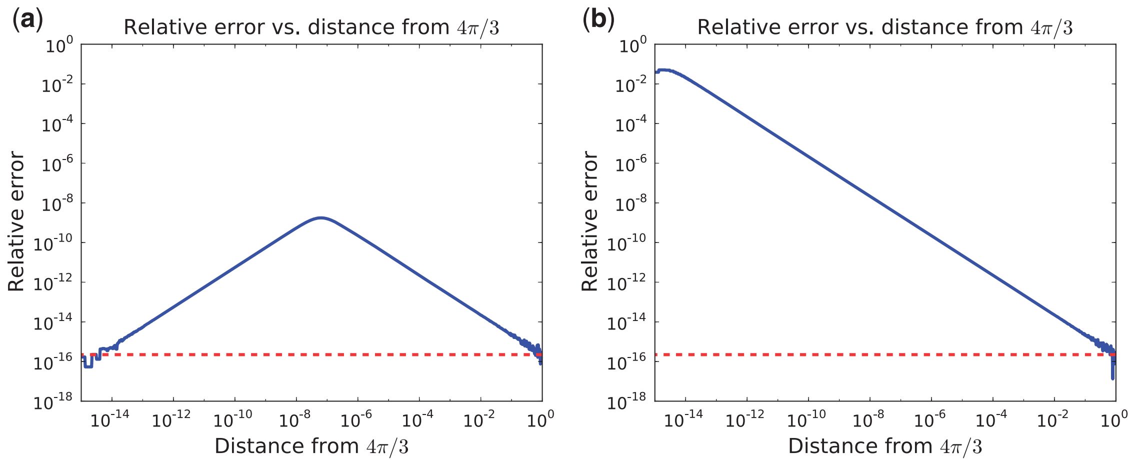

FIG. 4. Illustration of the instability in (6.1) when $|x - x_k|$ is close to π for some k. The solid lines show the relative error in the evaluations for the experiment described in Section 6 with (a) $f_1 = 10^{15}$ and (b) $f_1 = 10^{30}$. The dashed lines depict the product of the condition number (computed using the even-K analogue of Lemma 4.1) and the unit roundoff $u = 2^{-52}$.

formula is still poorly conditioned. Moreover, there are more ways to excite this new instability than there are for the previous one, since for a given interpolation grid, there are several choices of x and x_k such that $|x - x_k|$ is close to π, while there is only one such that $|x - x_k|$ is close to 2π. A method for stabilizing (6.1), if one exists, will likely require several modified formulae, one for each possible case, in addition to the even-K analogues of (3.1) and (3.2).

7. Conclusion

We have shown that, unlike its polynomial counterpart, the second barycentric formula for trigonometric interpolation is unstable for some evaluations. We have given a method for correcting this instability in the case where the number of interpolation points is odd, and have proved that the resulting algorithm is forward stable. We have investigated the extent to which our results for interpolation on $[0, 2\pi]$ can be extended to interpolation on other intervals of length 2π, and have established that this is possible in many cases. Finally, we have shown that if the number of interpolation points is even, the formula possesses an additional instability that is more challenging to correct.

Acknowledgements

We would like to thank Nick Trefethen for his guidance and support during our work on this project. We are also grateful for discussions we had on this subject with our colleagues Mohsin Javed, Hadrien Montanelli, Richard M. Slevinsky and Alex Townsend, some of whom also provided valuable commentary on an early draft of this paper. We thank André Camargo and Walter Mascarenhas for several useful remarks. Finally, we thank the anonymous referees, whose comments and suggestions led us to greatly improve the manuscript.

Funding

The European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement no. 291068. The views expressed in this article are not those of the ERC or the European Commission, and the European Union is not liable for any use that may be made of the information contained here.

References

BERRUT, J.-P. (1984a) Baryzentrische formeln zur trigonometrischen interpolation (I). *Z. Angew. Math. Phys.*, **35**, 91–105.

BERRUT, J.-P. (1984b) Baryzentrische formeln zur trigonometrischen interpolation (II): stabilität und anwendung auf die fourieranalyse bei ungleichabständigen stützstellen. *Z. Angew. Math. Phys.*, **35**, 193–205.

BERRUT, J.-P. & TREFETHEN, L. N. (2004) Barycentric Lagrange interpolation. *SIAM Rev.*, **46**, 501–517.

BEZANSON, J., KARPINSKI, S., SHAH, V. B. & EDELMAN, A. (2012) Julia: a fast dynamic language for technical computing. arXiv:1209.5145v1 [cs.PL].

CHENEY, E. W. & RIVLIN, T. J. (1976) A note on some Lebesgue constants. *Rocky Mountain J. Math*, **6**, 435–440.

DE LA VALLÉE POUSSIN, C.-J. (1908) Sur la convergence des formules d'interpolation entre ordonnées équidistantes. *Acad. Roy. Belg. Bull. Cl. Sci.*, **1908**, 319–410.

FOUSSE, L., HANROT, G., LEFÈVRE, V., PÉLISSIER, P. & ZIMMERMANN, P. (2007) MPFR: a multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Software*, **33**, 13:1–13:15.

GAUSS, C. F. (1866) Theoria interpolationis methodo nova tractata. *Werke,* vol. III. Göttingen: Dieterich, pp. 265–327.

GAUTSCHI, W. (2001) Barycentric formulae for cardinal (SINC-) interpolants by Jean-Paul Berrut. *Numer. Math.*, **87**, 791–792.

HENRICI, P. (1979) Barycentric formulas for interpolating trigonometric polynomials and their conjugates. *Numer. Math.*, **33**, 225–234.

HIGHAM, N. J. (2002) *Accuracy and Stability of Numerical Algorithms*, 2nd edn. Philadelphia: SIAM.

HIGHAM, N. J. (2004) The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, **24**, 547–556.

IEEE (2008) *IEEE Standard for Floating-Point Arithmetic*. IEEE Std. 754-2008.

MASCARENHAS, W. F. & DE CAMARGO, A. P. (2017) The effects of rounding errors in the nodes on barycentric interpolation. *Numer. Math.*, **135**, 113–141.

MULLER, J.-M., BRISEBARRE, N., DE DINECHIN, F., JEANNEROD, C.-P., LEFÈVRE, V., MELQUIOND, G., REVOL, N., STEHLÉ, D. & TORRES, S. (2010) *Handbook of Floating-Point Arithmetic*. Boston: Birkhäuser.

SALZER, H. E. (1948) Coefficients for facilitating trigonometric interpolation. *J. Math. Phys.*, **27**, 274–278.

SALZER, H. E. (1960) New formulas for trigonometric interpolation. *J. Math. Phys.*, **39**, 83–96.

WRIGHT, G. B., JAVED, M., MONTANELLI, H. & TREFETHEN, L. N. (2015) Extension of Chebfun to periodic functions. *SIAM J. Sci. Comput.*, **37**, C554–C573.

Appendix A—Lemmas

LEMMA A.1 If $\alpha \in [0, 1/2]$, then for $1 \leq k \leq K-1$ and all $x \in [0, 2\pi]$,

$$\left| \frac{x - x_k}{2} \cot\left(\frac{x - x_k}{2}\right) \right| \leq 2K - 1.$$

If $k = 0$, then the same holds for $x \in [0, 2\pi - \pi(1 - 2\alpha)/K]$.

Proof. We use the following inequality, valid for $t \in [-\pi, \pi]$, whose proof we omit:

$$|\cot(t)| \leq \max\left(\frac{1}{|t|}, \frac{1}{\pi - |t|}\right). \tag{A.1}$$

Thus, we have

$$\left|\frac{x - x_k}{2} \cot\left(\frac{x - x_k}{2}\right)\right| \leq \max\left(1, \frac{|x - x_k|}{2\pi - |x - x_k|}\right),$$

for $0 \leq k \leq K-1$, since $(x - x_k)/2 \in [-\pi, \pi]$. The second argument to max is maximized when $|x - x_k|$ is as close to 2π as possible. If $1 \leq k \leq K - 1$ then, since $\alpha \in [0, 1/2]$, this happens when $x = 0$ and $k = K - 1$, giving

$$\frac{|x - x_k|}{2\pi - |x - x_k|} \leq \frac{(K - 1 + \alpha)\frac{2\pi}{K}}{2\pi - (K - 1 + \alpha)\frac{2\pi}{K}} = \frac{K}{1 - \alpha} - 1 \leq 2K - 1.$$

On the other hand, if $k = 0$ and x is restricted to $[0, 2\pi - \pi(1 - 2\alpha)/K]$, then $|x - x_0|$ is closest to 2π when x is at the right endpoint of that interval, so

$$\frac{|x - x_0|}{2\pi - |x - x_0|} \leq \frac{2\pi - \pi\frac{1-2\alpha}{K} - \alpha\frac{2\pi}{K}}{2\pi - \left(2\pi - \pi\frac{1-2\alpha}{K} - \alpha\frac{2\pi}{K}\right)} = 2K - 1$$

as well. As $2K - 1 \geq 1$, this completes the proof. □

LEMMA A.2 If $\alpha \in [0, 1/2]$, then for $x \in (2\pi - \pi(1 - 2\alpha)/K, 2\pi]$,

$$\left|\frac{x - x_0 - 2\pi}{2} \cot\left(\frac{x - x_0 - 2\pi}{2}\right)\right| \leq 1.$$

Proof. Since $|x - x_0| \leq 2\pi$, we have $|x - x_0 - 2\pi| = 2\pi - (x - x_0)$, and for x in the given interval, we have $x \geq x_0$, so $|x - x_0| = x - x_0$. Thus, by (A.1),

$$\begin{aligned}\left|\frac{x - x_0 - 2\pi}{2} \cot\left(\frac{x - x_0 - 2\pi}{2}\right)\right| &= \left|\frac{x - x_0 - 2\pi}{2} \cot\left(\frac{x - x_0}{2}\right)\right| \\ &\leq \max\left(\frac{|x - x_0 - 2\pi|}{|x - x_0|}, \frac{|x - x_0 - 2\pi|}{2\pi - |x - x_0|}\right) \\ &= \max\left(\frac{2\pi - (x - x_0)}{x - x_0}, 1\right).\end{aligned}$$

The first argument to max in the final line is maximized when $x - x_0$ is as small as possible. Given the restrictions on x, this happens when $x = 2\pi - \pi(1 - 2\alpha)/K$, so we have

$$\frac{2\pi - (x - x_0)}{x - x_0} \leq \frac{2\pi - 2\pi + \pi\frac{1-2\alpha}{K} + \alpha\frac{2\pi}{K}}{2\pi - \pi\frac{1-2\alpha}{K} - \alpha\frac{2\pi}{K}} = \frac{1}{2K - 1}.$$

The result follows, since $1/(2K - 1) \leq 1$. □