PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Volume 10 · 2017 · Pages 23–32

A comparison between extended Floater–Hormann interpolants and trigonometric interpolation

André Pierro de Camargo[a]

Communicated by K. Hormann

Abstract

In this note we present a broad comparison between trigonometric interpolation and a specific version of extended Floater–Hormann interpolants which is accurate and stable for the interpolation of periodic functions at equally spaced nodes. The conclusion is that both techniques are equivalent in practice.

1 Introduction

In 2007, Floater and Hormann [9] introduced the following rational function $r_d(x,\mathbf{x},\mathbf{y})$ for interpolating a vector $\mathbf{y}=(y_0,y_1,\ldots,y_n)$ at the nodes $\mathbf{x}=(x_0,x_1,\ldots,x_n)$ which has no real poles and exactly reproduces polynomials of degree less than or equal to $0\leq d\leq n$ (for $d=n$, it reduces to the Lagrange polynomial interpolant)

$$r_d(x,\mathbf{x},\mathbf{y}):=\frac{\sum_{i=0}^{n-d}\lambda_i(x,\mathbf{x})p_i(x,\mathbf{x},\mathbf{y})}{\sum_{i=0}^{n-d}\lambda_i(x,\mathbf{x})}, \tag{1}$$

where $p_i(x,\mathbf{x},\mathbf{y})$ is the unique polynomial of degree at most d which interpolates $(x_i,y_i),\ldots,(x_{i+d},y_{i+d})$ and the functions $\lambda_i(x,\mathbf{x})$ are defined by

$$\lambda_i(x,\mathbf{x}):=\frac{(-1)^i}{(x-x_i)\ldots(x-x_{i+d})},\quad i=0,1,\ldots,n-d.$$

When the vector $\mathbf{y}$ comes from a smooth function $f:[a,b]\longrightarrow\mathbb{R}$, that is $\mathbf{y}=(f(x_0),f(x_1),\ldots,f(x_n))$, the interpolation error $|f(x)-r_d(x,\mathbf{x},\mathbf{y})|$ is

$$O(h^{d+1})$$

for $x\in[x_0,x_n]$, where h is the maximum distance between any two adjacent nodes. If the nodes are well spaced [5] and $d<<n$, then all the local grids formed by $d+1$ adjacent nodes are approximately equally spaced and, in this case, the resulting interpolant (1) should somehow inherit the properties of those local interpolants. For instance, the Lebesgue constant for Lagrange polynomial interpolation at $d+1$ equally spaced nodes,

$$x_i=a+ih,\ i=0,1\ldots,n,\ h:=\frac{b-a}{n}, \tag{2}$$

has the asymptotic limit

$$2^{d+1}/en\log(n),$$

while [4] gives the following bounds on the Lebesgue constant Λ_{r_d} of the interpolant (1)

$$\frac{1}{2^{d+2}}\,\frac{2^{2d+1}}{2d+2}\ln\Big(\frac{n}{d}-1\Big)\ \leq\ \frac{1}{2^{d+2}}\begin{pmatrix}2d+1\\ d\end{pmatrix}\ln\Big(\frac{n}{d}-1\Big)\ \leq\ \Lambda_{r_d}\ \leq\ 2^{d-1}(2+\ln(n)) \tag{3}$$

when the nodes $\mathbf{x}$ are equally spaced. This tells us that the interpolant (1) is ill conditioned for large values of d under (2). However, Klein [14] recently noticed that this bad conditioning affects (1) only close to the ends of the interpolation interval $[x_0,x_n]$. More precisely, the Lebesgue constant associated to the whole interpolation interval $[x_0,x_n]$ has exponential growth

[a]Centro de Matemática, Computação e Cognição; Universidade Federal do ABC - UFABC; Rua Santa Adélia, 166, bairro Bangu, cep 09210-170 ; Santo André, SP - Brasil (Email: andrecamargo.math@gmail.com).

with respect to d, but it has logarithmic growth with respect to d in the subinterval $[x_d, x_{n-d}]$, $n > 2d$. Based on this property, he introduced the extended Floater-Hormann interpolants, which use

$$\widetilde{r}_d(x, \mathbf{x}, f) \ := \ r_d(x, \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) \tag{4}$$

to approximate (in the interval $[a,b] = [x_0, x_n]$) a function $f : [x_0 - dh, x_n + dh] \longrightarrow \mathbb{R}$, where

$$\widetilde{\mathbf{x}} = (\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_N) \quad \text{and} \quad \widetilde{\mathbf{y}} = (\widetilde{y}_0, \ldots, \widetilde{y}_1, \ldots, \widetilde{y}_N), \quad N := n + 2d,$$

$$\widetilde{x}_i \ = \ x_0 + (i-d)h, \ i \ = \ 0, 1 \ldots, \ N, \ \ \widetilde{y}_i \ = \ f(\widetilde{x}_i), \ i \ = \ d, d+1 \ldots, \ n+d \tag{5}$$

and $\widetilde{y}_0, \ldots, \widetilde{y}_{d-1}, \widetilde{y}_{n+d+1}, \ldots, \widetilde{y}_N$ are approximations to the unknown values $f(\widetilde{x}_0), \ldots, f(\widetilde{x}_{d-1}), f(\widetilde{x}_{n+d+1}), \ldots, f(\widetilde{x}_N)$, respectively. Klein proposed a method for estimating these values, but it turned out that his method is not simultaneously accurate and stable [7]. In fact, [18] proves the impossibility of a stable and accurate approximation of arbitrary analytic functions whose values are known at equally spaced nodes, for example.

However, if f is a continuous periodic function with period $T = \frac{x_n - x_0}{M}$, for some positive integer M, then the unknown values of f can be recovered exactly from the original data $(\mathbf{x}, f(\mathbf{x}))$ and the resulting interpolant (4), with $\widetilde{\mathbf{y}}$ given by

$$\widetilde{y}_i \ = \ f(\widetilde{x}_i), i = 0, 1, \ldots, N, \tag{6}$$

is truly stable (even for $d \approx n$ or higher.) Klein actually advised the reader that $f(\widetilde{x}_0), \ldots, f(\widetilde{x}_{d-1}), f(\widetilde{x}_{n+d+1}), \ldots, f(\widetilde{x}_N)$ could be estimated by other methods, but he did not explicitly consider the interpolation of periodic functions. Moreover, what remains is to compare it with its natural competitor: trigonometric interpolation.

Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a continuous periodic function such that the length $b - a > 0$ of the interpolation interval is an integer multiple of f's period T. Up to an affine change of variable, say,

$$\varphi : [a,b] \longrightarrow [\gamma, \gamma + 2\pi], \quad \varphi(x) := \gamma + \frac{2\pi}{b-a}(x-a), \tag{7}$$

we may assume that

$$b - a \ = \ 2\pi. \tag{8}$$

In the case of the equally spaced nodes (2), perhaps the most natural approach would be to approximate f by the unique trigonometric polynomial

$$\psi(x, \mathbf{x}, \mathbf{y}) \ = \ \alpha_0 + \sum_{k=1}^{p} \alpha_k \cos(kx) + \sum_{k=1}^{q} \beta_k \sin(kx) \ \left(\begin{array}{l} p = q = \frac{n-1}{2}, n \text{ odd}, \\ p = q + 1 = \frac{n}{2}, n \text{ even} \end{array} \right) \tag{9}$$

that interpolates f at $x_0, x_1, \ldots, x_n$ [13].

In this note we present a broad comparison between the extended Floater-Hormann interpolants and trigonometric interpolation. The conclusion is that both techniques are equivalent in practice. Throughout the text, the sup norm of a continuous function g in the intervals $[a,b] = [x_0, x_n] = [\widetilde{x}_d, \widetilde{x}_{N-d}]$ and $[\widetilde{x}_0, \widetilde{x}_N]$ will be denoted by $||g||_\infty$ and $||g||_\infty^{ext}$, respectively, that is,

$$||g||_\infty := \max_{x \in [a,b]} |g(x)| \quad \text{and} \quad ||g||_\infty^{ext} := \max_{x \in [\widetilde{x}_0, \widetilde{x}_N]} |g(x)|. \tag{10}$$

We emphasize that the extended interpolants are only defined for equally spaced nodes.

2 Extended Floater–Hormann interpolants vs. trigonometric interpolation

2.1 Accuracy

2.1.1 Extended Floater–Hormann interpolants

The accuracy of the extended Floater–Hormann interpolants for arbitrary continuous functions can be understood in terms of the Lebesgue constant

$$\Lambda[\widetilde{r}_d](\mathbf{x}) \ := \ \max_{||f||_\infty^{ext} \leq 1} \left(\max_{x \in [x_0, x_n]} |\widetilde{r}_d(x, \mathbf{x}, f)| \right), \tag{11}$$

which is the norm of the linear operator $\Phi : \left(C[x_0 - dh, x_N + dh], ||.||_\infty^{ext} \right) \longrightarrow (C[x_0, x_n], ||.||_\infty)$,

$$f \ \stackrel{\Phi}{\longmapsto} \ \widetilde{r}_d(., \mathbf{x}, f).$$

Let $f \in C[x_0-dh, x_n+dh]$ and p_d^* be the best polynomial approximation (of degree less than or equal to d) for f in $C[x_0-dh, x_n+dh]$. Since $\widetilde{r}_d$ reproduces polynomials of degree less than or equal to d, then $p_d^* = \widetilde{r}_d(.,\mathbf{x},p_d^*)$ and we get

$$\begin{aligned} ||\widetilde{r}_d(.,\mathbf{x},f)-f||_\infty &\leq ||\widetilde{r}_d(.,\mathbf{x},f)-p_d^*||_\infty + ||p_d^*-f||_\infty \\ &= ||\widetilde{r}_d(.,\mathbf{x},f)-\widetilde{r}_d(.,\mathbf{x},p_d^*)||_\infty + ||p_d^*-f||_\infty \\ &\overset{(11)}{\leq} \Lambda[\widetilde{r}_d](\mathbf{x})||f-p_d^*||_\infty^{ext} + ||p_d^*-f||_\infty \\ &\leq (1+\Lambda[\widetilde{r}_d](\mathbf{x}))\,||p_d^*-f||_\infty^{ext}. \end{aligned} \tag{12}$$

The fundamental property of the extended Floater–Hormann interpolants is that the Lebesgue constant (11) has logarithmic growth with respect to d. More precisely, we have

Theorem 2.1 ([14])**.**

$$\Lambda[\widetilde{r}_d](\mathbf{x}) \leq 0.65(2+\log(N)),\quad N = n+2d,\ d \geq 5. \tag{13}$$

A consequence of (12) and (13) is that the sequence of extended Floater–Hormann interpolants $(\widetilde{r}_d(.,\mathbf{x},f))_{n\geq 2}$, $d=n$, converges uniformly when f is Lipschitz continuous (with the convention that $\mathbf{x}$ is formed by $n+1$ equally spaced points with fixed end points $x_0=a$ and $x_n=b$). This is a direct consequence of Jackson's theorem ([19], p. 196), which states that

$$||p_n^*-f||_\infty^{ext} \leq \frac{Lh(3n)\pi}{4(n+1)} = \frac{3L\pi(b-a)}{4(n+1)},\quad L := f\text{'s Lipschitz constant.}$$

For smooth functions f with $d+2$ continuous derivatives, Floater and Hormann [9] have presented the following bound for the interpolation error

$$|\widetilde{r}_d(x,\mathbf{x},f)-f| \leq h^{d+1}\left((n+2d)h\frac{||f^{(d+2)}||_\infty^{ext}}{d+2}+\frac{||f^{(d+1)}||_\infty^{ext}}{d+1}\right). \tag{14}$$

This bound is valid for all x in $[\widetilde{x}_0,\widetilde{x}_N]$, which is the interpolation interval for the nodes $\widetilde{\mathbf{x}}$. However, for x in the original interpolation interval $[a,b]$, this bound can be considerably improved (see Appendix A for a proof):

Theorem 2.2. *If $f \in C^{d+2}([\widetilde{x}_0,\widetilde{x}_N])$, $d\geq 1$ and $x \in [\widetilde{x}_d,\widetilde{x}_{N-d}]$, then*

$$|\widetilde{r}_d(x,\mathbf{x},f)(x)-f(x)| \leq \frac{1}{2^{d+1}}h^{d+1}\left((n+2d)h\frac{||f^{(d+2)}||_\infty^{ext}}{d+2}+\frac{||f^{(d+1)}||_\infty^{ext}}{d+1}\right). \tag{15}$$

In order to make a comparison, let us recall that the error of Lagrange interpolation for smooth functions with $(n+1)$ nodes $\mathbf{t}=(t_0,t_1,\ldots,t_n)$ in $[a,b]$ is bounded by (see, e. g., [12], p. 78)

$$|p_n(x)-f| \leq \frac{||f^{(n+1)}||_\infty}{(n+1)!}|\omega(x)|,\quad \omega(x) := \prod_{i=0}^{n}(x-t_i).$$

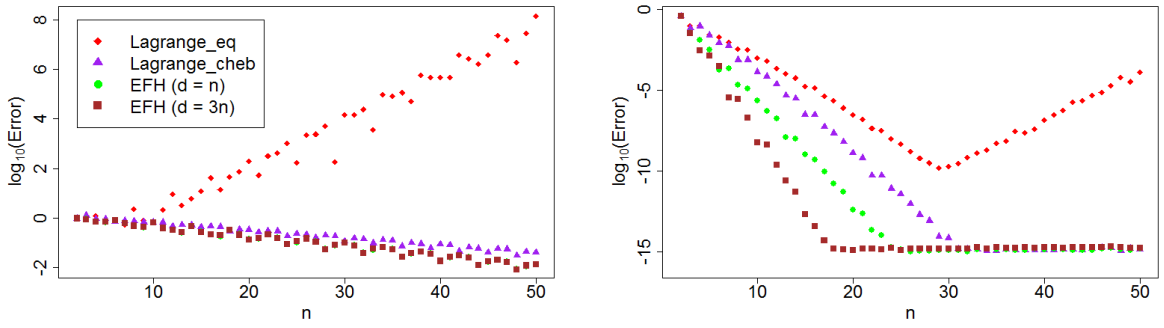
For $a=-1$ and $b=1$, the norm of the polynomial $\omega(x)$ is always greater than or equal to $\frac{1}{2^n}$, and it is equal to $\frac{1}{2^n}$ for the Chebyshev nodes of the first kind (see, e.g., [12], p. 88.) Therefore, the best estimate for the error of Lagrange interpolation with $(n+1)$ nodes in $[-1,1]$ is

$$\frac{||f^{(n+1)}||_\infty}{(n+1)!}\frac{1}{2^n} \approx \frac{||f^{(n+1)}||_\infty}{2^n\sqrt{2\pi n}}\left[\frac{e}{n}\right]^n, \tag{16}$$

while our bound (15) for $d=n-1$ is approximately $\dfrac{||f^{(n+1)}||_\infty}{2^n}\left[\dfrac{2}{n}\right]^n$ (when $||f^{(n)}||_\infty$ and $||f^{(n+1)}||_\infty$ are comparable) and this estimate is considerably smaller than the estimate (16) for large values of n.

It is worthwhile to notice that if the derivatives of f do not grow too fast, then (15) shows that we can accelerate the convergence of the extended interpolants by increasing d beyond[1] n, for a fixed value of n. This fact is illustrated in Figure 1 (a), which shows the maximum absolute value (in $[-1,1]$) of the interpolation error for $f(x)=\sin(\pi x)$. The values in this plot show that the extended Floater–Hormann interpolants (EFH), for $d=n$, can be more accurate than Lagrange interpolation at the Chebyshev nodes (Lagrange_cheb). A faster convergence is obtained for $d=3n$, as we already suspected. The values in plot (b) show that this can also happen for non-entire functions, such as $f=\cos\left(\sqrt{2+\sin(\pi x)}\right)$ (although the analysis of the growth of the derivatives in this case is not so trivial). However, we emphasize that (15) and (16) are just upper bounds and it is not true that extended Floater interpolants always perform better than Lagrange interpolation at the Chebyshev nodes. Moreover, for functions whose derivatives grows fast, e.g., the periodized Runge function $f(x)=\dfrac{1}{1+25\cos(x)^2}$, it may not be possible to accelerate the convergence of extended Floater-Hormann interpolants.

[1] Notice that the number of nodes $N+1 = n+2d+1$ which is used in the extended Floater interpolants is always greater than d.



(a) $f = \sin(\pi x)$ (b) $f = \cos(\sqrt{2+\sin(\pi x)})$

Figure 1: Interpolation error for $f = \sin(\pi x)$ (a) and for $f = \cos(\sqrt{2+\sin(\pi x)})$ (b).

2.1.2 Trigonometric interpolation

The convergence rate of trigonometric interpolation for Lipschitz continuous functions is studied in [17], for example. In addition, Jackson [11] derived a result analogous to (12) for trigonometric interpolation, and Cheney and Rivlin showed that the Lebesgue constant for trigonometric interpolation has logarithmic growth with respect to n [8]. An estimate for the interpolation error for smooth functions is presented in [3]:

$$|f(x)-\psi(x,\mathbf{x},\mathbf{y})| \leq \frac{2V_k}{n^k}\left(\frac{1}{n}+\frac{1}{k}\right), \tag{17}$$

for k times differentiable functions such that $f^{(k)}$ has bounded variation V_k in $[0,2\pi]$.

2.2 Performance

According to [13] and [20], two good options to compute (9) are either to compute the coefficients α_k, β_k by a discrete Fourier transform and apply (9) directly, or to use the barycentric formula

$$\psi(x,\mathbf{x},\mathbf{y}) = \begin{cases} \dfrac{\sum_{i=0}^{n-1}(-1)^i y_i \phi\left(\frac{(x-x_i)}{2}\right)}{\sum_{i=0}^{n-1}(-1)^i \phi\left(\frac{(x-x_i)}{2}\right)}, & \text{if } x \notin \{x_0,x_1,\ldots,x_n\}, \\ y_j, & \text{if } x = x_j, \end{cases} \tag{18}$$

$\phi = \csc$, n odd, or $\phi = \operatorname{ctg}$, n even. Here we consider only the second approach, because it is known that the computation of $\sin(kx)$ is unstable for large values of k [13]. However, if n is not large and one also needs to compute the derivatives of the interpolant, then the first approach is indicated. Alternatively, one can also use Baltensperger's formula [2].

Remark 1. The input data for trigonometric interpolants[2] (and also for extended Floater-Hormann interpolants) are the $n+1$-size vectors $\mathbf{x}$ and $\mathbf{y}$. It is useless to use the extended data $(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})$ (5) for trigonometric interpolants because they yield the same interpolants as for $(\mathbf{x},\mathbf{y})$. This is true by the unicity of the trigonometric interpolant.

Floater and Hormann [9] have shown that the interpolant (4) can be expressed in terms of the the second barycentric formula for rational interpolation

$$r_d(x,\widetilde{\mathbf{x}},\widetilde{\mathbf{y}}) = q(x,\widetilde{\mathbf{x}},\widetilde{\mathbf{y}},\mu_d(\widetilde{\mathbf{x}})) = \begin{cases} \sum_{i=0}^{N} \dfrac{\mu_d(\widetilde{\mathbf{x}})_i y_i}{x-x_i} \Big/ \sum_{i=0}^{N} \dfrac{\mu_d(\widetilde{\mathbf{x}})_i}{x-x_i}, & \text{if } x \notin \{\widetilde{x}_0,\widetilde{x}_1,\ldots,\widetilde{x}_N\}, \\ y_j, & \text{if } x = x_j \text{ for some } j, \end{cases} \tag{19}$$

where the barycentric weights are given by

$$\mu_d(\widetilde{\mathbf{x}})_i = \begin{cases} \sum_{j=\max\{0,i-d\}}^{\min\{n-d,i\}} (-1)^j \prod_{\substack{\tau=j \\ \tau\neq i}}^{j+d} \dfrac{1}{\widetilde{x}_i-\widetilde{x}_\tau}, & d \geq 1. \\ (-1)^i, & d = 0. \end{cases} \tag{20}$$

[2] Only the first n elements of $\mathbf{x}$ are used in (18).

Once the weights have been computed, (19) can be evaluated with $O(N)$ arithmetic operations. A naive algorithm for computing the weights

$$\mu_d(\widetilde{\mathbf{x}})_i, \quad i = 0, 1, \ldots, N. \tag{21}$$

through (20) uses $O(Nd^2)$ arithmetic operations. Hormann and Schaefer [10] recently showed that those weights can be computed with $O(Nd)$ arithmetic operations, which is a significant improvement over $O(Nd^2)$ for large values of d (such as $d = n$ as in Figure 1).

The algorithm by Hormann and Schaefer can be used for arbitrary nodes, but for equally spaced nodes the performance can still be improved. In Appendix B we present an algorithm (ALG) which computes all the weights (21) with only $O(N)$ arithmetic operations. This algorithm is crucial for a fair comparison, because the barycentric formula (18) for trigonometric interpolation can be computed with $O(n)$ arithmetic operations. The values in the next table shows that the use of algorithm (ALG) renders both approaches (trigonometric interpolation and extended Floater-Hormann) equivalent in practice in terms of performance.

n	80	160	320	640	1280	2560	5120	10240	20480	40960
$\frac{\text{time}(\widetilde{r})}{\text{time}(\psi)}$ $(*)$	1.31	1.36	1.30	1.24	1.34	1.36	1.35	1.34	1.54	1.80
$\frac{\text{time}(\widetilde{r})}{\text{time}(\psi)}$ $(**)$	3.60	3.53	3.44	3.36	3.43	3.67	3.75	3.66	4.04	4.57

Table 1: Time for evaluating the interpolants at 10^5 equally spaces points in $[0, 2\pi]$ $(d = n)$. $(*)$ weights are computed only once. $(**)$ weights are recomputed at each evaluation.

2.3 Numerical stability

The numerical error $\text{fl}(\widetilde{r}_d(x, \mathbf{x}, f)) - \widetilde{r}_d(x, \mathbf{x}, f)$ between the computed and the exact value of $\widetilde{r}_d(x, \mathbf{x}, f)$ satisfies

$$|\text{fl}(\widetilde{r}_d(x, \mathbf{x}, f)) - \widetilde{r}_d(x, \mathbf{x}, f)| \quad \leq \quad |\text{fl}(r_d(x, \widetilde{\mathbf{x}}, \text{fl}(\widetilde{\mathbf{y}}))) - r_d(x, \widetilde{\mathbf{x}}, \text{fl}(\widetilde{\mathbf{y}}))| \; + |r_d(x, \widetilde{\mathbf{x}}, \text{fl}(\widetilde{\mathbf{y}})) - r_d(x, \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})|, \tag{22}$$

where $\text{fl}(\widetilde{\mathbf{y}})$ denotes the rounded counterpart of $\widetilde{\mathbf{y}}$. It can be shown that the magnitude of the first summand in the right hand side of (22) is governed by the Lebesgue constant (11) (see [6] or [16], for example), which is identical to

$$\Lambda_{r_d}(\widetilde{\mathbf{x}}) \; := \; \max_{\substack{||v||_\infty \leq 1, \\ v \, \in \, \mathbb{R}^{N+1}}} ||r_d(., \widetilde{\mathbf{x}}, v)||_\infty. \tag{23}$$

The second summand in the right hand side of (22) can also be bounded in terms of (11) or (23). In fact, we have

$$|r_d(x, \widetilde{\mathbf{x}}, \text{fl}(\widetilde{\mathbf{y}})) - r_d(x, \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})| \; \leq \; \Lambda_{r_d}(\widetilde{\mathbf{x}}) \, ||\text{fl}(\widetilde{\mathbf{y}}) - \widetilde{\mathbf{y}}||_\infty.$$

For the interpolation of periodic functions, the $\widetilde{\mathbf{y}}$ in (6) can be defined by the following linear transformation of $\mathbf{y}$

$$\widetilde{y}_i \; = \begin{cases} y_{i-d}, & \text{if } \; d \leq i \leq n + d \\ \widetilde{y}_{i-d-n}, & \text{if } \; n + d < i \leq n + 2d \\ \widetilde{y}_{n+i-d}, & \text{if } \; 0 \leq i < d \end{cases} \tag{24}$$

and we have

$$||\text{fl}(\widetilde{\mathbf{y}}) - \widetilde{\mathbf{y}}||_\infty \; = \; ||\text{fl}(\mathbf{y}) - \mathbf{y}||_\infty. \tag{25}$$

Therefore, the numerical error (22) can be fully bounded in terms of the Lebesgue constant (23).

The main inconvenience of the definition of the extended Floater–Hormann interpolants for generic functions on p. 2276 of [14] is that, unlike (25), the method for approximating the unknown values $f(\widetilde{x}_0), \ldots, f(\widetilde{x}_{d-1}), f(\widetilde{x}_{n+d+1}), \ldots, f(\widetilde{x}_N)$ in [14] leads to large numerical errors:

$$||\text{fl}(\widetilde{\mathbf{y}}) - \widetilde{\mathbf{y}}||_\infty \; \gg \; ||\text{fl}(\mathbf{y}) - \mathbf{y}||_\infty.$$

This issue is discussed in detail in [7].

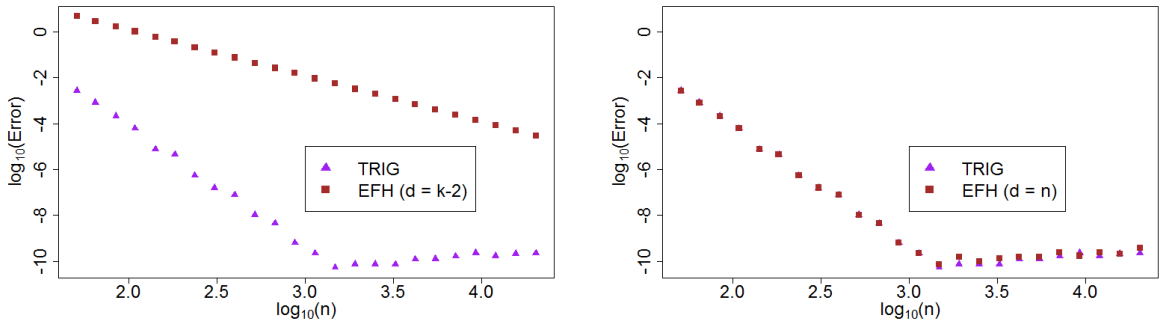
The complete stability analysis of the extended Floater–Hormann interpolants is subtle. For instance, when we round the nodes $\widetilde{\mathbf{x}}$ we must also be aware of the effects of these changes on the Lebesgue constant (23). It also depends on how the calculations are performed (the exact evaluation algorithm.) A detailed stability analysis for a few algorithms can be found in [6] and, for this reason, we shall not duplicate it here. Instead, we carry out an empirical analysis in the next section. For those interested in reading [6], it is sufficient to keep in mind the bounds (13) and (22) and the equality (25).

2.3.1 Numerical experiments

In this section we consider the approximation of the real 2π-periodic functions $f_{k,c}$ defined by

$$f_{k,c}(x) = (x-c)^{k+1}(x-c-2\pi)^{k+1} \quad \forall\ x\ \in\ [c, 2\pi+c].$$

Since the first k derivatives of $f_{k,c}$ vanish at $x = c$ and $x = 2\pi + c$, f_k is k-times continuously differentiable. The reference interval for approximation here is $[0, 2\pi]$ (we use the interval $[c, 2\pi + c]$ to describe the functions $f_{k,c}$ just for convenience.) Since $f_{k,c}$ is k-times differentiable, Theorem 2.2 gives an upper bound $O\left(\left[\frac{1}{n}\right]^{k-1}\right)$ for the error, for $d = k-2$, while the error for the trigonometric interpolant is $O\left(\left[\frac{1}{n}\right]^{k}\right)$ (see (17)). The consequences of this difference are shown in plot (a) of Figure 2. However, if use $d = n$, the methods become practically indistinguishable (see the values in plot (b)).



(a) $d = k-2$ (b) $d = n$

Figure 2: The overall interpolation error for $f_{3,\pi}$ in the presence of rounding errors.

In a previous draft of this manuscript we claimed that the extended Floater-Hormann interpolants are more stable than the trigonometric ones with respect to perturbations in the nodes. However, this is not true in general. Our claim was based on the error analysis of $f_{8,0}$ in an experiment executed in high arithmetic precision (100 digit accuracy). We performed the experiment in two settings: first using 100 digit accurate nodes and then rounding the nodes to double precision ($\approx$ 16 digit accuracy) in order to simulate a significant perturbation of the nodes (all other calculations were made in high precision.) The results of this experiment are displayed in Figure 3. We computed the weights for the extended Floater-Hormann interpolants with two different algorithms: using the algorithm (ALG) presented in Appendix B (EFH_{aw}) and evaluating the expression (20) directly (EFH). In this experiment we used $d = k$ because the methods are equivalent for this choice of d in terms of accuracy (see plot (a)) and the weights (20) can be quickly evaluated. The values in Figure 3 shows that there is a significant loss of accuracy for the trigonometric interpolant ($TRIG$) and for (EFH_{aw}) when the nodes are perturbed. The reason why the error gets stuck at some level for (EFH_{aw}) in plot (b) is that algorithm (ALG) evaluates the analytic expressions for the barycentric weights for equally spaced nodes, but in the situation considered in plot (b) the nodes are only about 16 digit accurate and these uncertainties are much larger than the rounding errors in this case, because all computations are made with 100 digit accuracy. The error introduced by combining analytic weights with rounded nodes is roughly

$$O(||f||_\infty N \log(d)\delta),$$

where δ is the maximum distance between the exact nodes and the rounded ones (see Lemma 2 and Theorem 2 of [6]).) In plot (a), the nodes have 100 digit accuracy and $\delta \approx 10^{-100}$ is negligible. In plot (b), however, the nodes are rounded to double precision and we have $\delta \approx 3 \times 10^{-16}$. Moreover, the supremum norm of $f_{8,0}$ is $||f_{8,0}||_\infty = \pi^{18} \approx 8.9 \times 10^8$. Combining this with N of order 10^3 explains the errors of order 10^{-5} in plot (b). We are unaware of any study regarding the effects of rounding the nodes for trigonometric interpolation. Austin and Xu [1] analyzed the stability of the trigonometric formula (18), but their analysis does not cover errors in the nodes. It may be possible to adapt the analysis of the rational barycentric case in [15] and [16] to the trigonometric one, but this still needs to be formalized. Nevertheless, this is not crucial in our case because we do not make any special claim regarding the stability of these interpolants.

The values in plot (b) of Figure 3 suggest that the extended Floater-Hormann interpolants with weights computed by evaluating (20) directly (EFH) are more robust than trigonometric interpolation with respect to perturbations of the nodes. **However**, this experiment is biased in favor of extended Floater-Hormann interpolants. This was noticed by an anonymous referee (and we are very thankful for that):

I am not sure whether the functions $f_{k,0}$ are so good examples. Does the vanishing of so many derivatives at the endpoints, where the FH interpolants are the least precise on average, not influence the results, in view of the fact that the function becomes almost a line there? I would prefer this singularity to lie inside the interval.

In fact, if we use $f_{k,\pi}$ instead of $f_{k,0}$ (putting the singularity in the middle of the interval) and re-execute the experiment described above, we obtain a completely different picture: the error for the extended Floater-Hormann interpolants can not be significantly

diminished by using the weights for rounded nodes (see Figure 4.) This is due to another source of error which passed unnoticed in the case $c = 0$: the values of $\tilde{y}_i$ and $\tilde{y}_{i+n}$ are identical (see (24)), but $\tilde{y}_{i+n}$ might not be close to $f_{k,c}(\tilde{x}_{i+n})$ for the perturbed nodes because $\tilde{x}_{i+n} - \tilde{x}_i$ might not be close to 2π. More precisely, a Taylor expansion of $f_{k,c}$ around $\tilde{x}_{i+n}$ gives

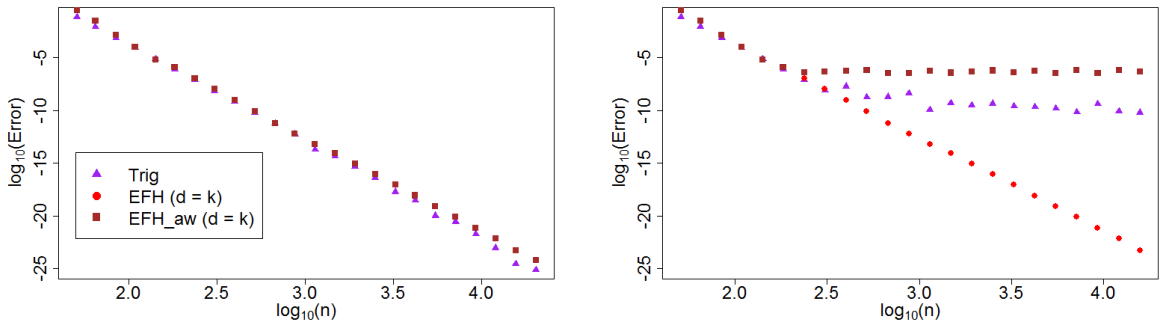
$$\begin{aligned} \tilde{y}_{i+n} &= \tilde{y}_i = f_{k,c}(\tilde{x}_i) = f_{k,c}(\tilde{x}_i + 2\pi) \\ &= f_{k,c}(\tilde{x}_{i+n}) + \sum_{j=1}^{k-1} \frac{f_{k,c}^{(j)}(\tilde{x}_{i+n})}{j!}[\tilde{x}_i + 2\pi - \tilde{x}_{i+n}]^j + \frac{f_{k,c}^{(k)}(\xi)}{k!}[\tilde{x}_i + 2\pi - \tilde{x}_{i+n}]^k. \end{aligned}$$

Hence, the error in estimating $f_{k,c}(\tilde{x}_{i+n})$ by $\tilde{y}_{i+n}$ is closely related to the displacements

$$\tilde{x}_i + 2\pi - \tilde{x}_{i+n} \tag{26}$$

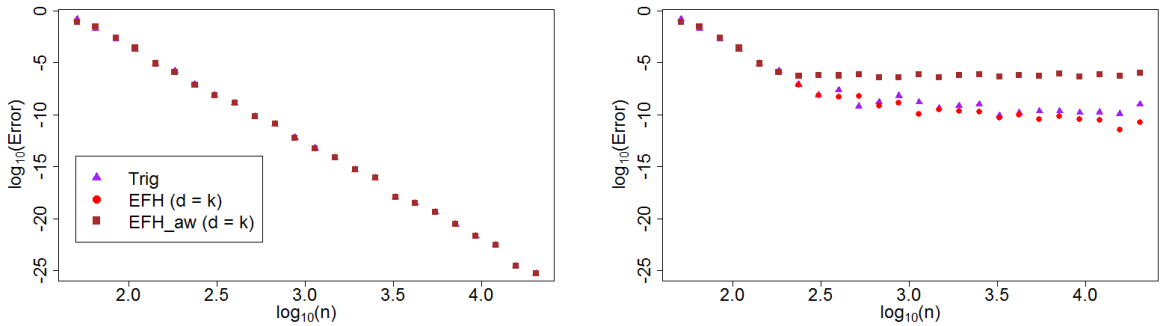
and to the derivatives of the interpolated function. In our case, the values (26) are about 6×10^{-16} for double precision nodes. For $c = 0$, the derivatives $f_{8,0}^{(j)}(x_{i+n})$ are very small for x_{i+n} close to the endpoint 2π (an instability region for Floater-Hormann interpolants), for example, because $f_{8,0}(2\pi) = 0$ and $f_{8,0}$ has its first eight derivatives vanishing at the endpoint $x = 2\pi$. On the other hand, $f_{8,\pi}$ has only the first derivative vanishing at this point and the difference between $f_{k,c}(\tilde{x}_{i+n})$ and $\tilde{y}_{i+n}$ becomes significant.

Therefore, the claim that the *extended Floater-Hormann interpolants are more stable than the trigonometric ones with respect to perturbations in the nodes* can not be supported. However, the values in all our figures also show that the converse can not be supported either, that is: the extended Floater-Hormann interpolants and trigonometric interpolants are equally stable in practice.



(a) $k = 3$ (b) $k = 8$

Figure 3: The interpolation error for $f_{k,0}$.



(a) $k = 3$ (b) $k = 8$

Figure 4: The interpolation error for f_3 (a) and f_8 (b).

3 Summary

In this note we investigated the properties of an accurate and stable version of extended Floater–Hormann interpolants for the approximation of periodic functions. For $d = n$, the extended Floater–Hormann interpolants have practically the same accuracy

as trigonometric interpolation and they can be computed equally fast with Algorithm (ALG). It is worthwhile noting that every particular trigonometric polynomial can be uniformly approximated by a sequence of extended Floater–Hormann interpolants (convergence is guaranteed by (15)) and this suggests that the extended Floater-Hormann interpolants and trigonometric interpolation should have similar asymptotic convergence properties. Our overall conclusion is that extended Floater-Hormann interpolants and trigonometric interpolation are equivalent in practice.

Acknowledgement

We thank Anthony Austin and Kuan Xu for fruitful discussions of their paper [1] and Pedro S. Peixoto for the comment on the derivatives of the trigonometric interpolants using Fourier transforms. We also thank the referees for their valuable suggestions and criticism regarding previous drafts of this manuscripts, especially for the comment quoted in Section 2.3.1 and for the inclusion of references [2] and [3].

A Proof of Theorem 2.2

Following the proof of Theorem 2 in [9] and recalling that $\widetilde{r}_d(.,\mathbf{x},f) := r_d(.,\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})$, we have

$$f(x)-\widetilde{r}_d(.,\mathbf{x},f)(x) \;=\; \sum_{i=0}^{N-d}(-1)^i f[\widetilde{x}_i,\ldots,\widetilde{x}_{i+d},x]\Big/\sum_{i=0}^{N-d}\lambda_i(x,\widetilde{\mathbf{x}}),$$

with

$$\left|\sum_{i=0}^{N-d}(-1)^i f[\widetilde{x}_i,\ldots,\widetilde{x}_{i+d},x]\right| \;\le\; \left((d+1)(n+2d)h\frac{||f^{(d+2)}||_\infty^{ext}}{(d+2)!}+\frac{||f^{(d+1)}||_\infty^{ext}}{(d+1)!}\right). \tag{27}$$

On the other hand, Section 4 of [9] shows that

$$\sum_{i=0}^{N-d}\lambda_i(x,\widetilde{\mathbf{x}}) \;=\; \sum_{i=0}^{N}\frac{w_i}{x-x_i},\quad \text{with}\quad w_i \;=\; \frac{(-1)^i}{d!h^d}\sum_{j=\max\{0,i-d\}}^{\min\{N-d,i\}}\binom{d}{i-j},\; i=0,1,\ldots,N. \tag{28}$$

Let $x\in[\widetilde{x}_d,\widetilde{x}_{N-d}]$, with $\widetilde{x}_k<x<\widetilde{x}_{k+1}$, for some index k. By rewriting the barycentric weights as

$$w_i \;=\; \begin{cases} \dfrac{(-1)^i}{d!h^d}\displaystyle\sum_{j=0}^{i}\binom{d}{j} & ,\; i<d,\\[2ex] \dfrac{(-1)^i}{d!h^d}\displaystyle\sum_{j=0}^{d}\binom{d}{j} \;=\; \dfrac{(-1)^i}{d!h^d}2^d & ,\; d\le i\le N-d,\\[2ex] \dfrac{(-1)^i}{d!h^d}\displaystyle\sum_{j=0}^{N-i}\binom{d}{j} & ,\; N-d+1\le i\le N, \end{cases} \tag{29}$$

one can easily check that both sequences $\frac{w_k}{x-\widetilde{x}_k},\frac{w_{k-1}}{x-\widetilde{x}_{k-1}},\ldots,\frac{w_0}{x-\widetilde{x}_0}$ and $\frac{w_{k+1}}{x-\widetilde{x}_{k+1}},\frac{w_{k+2}}{x-\widetilde{x}_{k+2}},\ldots,\frac{w_N}{x-\widetilde{x}_N}$ are decreasing in magnitude and alternate in sign. As a consequence, the denominator (28) can be grouped as a sum of terms of the same sign and we get

$$\left|\sum_{i=0}^{N-d}\lambda_i(x,\widetilde{\mathbf{x}})\right| \;\ge\; \left|\frac{w_{k-1}}{x-\widetilde{x}_{k-1}}+\frac{w_k}{x-\widetilde{x}_k}+\frac{w_{k+1}}{x-\widetilde{x}_{k+1}}+\frac{w_{k+2}}{x-\widetilde{x}_{k+2}}\right|.$$

In addition, we have

$$\left|\frac{w_{k-1}}{x-\widetilde{x}_{k-1}}\Big/\frac{w_k}{x-\widetilde{x}_k}\right| \le \left|\frac{x-\widetilde{x}_k}{x-\widetilde{x}_{k-1}}\right| = \left|1-\frac{\widetilde{x}_k-\widetilde{x}_{k-1}}{x-\widetilde{x}_{k-1}}\right| < \left|1-\frac{\widetilde{x}_k-\widetilde{x}_{k-1}}{\widetilde{x}_{k+1}-\widetilde{x}_{k-1}}\right| = \frac{1}{2}\quad\text{and, analogously,}\quad \left|\frac{w_{k+2}}{x-\widetilde{x}_{k+2}}\Big/\frac{w_{k+1}}{x-\widetilde{x}_{k+1}}\right| < \tfrac{1}{2}.$$

Consequently, since $\frac{w_k}{x-\widetilde{x}_k}$ and $\frac{w_{k+1}}{x-\widetilde{x}_{k+1}}$ have the same sign, we obtain

$$\left|\sum_{i=0}^{N-d}\lambda_i(x,\widetilde{\mathbf{x}})\right| \;\ge\; \frac{1}{2}\left(\left|\frac{w_k}{x-\widetilde{x}_k}\right|+\left|\frac{w_{k+1}}{x-\widetilde{x}_{k+1}}\right|\right) = \frac{2^d}{2d!h^d}\left|\frac{h}{(x-\widetilde{x}_k)(x-\widetilde{x}_{k+1})}\right| \ge \frac{2^d}{2d!h^d}\left|\frac{h}{(1/4)h^2}\right| = \frac{2^{d+1}}{d!h^{d+1}}. \tag{30}$$

Equation (15) follows by (27) and (30).

Remark 2. The first inequality in (30) can be used to derive a cleaner proof of logarithmic growth of the Lebesgue constant stated in Theorem 3.2 of [14] (our Theorem 2.1.) See the definition of the function $D(x)$ in equation (3.2) of [14].

B An efficient algorithm for computing the barycentric weights of Floater-Hormann interpolants for equally spaced nodes

An advantage of the barycentric representation of rational interpolants is that its weights can be rescaled in order to avoid overflow/underflow. In the case of the equally spaced nodes with which we are concerned, [9] shows that the weights (21) can be rescaled to the following integer weights

$$\left(\mu_{d,N}^{eq}\right)_i = (-1)^i \sum_{j=\max\{0,i-d\}}^{\min\{N-d,i\}} \binom{d}{i-j}, \; i = 0,1,\ldots,N, \tag{31}$$

which can be rewritten as

$$\left(\mu_{d,N}^{eq}\right)_i = \begin{cases} (-1)^i \sum_{j=0}^{i} \binom{d}{j} & , \; i < d. \\ (-1)^i \sum_{j=0}^{d} \binom{d}{j} = (-1)^i 2^d & , \; d \le i \le N-d. \\ (-1)^i \sum_{j=0}^{N-i} \binom{d}{j} & , \; N-d+1 \le i \le N. \end{cases} \tag{32}$$

By (32), we see that the absolute values of these weights are symmetric and, therefore, we only need to compute them for $i \le d$.

A side effect of having an interpolation formula for equally spaced nodes that is exact for polynomials of degree less than or equal to d can be readily seen in (32): the ratio 2^d between the largest and the smallest weight (in magnitude) grows exponentially with d. As a consequence, either overflow or underflow will occur in the computation of these weights for large values of d, no matter how we choose to rescale them. For instance, the smallest and largest IEE 754 double precision positive numbers are 2^{-1074} and 2^{1024}, respectively. Thus, either underflow or overflow will occour for $d > 1074 + 1024 = 2098$ in this case. Since the largest weights are those with indices between d and $N-d$ and we shall not evaluate (4) outside the interval $[\widetilde{x}_d, \widetilde{x}_{N-d}] = [x_0, x_n]$, it is convenient to rescale (32) in order to have underflow (possibly) in the smallest weights in magnitude, that is, those with smallest and largest indices. In our algorithm we chose to rescale the weights (32) by dividing them by the largest of the binomial coefficients $\binom{d}{j}$. More precisely, we compute

$$w_i = \left(\mu_{d,N}^{eq}\right)_i \Big/ \gamma, \; i = 0,1,\ldots,N, \quad \text{where} \quad \gamma := \begin{cases} \binom{d}{d/2}, & d \text{ even}, \\ \binom{d}{(d-1)/2}, & d \text{ odd}. \end{cases} \tag{33}$$

The algorithm is quite simple. First we compute all the normalized binomial coefficients

$$c_j = \binom{d}{j} \Big/ \gamma, j = 0,1,\ldots,d \tag{34}$$

and then we add these numbers in order to obtain the weights (33). The precise description of the algorithm for odd d (the other case is similar) is as follows.

Algorithm 1. [(ALG)]

- Set $c_{(d-1)/2} = 1$ and $c_{(d+1)/2} = 1$.
- Use the relations $c_{k-1} = c_k \frac{k}{d-k+1}, k = 1,2,\ldots,d-1$, to compute the other coefficients c_js (use it for $k = (d-1)/2, k = (d-1)/2-1,\ldots,k = 1$ to compute $c_{(d-1)/2-1}, c_{(d-1)/2-2},\ldots,c_0$ and use it for $k-1 = (d+1)/2$, $k-1 = (d+1)/2+1,\ldots,$ $k-1 = d-1$ to compute $c_{(d+1)/2+1}, c_{(d+1)/2+2},\ldots,c_d$.)
- Set $w_0 = c_0$ and use the relations $|w_{k+1}| = |w_k| + c_{k+1}, k = 0,1,\ldots,d-1$, and the fact that the weights oscillate in sign to compute the other weights.

Clearly, Algorithm (ALG) has complexity $O(d)$ in terms of the number of arithmetic operations. Our simulations have suggested that the underflow in the computation of the weights with smallest and largest indices for large values of d is not an issue and this is actually quite intuitive. In fact, the occurrence of underflow implies that the summands (in the denominator and the numerator of (19)) with the largest weights (in magnitude) are much larger than those summands of the tails which will vanish computationally, that is, the overall contribution of these small summands to (19) is irrelevant in practice. It is possible to formalize this argument in precise mathematical statements, but, for the sake of brevity, we will not do that here.

References

[1] Austin, A., and Xu, K., *On the numerical stability of the second barycentric formula for trigonometric interpolation in shifted equispaced points.* IMA J. Numer. Anal. (2016) DOI: 10.1093/imanum/drw038.

[2] Baltensperger, R., *Some results on linear rational trigonometric interpolation.* Comput. Math. Appl. 43 (2002) pp. 737–746.

[3] Berrut, J., and Welscher, A., *Fourier and barycentric formulae for equidistant Hermite trigonometric interpolation.* Appl. Comput. Harmon. Anal. 23 (2007) pp. 307–320.

[4] Bos, L., De Marchi, S., Hormann, K., and Klein, G., *On the Lebesgue constant of barycentric rational interpolation at equidistant nodes.* Numer. Math. 121 (3) (2012) pp. 461–471.

[5] Bos, L., De Marchi, S., Hormann, K., and Sidon, J., *Bounding the Lebesgue constant for Berrut's rational interpolant at general nodes.* J. Approx. Theory 169 (2013) pp. 7–22.

[6] Camargo, A., *On the numerical stability of Floater-Hormann's rational interpolant.* Numer. Algorithms 72 (1) (2016) pp. 131–152.

[7] Camargo, A., and Mascarenhas, W., *The stability of the extended Floater-Hormann interpolants.* Numer. Math. (2016) DOI: 10.1007/s00211-016-0840-z.

[8] Cheney, E., and Rivlin. T., *A note on some Lebesgue constants.* Rocky Mountain J. Math. 6 (3) (1976) pp. 435–439.

[9] Floater, S., and Hormann, K., *Barycentric rational interpolation with no poles and high rates of approximation.* Numer. Math. 107 (2) (2007) pp. 315–331.

[10] Hormann, K., and Schaefer, S., *Pyramid algorithms for barycentric rational interpolation.* Comput. Aided Geom. Design 42 (2016) pp. 1–6.

[11] Jackson, D., *On the accuracy of trigonometric interpolation.* Trans. Amer. Math. Soc. 14 (4) (1913) pp. 453–461.

[12] Gautschi, W., *Numerical Analysis, 2nd ed.*, Springer, New York (2012).

[13] Henrici, P., *Barycentric formulas for interpolating trigonometric polynomials and their conjugates.* Numer. Math. 33 (1979) pp. 225–234.

[14] Klein, G., *An extension of the Floater-Hormann family of barycentric rational interpolants.* Math. Comp. 82 (284) (2013) pp. 2273–2292.

[15] Mascarenhas, W., and Camargo, A., *The effects of rounding errors in the nodes on barycentric interpolation.* Numer. Math. 135 (1) (2017) pp. 113–141.

[16] Mascarenhas, W., and Camargo, A., *The backward stability of the second barycentric formula for interpolation.* Dolomites Res. Notes Approx. 7 (2014) pp. 1–12.

[17] Petras, K., *Error estimates for trigonometric interpolation of periodic functions in Lip 1.* in Chui C. K. and L. L. Schumaker, Approximation Theory VIII, World Scientific Publishing 1 (1995) pp. 459–466.

[18] Platte, R., Trefethen, L. N., and Kuijlaars, A., *Impossibility of fast stable approximation of analytic functions from equispaced samples.* SIAM Rev. 53 (2) (2011) pp. 308–318.

[19] Powell, M. J. D., *Approximation Theory and Methods.* Cambridge University Press, New York (1981).

[20] Wright, G., Javed, M., Montanelli, H., and Trefethen, L. N., *Extension of Chebfun to periodic functions* SIAM J. Sci. Comput. 37 (5) (2015) pp. C554–C573.