The numerical stability of extended Floater Hormann interpolants

André Pierro de Camargo and Walter F. Mascarenhas

the date of receipt and acceptance should be inserted later

Abstract We present a new analysis of the numerical stability of extended Floater-Hormann interpolants. Contrary to what is claimed in the current literature, we show that the Lebesgue constant of these interpolants can grow exponentially with the parameters that define them. We also show that extended interpolants are not backward stable and their extrapolation step is a source of numerical instability. We emphasize the importance of using the proper interpretation of the basic approximation theoretic concepts in order to evaluate correctly the effects of noise in practical approximation algorithms.

1 Introduction

Given nodes \( x = (x_0, \ldots, x_n) \) and function values \( y = (y_0, \ldots, y_n) \), the Floater-Hormann interpolation formula is defined as

\[
 r_d(t, x, y) := \frac{\sum_{i=0}^{n-d} \lambda_i(t, x) p_i(t, x, y)}{\sum_{i=0}^{n-d} \lambda_i(t, x)},
\]

where \( p_i(t, x, y) \) is the unique polynomial of degree at most \( d \) which interpolates \( y_i, y_{i+1}, \ldots, y_{i+d} \) at \( x_i, x_{i+1}, \ldots, x_{i+d} \), and the weights \( \lambda_i \) are defined as

\[
 \lambda_i(t, x) := \frac{(-1)^i}{(t - x_i)(t - x_{i+1}) \cdots (t - x_{i+d})}, \quad \text{for } i = 0, \ldots, n - d.
\]
In exact arithmetic, when \( f \) is smooth, the error incurred by the Floater-Hormann interpolant defined by \( x \) and \( d \) is of order \( h^{d+1} \), where
\[
h := \max_{0 \leq k < n} x_{k+1} - x_k.
\]

Unfortunately, \([9]\) proves that when the nodes are equally spaced the Lebesgue constant of the Floater-Hormann interpolant defined by \( x \) and \( d \) grows exponentially with \( d \). Therefore, the parameter \( d \) must be chosen carefully in order to balance the high order of approximation \( h^{d+1} \) with the numerical errors due to large Lebesgue constants.

In an attempt to reduce the effects of the large Lebesgue constants for equally spaced nodes, \([11]\) introduced the extended Floater-Hormann interpolants. These interpolants are defined in terms of extended nodes, \([11]\) introduced the extended Floater-Hormann interpolants. These interpolants are defined in terms of extended nodes \( \tilde{x}_{-d}, \tilde{x}_{-d+1}, \ldots, \tilde{x}_0, \ldots, \tilde{x}_n, \ldots, \tilde{x}_{n+d} \), with \( \tilde{x}_i = x_0 + i(x_1 - x_0) \), and extended function values \( \tilde{y}_{-d}, \ldots, \tilde{y}_0, \ldots, \tilde{y}_n, \ldots, \tilde{y}_{n+d} \). The extended Floater-Hormann corresponding to \( \tilde{y}, x \) and \( d \) is defined as
\[
\tilde{r}_d(t, x, \tilde{y}) := r_d(t, \tilde{x}, \tilde{y}).
\]

This definition does not say how the \( \tilde{y}_i \) should be chosen. It is natural to define \( \tilde{y}_i = y_i \) for \( 0 \leq i \leq n \), and \([10, 11]\) and \([12]\) suggest the use of extrapolation to obtain the remaining \( \tilde{y}_i \). In this case, one would choose parameters \( \tilde{n} \) and \( \tilde{d} \) and define \( \tilde{y}_{-\tilde{d}} \) as the value obtained by evaluating at \( \tilde{x}_{-\tilde{d}} \) the Taylor series of degree \( \tilde{d} \) of the usual Floater-Hormann interpolant corresponding to \( x_0, \ldots, x_{\tilde{n}} \). These extended Floater-Hormann interpolants \( \tilde{r}_{x, d, \tilde{n}, \tilde{d}}[y] \) are defined formally in \([11]\) and their numerical stability is the subject of the present article.

The article \([11]\) shows that the order of approximation of the extended interpolant \( \tilde{r}_{x, d, \tilde{n}, \tilde{d}}[y] \) is \( h^{D+1} \), where \( D = \min\{d, \tilde{d}\} \). Moreover, \([10, 11]\) and \([12]\) claim that the Lebesgue constant of this interpolant is bounded by \( 0.65 \log(n + 2d) \) for \( d \geq 5 \), regardless of the values of \( \tilde{n} \) and \( \tilde{d} \). This bound is misleading: as we show in the present article, the Lebesgue constant of the extended interpolant with \( d = \tilde{n} = \tilde{d} \) grows exponentially with \( d \), and it is much larger than the values reported by \([10, 11]\) and \([12]\) in other cases.

In fact, the logarithmic bound on the “Lebesgue constant” mentioned in \([10, 11]\) and \([12]\) is based on an unusual “interpretation” of this concept, which does not have the basic properties of the traditional Lebesgue constant. In particular, Theorem 5.1 in \([12]\) does not hold under the conditions displayed in italic in the last paragraph of the page preceding the statement of that theorem: “using only the given values of \( f \) in \( [a, b] \).”

To illustrate the instabilities of the extended Floater-Hormann interpolants based on extrapolation, in Section 2 we present a practical example showing that their Lebesgue constant grows exponentially when we follow the procedure suggested in \([10, 11]\) and \([12]\). In the remaining sections we present an analysis of extended Floater-Hormann interpolants based on the appropriate “interpretation” of the Lebesgue constant. Formally, this constant can be defined either as the norm of the linear operator \( y \mapsto r_{x, d, \tilde{n}, \tilde{d}}[y] \in C^0[a, b] \) or as the sup of the Lebesgue function
\[
A_{n, d, \tilde{n}, \tilde{d}}(t) := \sup_{y \text{ with } \|y\|_\infty = 1} \left| r_{x, d, \tilde{n}, \tilde{d}}[y](t) \right|.
\]
These two definitions are equivalent, and lead to a concept which has a fundamental role in theory and in practice. In addition to establishing misleading upper bounds on the Lebesgue constants, the current literature does not pay due attention to two important issues regarding extended interpolants. These interpolants require the evaluation of extrapolated function values, and we are not aware of any detailed analysis of the instabilities in this extrapolation process. The difficulty in choosing the three parameters \( d, \tilde{n} \) and \( \tilde{d} \) is also underestimated. For instance, [11] claims that extended interpolants are more robust to changes in \( d \) than usual Floater-Hormann interpolants. However, this robustness is only reached after \( \tilde{n} \) and \( \tilde{d} \) are defined, and [11] does not mention how \( \tilde{n} \) and \( \tilde{d} \) should be chosen.

In sections 3 and 4 we analyze the Lebesgue constants of extended interpolants from a theoretical perspective. We present an exponential lower bound on the Lebesgue constant when \( d = \tilde{n} = \tilde{d} \). We also present experimental data showing that the dependency of the Lebesgue function on these three parameters is subtle. Therefore, choosing them in practice may be at least as difficult as choosing the single parameter \( d \) for usual Floater-Hormann interpolants.

Sections 5 and 6 present an empirical analysis of the stability of extended interpolants. We show that these interpolants are not backward stable and, therefore, the Lebesgue constants may not tell the whole story about their numerical stability. We explain that the extrapolation step may lead to numerical instability, and due to this instability the overall error incurred by an extended interpolant can be much larger than \( n \Lambda_{x,d,\tilde{n},\tilde{d}} \epsilon \), where \( \Lambda_{x,d,\tilde{n},\tilde{d}} \) is its Lebesgue constant and \( \epsilon \) is the machine precision. In order to illustrate this fact, we present the results of experiments in which the accuracy of the extended interpolants is much worse than the accuracy of the usual Floater-Hormann interpolants.

On the positive side, once we become aware of the problems caused by the extrapolation step, we may consider ways to reduce them. When we evaluate the interpolants for many values of \( t \in [x_0, x_n] \), it is worth computing the relatively few extrapolated function values in multiple precision. Numerical experiments show that this strategy leads to more accurate extended interpolants.

Section 7 shows that extended interpolants are sensitive to the way we implement them. Depending on how we evaluate the extrapolated function values we may obtain inaccurate results. Moreover, even if we code the extrapolation step as suggested in [11], the accuracy of extended interpolants depends on the cancellation of rounding errors. This cancellation happens with high probability, and the inaccuracy of the extended interpolants considered in the experiments in section 7 is unlikely to occur in practice. However, our experiments indicate that it is difficult to formulate reasonable hypotheses under which we could prove upper bounds of order \( n \Lambda_{x,d,\tilde{n},\tilde{d}} \epsilon \) on the numerical errors incurred by extended interpolants. By contrast, under reasonable models of floating point arithmetic [8], we already have bounds on the effects of rounding errors for usual Floater-Hormann interpolants and other barycentric formulae [4] [7] [13] [14] [15].

2 The Lebesgue constant in practice

The Lebesgue constant is a fundamental concept in approximation theory. It is also fundamental in practice, because it measures the sensitivity of the interpolants
to perturbations (or noise) in the data. In its proper interpretation, the Lebesgue constant is equivalent to what numerical analysts call “condition number”, and use to evaluate the numerical stability of algorithms.

This section shows that in practice the condition number of extended Floater Hormann interpolants grows exponentially when \( d = \tilde{n} = \tilde{d} \). Therefore, the claims regarding the logarithmic growth of this condition number presented in [10], [11] and [12] are unfounded, and may lead the readers of these articles to believe that these interpolants are much less affected by noise than they really are.

We consider the approximation of \( f(t) = \sin(2t) \) in the interval \([-1, 1]\), and this simple example already illustrates the exponential growth of the condition number in practice. Our conclusions are summarized by the two plots in Figure 1. The plot on the left shows that by implementing the extrapolation procedure proposed in [10], [11] and [12] with the usual IEEE 754 double precision arithmetic we may have numerical errors of order \( 10^{14} \), while the logarithmic bound on the “Lebesgue constant” mentioned in these articles is of order \( 0.65 \log(n + 2d) \approx 3.7 \).

The plot on the right illustrates the sensitivity of extended interpolant to noise in the function values. This plot was obtained by adding random values of order \( 10^{-10} \) to \( y_0, \ldots, y_n \). The effects of rounding errors in this plot are negligible because we used the high precision arithmetic provided by the MPFR library [16], with a mantissa of 640 bits. The experiment on the right indicates that in this case the condition number is of order \( 10^{10} \), and not 3.7 as suggested by the bound in [10], [11] and [12].

Figure 1 raises an interesting question: why does the plot on the left display errors of order \( 10^{14} \) while the plot on the right shows errors of order one? This question is intriguing because the IEEE 754’s double precision machine epsilon is of order \( 10^{-16} \), and is much smaller than the \( O(10^{-10}) \) perturbations used to generate the plot on the right. As we explain in the rest of the article, the answer to this question lies in the instabilities in the extrapolation process, and this is one more reason why, in practice, the logarithmic bound presented in [10], [11] and [12] leads to misleading estimates of the effects of noise.

3 The barycentric and reduced forms and the Lebesgue function

In this section we show how to write extended interpolants in barycentric form and introduce another way to describe them, which we call reduced form. This form is numerically unstable and we do not advocate its use in practice. Its purpose is to
allow us to deduce an expression for the Lebesgue function of extended interpolants and expose their backward instability.

In our derivation of an expression for the Lebesgue function, we emphasize that the input to the interpolation process are the original function values \( y_i \) and not the extrapolated function values \( \tilde{y}_i \). The Lebesgue function measures the sensitivity of the output of the complete interpolation process to perturbations in its input. Perturbations in the \( y_i \) affect the \( \tilde{y}_i \) significantly, the Lebesgue function takes into account the dependency of \( \tilde{y} \) on \( y \), and \( \tilde{y} \) grows exponentially with \( d = n = \tilde{n} = \tilde{d} \) when \( y = (1, 1, -1, 1, -1, 1, -1, 1, -1, 1, \ldots) \).

We recall that extended Floater-Hormann interpolants are defined only for equally spaced nodes, and in this case \([5]\) shows that usual Floater-Hormann interpolants can be written in the barycentric form

\[
r_{\mathbf{x}, d}[y](t) = \sum_{i=0}^{n} \frac{w_{n, d, i} y_i}{t - x_i} / \sum_{i=0}^{n} w_{n, d, i} t - x_i, \tag{3}
\]

with weights

\[
w_{n, d, i} = (-1)^{i-d} \min\{n-d, i\} \sum_{j=\max\{0, i-d\}}^{\min\{n-d, i\}} \binom{i}{d} i - j,
\]

where the \( y_i \) are the interpolated function values. Extended interpolants are defined by extrapolating the \( y_i \) according to the Taylor series

\[
\tilde{y}_i := y_0 + \sum_{k=1}^{d} \tilde{y}_{\mathbf{x}, d}^{(k)}(\tilde{x}_i - x_0)^k / k! \quad \text{for } -d \leq i < 0, \tag{4}
\]

\[
\tilde{y}_i := y_i \quad \text{for } 0 \leq i \leq n, \tag{5}
\]

\[
\tilde{y}_i := y_n + \sum_{k=1}^{d} \tilde{y}_{\mathbf{x}, d}^{(k)}(\tilde{x}_i - x_n)^k / k! \quad \text{for } n < i \leq n + d, \tag{6}
\]

where \( \tilde{x}_i = x_0 + ih \) for \(-d \leq i \leq n + d\) and \( \tilde{y}_{\mathbf{x}, d}^{(k)}(\tilde{x}_i)(t) \) is the \( k \)th derivative of the Floater-Hormann interpolant \( r_{\mathbf{x}, d}[y](t) \) in \([3]\) and \( \mathbf{x} := (x_0, \ldots, x_n), y := (y_0, \ldots, y_n), \tilde{x} := (x_0 - \tilde{n}, \ldots, x_n) \) and \( \tilde{y} := (y_0 - \tilde{n}, \ldots, y_n) \). We then have the following barycentric form for extended interpolants:

\[
\tilde{r}_{\mathbf{x}, d, \tilde{n}, \tilde{d}}[\tilde{y}](t) = \sum_{i=-d}^{n+d} \tilde{w}_{n, d, i} \tilde{y}_i / \sum_{i=-d}^{n+d} \tilde{w}_{n, d, i} t - \tilde{x}_i, \tag{7}
\]

with weights

\[
\tilde{w}_{n, d, i} := \tilde{w}_{n+2d, d, i+d} = (-1)^{i+d} \min\{n+d, i+d\} \sum_{j=\max\{0, i\}}^{\min\{n+d, i+d\}} \binom{i}{d} i - j.
\]

It is difficult to derive the Lebesgue function of the extended interpolant \( \tilde{r}_{\mathbf{x}, d, \tilde{n}, \tilde{d}} \) directly from equation \((7)\), because this equation depends on \( \tilde{y} \), which is not part of the original interpolation problem. To derive an expression for this Lebesgue function, it is helpful to write the extended interpolant only in terms of the original \( y \). The next lemma explains how to achieve this goal when \( 2\tilde{n} < n \):
Lemma 1  The extrapolated function values \( \tilde{y}_i \) used to define the extended interpolant \( \tilde{r}_{x,d,n,d} \) can be written as

\[
\tilde{y}_i = \sum_{j=0}^{\tilde{n}} a_{ij} y_j, \quad \text{for } -d \leq i < 0, \tag{8}
\]

\[
\tilde{y}_i = \sum_{j=n-\tilde{n}}^{n} b_{(i-n)(j-n)} y_j, \quad \text{for } n \leq i \leq n + d, \tag{9}
\]

where the constants \( \{ a_{ij}, -d \leq i < 0, 0 \leq j \leq \tilde{n} \} \) and \( \{ b_{ij}, 0 < i \leq d, -\tilde{n} \leq j \leq 0 \} \) depend on \( i \) and \( j \) but do not depend on \( h \) or \( n \). When \( 2\tilde{n} < n \) the extended interpolant can be written in the reduced form

\[
\tilde{r}_{x,d,n,d}[y](t) = \sum_{j=0}^{n} c_j(t) y_j / \sum_{j=-d}^{n+d} \tilde{w}_{n,d,j} t - x_j, \tag{10}
\]

where the functions \( c_j \) are given by

\[
c_j(t) = \tilde{w}_{n,d,j} t - x_j + \sum_{i=-d}^{n} \tilde{w}_{n,d,i} a_{ij} \quad \text{for } 0 \leq j \leq \tilde{n}, \tag{11}
\]

\[
c_j(t) = \tilde{w}_{n,d,j} t - x_j \quad \text{for } \tilde{n} < j < n - \tilde{n}, \tag{12}
\]

\[
c_j(t) = \tilde{w}_{n,d,j} t - x_j + \sum_{i=n+1}^{n+d} \tilde{w}_{n,d,i} b_{(i-n)(j-n)} \quad \text{for } n - \tilde{n} \leq j \leq n. \tag{13}
\]

In the end of this section we prove Lemma 1 and present explicit expressions for \( a_{ij} \) and \( b_{ij} \). We also provide formulae analogous to (11)–(13) for the case \( 2\tilde{n} \geq n \).

The Lebesgue functions of the interpolants \( r_{x,d} \) and \( \tilde{r}_{x,d,n,d} \) are defined as

\[
A_{x,d}(t) := \sup_{\|y\|_{\infty} = 1} |r_{x,d}[y](t)| \quad \text{and} \quad \tilde{A}_{x,d,n,d}(t) := \sup_{\|y\|_{\infty} = 1} |\tilde{r}_{x,d,n,d}[y](t)|,
\]

and using equations (11) and the reduced form (10) it is easy to show that

\[
A_{x,d}(t) = \left\| \sum_{j=0}^{n} \tilde{w}_{n,d,j} t - x_j \right\| \left\| \sum_{j=0}^{n} \tilde{w}_{n,d,j} t - x_j \right\| \tag{14}
\]

and

\[
\tilde{A}_{x,d,n,d}(t) = \left\| \sum_{j=0}^{n} c_j(t) \right\| \left\| \sum_{j=-d}^{n+d} \tilde{w}_{n,d,j} t - x_j \right\|. \tag{15}
\]

We emphasize that, in general,

\[
\tilde{A}_{x,d,n,d}(t) \neq \left\| \sum_{j=-d}^{n+d} \tilde{w}_{n,d,j} t - x_j \right\| \left\| \sum_{j=-d}^{n+d} \tilde{w}_{n,d,j} t - x_j \right\|. \tag{16}
\]
that is, we can deduce (14) from (3), but \( \tilde{\Lambda}_{x,d,\tilde{n},\tilde{d}}(t) \) is not equal to, or even bounded by, the right hand side of (16). This is why equation [11]'s equation (3.2) is misleading. The fact that it refers to [11]'s peculiar “interpretation” of the Lebesgue constant, and not to the actual Lebesgue constant, becomes evident when we plot the right and the left hand side of (16) for \( n = 50, \ d = 3, \ \tilde{n} = 11 \) and \( \tilde{d} = 7 \), as in Figure 2 below.

![Figure 2](image)

Fig. 2: The correct Lebesgue function for \( n = 50, \ d = 3, \ \tilde{n} = 11 \) and \( \tilde{d} = 7 \).

The dependency of the Lebesgue function on the parameters \( d, \tilde{n} \) and \( \tilde{d} \) is subtle. For instance, Figure 3 shows that the Lebesgue function may decrease as we increase \( d \) and keep the other parameters fixed. Moreover, the Lebesgue constant for \( n = 50, \ d = \tilde{n} = \tilde{d} = 7 \) is 12 times smaller than the Lebesgue constant if \( n = 50, \ d = 3, \ \tilde{n} = 11 \) and \( \tilde{d} = 7 \), as considered in [11]. Given this, one could ask why should we choose \( d = 3, \ \tilde{n} = 11 \) and \( \tilde{d} = 7 \) instead of \( n = \tilde{n} = \tilde{d} = 7 \).

![Figure 3](image)

Fig. 3: The dependency of the Lebesgue function on \( d, \tilde{n} \) and \( \tilde{d} \).
3.1 Proof of Lemma 1

In this subsection we prove Lemma 1 and show that when \( n \leq 2\tilde{n} \) we have an analogous result with

\[
\begin{align*}
\mathcal{C}_j(t) &= \tilde{w}_{n,d,j}(t-x_j) + \sum_{i=-d}^{n} \tilde{w}_{n,d,i}(t-x_i) \quad \text{for } 0 \leq j < n - \tilde{n}, \quad (17) \\
\mathcal{C}_j(t) &= \tilde{w}_{n,d,j}(t-x_j) + \sum_{i=0}^{d-1} \tilde{w}_{n,d,i}(t-x_i) + \sum_{i=n+1}^{n+d} \tilde{w}_{n,d,i}(t-x_i) \quad \text{for } n - \tilde{n} \leq j \leq \tilde{n}, \quad (18) \\
\mathcal{C}_j(t) &= \frac{w_j}{x-x_j} + \sum_{i=n+1}^{n+d} \frac{\tilde{w}_{n,d,i}(t-x_i)}{x-x_i} \quad \text{for } \tilde{n} < j \leq n. \quad (19)
\end{align*}
\]

Our proof involves the constants \( D_{ij}^{(k)} \) mentioned in the third section of [12], but here we extend them to \( k = 0 \), defining

\[
D_{ii}^{(0)} := 1 \quad \text{and} \quad D_{ij}^{(0)} := 0 \quad \text{for } i \neq j. \quad (21)
\]

It follows that, for \( k \geq 1 \), the \( D_{ij}^{(k)} \) in [12] satisfy

\[
\begin{align*}
D_{ij}^{(k)} &= \frac{k}{x_i - x_j} \left( \frac{w_j}{w_i} D_{ii}^{(k-1)} - D_{ij}^{(k-1)} \right) \quad \text{for } i \neq j, \quad (22) \\
D_{ii}^{(k)} &= \sum_{j=0, j \neq i}^{n} D_{ij}^{(k)}. \quad (23)
\end{align*}
\]

When the nodes are equally spaced, \( x_i - x_j = (i - j)h \) and it is convenient to consider the normalized constants

\[
\overline{D}_{ij}^{(k)} := h^k D_{ij}^{(k)}/k!. \]

Replacing \( D_{ij}^{(k)} \) by \( \overline{D}_{ij}^{(k)} k!h^{-k} \) in (21)–(23) we obtain

\[
\begin{align*}
\overline{D}_{ii}^{(0)} &= 1 \quad \text{and} \quad \overline{D}_{ii}^{(0)} = 0 \quad \text{for } i \neq j, \\
\overline{D}_{ij}^{(k)} &= \frac{1}{i-j} \left( \frac{w_j}{w_i} \overline{D}_{ii}^{(k-1)} - \overline{D}_{ij}^{(k-1)} \right) \quad \text{for } i \neq j \text{ and } k \geq 1, \\
\overline{D}_{ii}^{(k)} &= \sum_{j=0, j \neq i}^{n} \overline{D}_{ij}^{(k)}.
\end{align*}
\]

Using these expressions we can compute \( \overline{D}_{ij}^{(k)} \) for all \( i, j \) and \( k \), and it is clear that \( \overline{D}_{ij}^{(k)} \) depends on \( i \) and \( j \) but does not depend on \( h, n \) or \( d \).
The arguments in the third section of [12] lead to

\[ r^{(k)}_{\tilde{x}}(y)(x_0) = \sum_{j=0}^n D^{(k)}_{0j} y_j = \frac{k!}{h^k} \sum_{j=0}^n D^{(k)}_{0j} y_j, \]

\[ r^{(k)}_{\tilde{x}}(y)(x_n) = \sum_{j=n-n+\tilde{n}}^n D^{(k)}_{n(j-n+\tilde{n})} y_j = \frac{k!}{h^k} \sum_{j=n-n+\tilde{n}}^n D^{(k)}_{n(j-n+\tilde{n})} y_j, \]

and we can rewrite (1) and (3) as

\[ \tilde{y}_i = \sum_{k=0}^d \frac{\tilde{n}}{k!} D^{(k)}_{0j} t^k y_j \quad \text{for } -d \leq i < 0, \]

\[ \tilde{y}_i = \sum_{k=0}^d \frac{n}{k!} D^{(k)}_{n(j-n+\tilde{n})} (i-n)^k y_j \quad \text{for } n < i \leq n+d. \]

These equations are equivalent to (8) and (9) with

\[ a_{ij} := \sum_{k=0}^d D^{(k)}_{0j} t^k \quad \text{for } -d \leq i < 0 \quad \text{and} \quad 0 \leq j \leq \tilde{n}, \quad \text{(24)} \]

\[ b_{ij} := \sum_{k=0}^d D^{(k)}_{n(j-n+\tilde{n})} t^k \quad \text{for } 0 < i \leq d \quad \text{and} \quad -\tilde{n} \leq j \leq 0. \quad \text{(25)} \]

These \( a_{ij} \) and \( b_{ij} \) do not depend on \( h \) or \( n \), as required by Lemma 1.

Equation (4) shows that

\[ \tilde{r}_{\tilde{x},d,\tilde{n},d}[y](t) = \frac{1}{Q(t)} \sum_{i=-d}^{n+d} \tilde{w}_{n,d,i} \tilde{y}_i \quad \text{with} \quad Q(t) = \sum_{i=-d}^{n+d} \tilde{w}_{n,d,i}. \]

It follows that

\[ Q(t) \tilde{r}_{\tilde{x},d,\tilde{n},d}[y](t) = -\sum_{i=-d}^{n-d} \tilde{w}_{n,d,i} \frac{\tilde{y}_i}{t-x_i} \sum_{j=0}^{\tilde{n}} a_{ij} y_j + \sum_{j=0}^{n} \tilde{w}_{n,d,j} \frac{y_j}{t-x_j} + \]

\[ \sum_{i=-\tilde{n}}^{-1} \tilde{w}_{n,d,i} \frac{y_i}{t-x_i} - \sum_{j=n}^{n+d} \tilde{w}_{n,d,j} \frac{y_j}{t-x_j} + \]

\[ \sum_{j=n-n+\tilde{n}}^{n-\tilde{n}} \frac{\tilde{w}_{n,d,j} (i-n)(j-n) y_j}{t-x_j} + \sum_{j=n-n+\tilde{n}}^{n-\tilde{n}} \frac{\tilde{w}_{n,d,j} (i-n)(j-n) y_j}{t-x_j}. \quad \text{(26)} \]

We now have two cases: (i) \( \tilde{n} < n - \tilde{n} \) and (ii) \( \tilde{n} \geq n - \tilde{n} \). In the first case we can rewrite (26) as

\[ Q(t) \tilde{r}_{\tilde{x},d,\tilde{n},d}[y](t) = \sum_{j=0}^{\tilde{n}} \tilde{w}_{n,d,j} \frac{y_j}{t-x_j} + \sum_{i=-d}^{-1} \tilde{w}_{n,d,i} a_{ij} y_j + \]

\[ \sum_{j=n-n+\tilde{n}}^{n-\tilde{n}} \frac{\tilde{w}_{n,d,j} (i-n)(j-n) y_j}{t-x_j} + \sum_{j=n-n+\tilde{n}}^{n-\tilde{n}} \frac{\tilde{w}_{n,d,j} (i-n)(j-n) y_j}{t-x_j}. \]
and this proves (10)–(13). In the second case, when \( \tilde{p} \), we can rewrite (26) as

\[
Q(t) \hat{r}_{x,d,\tilde{n},\tilde{d}}(y)(t) = \sum_{j=0}^{n-\tilde{n}-1} \left( \frac{\tilde{w}_{n,d,i} t - x_j}{t - x_i} \right) y_j + \sum_{j=\tilde{n}}^{\tilde{n}-1} \left( \frac{\tilde{w}_{n,d,i} t - x_j}{t - x_i} \right) y_j + \sum_{j=\tilde{n}}^{n} \left( \frac{\tilde{w}_{n,d,i} t - x_j}{t - x_i} \right) y_j.
\]

This proves (17)–(20) and we are done.

4 The Lebesgue constant when \( d = \tilde{n} = \tilde{d} \)

When \( \tilde{n} = \tilde{d} = d \), the analysis of the extended interpolant is simplified, because the \( \tilde{y}_i \) are obtained by polynomial extrapolation, and the same polynomials used for extrapolation are used to express the resulting interpolant. In this case we have

\[
\tilde{y}_i = p_0(x_i, x, y) \quad \text{for} \quad -d \leq i < 0, \quad (27)
\]

\[
\tilde{y}_i = p_{\tilde{n}-d}(x_i, x, y) \quad \text{for} \quad n < i \leq \tilde{n} + d, \quad (28)
\]

where \( p_i(t, x, y) \) is the polynomial with degree less than \( d + 1 \) that interpolates \( y_k \) at \( x_k \) for \( k = i, i + 1, \ldots, i + d \).

The Lebesgue constant of the extended interpolant \( \hat{r}_{x,d,\tilde{n},\tilde{d}} \) is

\[
\hat{A}_{x,d,\tilde{n},\tilde{d}} := \max_{x_0 \leq t \leq x_n} \hat{A}_{x,d,\tilde{n},\tilde{d}}(t),
\]

for the Lebesgue function \( \hat{A}_{x,d,\tilde{n},\tilde{d}}(t) \) in (15), and the Lebesgue constant for polynomial interpolation at \( d + 1 \) equally spaced points is

\[
A_d := \sup_{0 \leq t \leq d} \frac{|p_{d,y}(t)|}{y_\infty},
\]

where \( p_{d,y} \) is the polynomial with degree less than \( d + 1 \) such that \( p_{d,y}(i) = y_i \) for \( i = 0, \ldots, d \). In this section we show that \( \hat{A}_{x,d,\tilde{n},\tilde{d}} \) is not much smaller than \( A_d \).

Formally, we have the following theorem:

**Theorem 1** If \( n > d \geq 4 \) then

\[
\hat{A}_{x,d,\tilde{n},\tilde{d}} \geq \left( 1 - \frac{2}{d-1} - \frac{2}{d(d-1)} \right) A_d.
\]
We prove Theorem 1 at the end of this section. For now, let us explore its consequences and verify them experimentally. As explained in [17],
\[
\frac{2^{d-2}}{d^2} < A_d < \frac{2^{d+3}}{d}.
\]
Therefore, \(A_d\) grows exponentially with \(d\) and Theorem 1 shows that the same applies to \(\tilde{A}_{x,d,d,d}\). Moreover, [2] shows that the Lebesgue constant for the Floater-Hormann interpolants satisfies
\[
\tilde{A}_{x,d} \leq 2^{d-1} (2 + \log n).
\]
Combining the equations above for \(d \geq 6\) we conclude that
\[
\tilde{A}_{x,d} \leq 4d^2 (2 + \log n) \tilde{A}_{x,d,n,d},
\]
and the ratio \(\tilde{A}_{x,d}/\tilde{A}_{x,d,n,d}\) is definitely not as large as claimed in [11]. This observation is corroborated by Figure 4.

Fig. 4: log10 of the Lebesgue constants for \(n = 200\) and \(d = \tilde{n} = \tilde{d}\) varying from 1 to 35. The Lebesgue constant of the extended interpolant is about the same as the Lebesgue constant for polynomial interpolation at \((d+1)\) equally spaced nodes for \(d \geq 7\), and \(\tilde{A}_{x,d}\) is roughly equal to \(100\tilde{A}_{x,d,n,d}\) for large \(d = \tilde{n} = \tilde{d}\).

4.1 Proof of Theorem 1

According to [11],
\[
\tilde{r}_{x,d,n,d}(y)(t) = \frac{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x}) p_i(t, \tilde{x}, \tilde{y})}{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x})},
\]
where \(\tilde{y} = (y_{-d}, y_1, \ldots, y_{n+d})\),
\[
\lambda_i(t, x) := \frac{(-1)^i}{(t-x_i) \cdots (t-x_{i+d})}, \quad i = -d, 1 - d, \ldots, n,
\]
and \( p_i(t, x, y) \) is the polynomial with degree less than \( d + 1 \) such that \( p_i(x_k, x, y) = y_k \) for \( k = i, \ldots, i + d \). We always have

\[
p_i(t, \tilde{x}, \tilde{y}) = p_i(t, x, y) \quad \text{for} \quad 0 \leq i \leq n - d,
\]

and when \( d = \tilde{d} = d \) we also have

\[
p_i(t, \tilde{x}, \tilde{y}) = p_0(t, x, y) \quad \text{for} \quad -d \leq i < 0,
\]

\[
p_i(t, \tilde{x}, \tilde{y}) = p_{n-d}(t, x, y) \quad \text{for} \quad n - d < i \leq n,
\]

because in this case the interpolants \( r_{x, d}[y](x_0) \) and \( r_{x, d}[y](x_n) \) are polynomials, and the Taylor series of a polynomial is equal to itself. It follows that

\[
\tilde{r}_{x, d, d, d}[y](t) = \frac{\sum_{i=-d}^{0} \lambda_i(t, \tilde{x}) \lambda_0(t, x, y) p_0(t, x, y)}{\lambda_0(t, x, y)} + \frac{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x})}{\lambda_0(t, x, y)} \sum_{i=-d}^{n} \lambda_i(t, \tilde{x}) p_{-d}(t, x, y).
\]

(35)

Let \( t \in (x_0, x_1) \) be fixed. We claim that \( y_0^*, y_1^*, \ldots, y_n^* \in \{−1, 1\} \) defined by

\[
y_0^* := y_1^* := (-1)^d \quad \text{and} \quad y_j^* := (-1)^{d+j-1} \text{ for } 2 \leq j \leq n
\]

satisfy

\[
\lambda_i(t, x) p_i(t, x, y^*) = (-1)^i \sum_{j=0}^{d} \frac{|u_j|}{|t - x_{i+j}|} \quad \text{for } 0 \leq i \leq n - d,
\]

(36)

where

\[
u_j := \frac{(-1)^{d-j}}{d! h^d} \binom{d}{j}.
\]

In fact, [6] shows that

\[
\lambda_i(t, x) p_i(t, x, y^*) = (-1)^i \sum_{j=0}^{d} \frac{u_j y_{i+j}^*}{t - x_{i+j}},
\]

and (36) follows from

\[
\frac{u_0 y_0^*}{t - x_0} = \frac{|u_0|}{|t - x_0|}, \quad \frac{u_1 y_1^*}{t - x_1} = \frac{|u_1|}{|t - x_1|}
\]

and

\[
\frac{u_j y_{i+j}^*}{t - x_{i+j}} = (-1)^{d-j} |u_j| |y_{i+j}^*| = (-1)^{-1} |u_j| \frac{|u_j|}{|t - x_{i+j}|}
\]

for \( 2 \leq i + j \leq n \).

Note that

\[
\frac{\sum_{i=-d}^{0} \lambda_i(t, \tilde{x})}{\lambda_0(t, x, y)} > 0 \quad \text{and} \quad 1 > \frac{\sum_{i=n-d}^{n} \lambda_i(t, \tilde{x})}{\lambda_{n-d}(t, x, y)} > 0,
\]

(37)

because the numbers \( \lambda_{-d}(t, \tilde{x}), \lambda_{-d}(t, \tilde{x}), \ldots, \lambda_0(t, \tilde{x}) = \lambda_0(t, x) \) have the same sign and the numbers \( \lambda_{n-d}(t, x) = \lambda_{n-d}(t, \tilde{x}), \lambda_{n-d+1}(t, \tilde{x}), \ldots, \lambda_n(t, \tilde{x}) \) decrease
in magnitude and their signs alternate. The signs of the terms \(\lambda_i(t, x) p_i(t, x, y^*)\) in (36) also alternate and the magnitude of these terms decreases as the index \(i \geq 1\) increases. Combining these facts, (35) and (37), we obtain

\[
|\tilde{r}_{x, d, d, d}[y^*](t)| \geq \frac{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x}) |\lambda_0(t, x)p_0(t, x, y^*) + \lambda_1(t, x)p_1(t, x, y^*)|}{|\sum_{i=-d}^{n} \lambda_i(t, \tilde{x})|}. \tag{38}
\]

Moreover, (36) yields

\[
\lambda_0(t, x)p_0(t, x, y^*) + \lambda_1(t, x)p_1(t, x, y^*) =
\]

\[
\frac{|u_0|}{|t - x_0|} + \left\{ \frac{|u_1| - |u_0|}{|t - x_1|} - \frac{|u_1|}{|t - x_1|} \right\} + \sum_{j=2}^{d} \left( \frac{|u_j|}{|t - x_j|} - \frac{|u_j|}{|t - x_j+1|} \right). \tag{39}
\]

The last sum in (39) is positive for all \(t \in (x_0, x_1)\). The term in brackets in also positive for \(t \in (x_0, x_1)\), because

\[
\frac{|u_1| - |u_0|}{|u_1|} = 1 - \frac{1}{d} > \frac{1}{2} \geq \frac{|t - x_1|}{|t - x_2|}.
\]

Since all the numbers \(\lambda_{-d}(t, \tilde{x}), \lambda_{-d}(t, \tilde{x}), \ldots, \lambda_0(t, \tilde{x}), \lambda_1(t, \tilde{x})\) have the same sign and the signs of the numbers \(\lambda_1(t, \tilde{x}), \lambda_2(t, \tilde{x}), \ldots, \lambda_n(t, \tilde{x})\) alternate, and their magnitude decreases, (38) leads to

\[
|\tilde{r}_{x, d, d, d}[y^*](t)| \geq \frac{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x}) |p_0(t, x, y^*)|}{\sum_{i=-d}^{n} \lambda_i(t, \tilde{x})}. \tag{40}
\]

We also have

\[
\frac{|\lambda_0(t, \tilde{x})|}{|\lambda_{-1}(t, \tilde{x})|} = \frac{|t - x_{-1}|}{|t - x_d|} \leq \frac{2h}{(d - 1) h} = \frac{2}{d - 1}
\]

and

\[
\frac{|\lambda_1(t, \tilde{x})|}{|\lambda_{-1}(t, \tilde{x})|} = \frac{|\lambda_0(t, \tilde{x})|}{|\lambda_{-1}(t, \tilde{x})|} \frac{|\lambda_1(t, \tilde{x})|}{|\lambda_0(t, \tilde{x})|} \leq \frac{2}{d - 1} \frac{|t - x_0|}{|t - x_{d+1}|} \leq \frac{2h}{(d - 1) dh} = \frac{2}{d(d - 1)}.
\]

Therefore,

\[
\left| \sum_{i=-d}^{n} \lambda_i(t, \tilde{x}) \right| \geq \left| \sum_{i=-d}^{1} \lambda_i(t, \tilde{x}) \right| - |\lambda_0(t, \tilde{x})| - |\lambda_1(t, \tilde{x})| \geq
\]

\[
\left| \sum_{i=-d}^{1} \lambda_i(t, \tilde{x}) \right| \left( 1 - \frac{2}{d - 1} - \frac{2}{d(d - 1)} \right). \tag{41}
\]

Bounds (40) and (41) yield

\[
|\tilde{r}_{x, d, d, d}[y^*](t)| \geq \left( 1 - \frac{2}{d - 1} - \frac{2}{d(d - 1)} \right) |p_0(t, x, y^*)|.
\]

Equation (36) shows that, for \(t \in [x_0, x_1]\), \(|p_0(t, x, y^*)|\) is identical to the Lebesgue function for polynomial interpolation at \((d + 1)\) equally spaced nodes in \([x_0, x_d]\).
According to [3], the Lebesgue function for polynomial interpolation at equally spaced nodes attains its maximum at some \( t^* \in (x_0, x_1) \). For this \( t^* \) we have

\[
\tilde{\Lambda}_{x,d,d,d} \geq \tilde{\Lambda}_{x,d,d,d}(t^*) \geq \left| \tilde{r}_{x,d,d,d}[y^*](t^*) \right| \geq \left( 1 - \frac{2}{d-1} - \frac{2}{d(d-1)} \right) \Lambda_d,
\]

and this completes the proof of Theorem 1.

5 Backward instability

Extended interpolants are not backward stable because the coefficients \( c_j(t) \) that define the reduced form (10) may be equal to zero for some \( j \) and \( t \). For instance, Figure 5 displays the function \( c_2(t) \) in the interval \([-0.918, -0.914]\), for the case \( n = 50, d = 3, \tilde{n} = 11 \) and \( \tilde{d} = 7 \) considered by [11]. When \( c_j(t) = 0 \) and we define the extended interpolant by taking \( y_i = 0 \) for \( i \neq j \) and \( y_j = 1 \), the computed value \( \hat{r}(\tilde{x}, \tilde{n}, \tilde{d}, y(t)) \) will most likely be different from zero — unless there is exact cancellation in the evaluation of \( c_j(t) \) — and there is no \( \hat{y} \) with \( \hat{y}_k = (1 + \delta_k) y_k \) and \( |\delta_k| < 1 \) such that \( \hat{r}(\tilde{x}, \tilde{n}, \tilde{d}, \hat{y}(t)) = \tilde{r}(\tilde{x}, \tilde{n}, \tilde{d}, y(t)) \). Therefore, this extended interpolant is not backward stable.

6 Sources of numerical instability for extended interpolants

This section shows that extended interpolants are affected by extrapolation errors when \( \tilde{d} \) is large. The current literature pays little attention to this point and presents experimental comparisons of usual and extended interpolants that highlight cases in which \( \tilde{d} \) is much smaller than \( d \). Such experiments are biased in favor of extended interpolants: increasing \( d \) for usual interpolants makes as much sense as increasing \( \tilde{d} \) for extended interpolants, because for \( d > \tilde{d} \) the order of approximation of the extended interpolants is \( h^{\tilde{d}+1} \), and not \( h^{d+1} \).

Fig. 5: The function \( c_2(t) \) for \( n = 50, d = 3, \tilde{n} = 11 \) and \( \tilde{d} = 7 \).
Figure 6 illustrates the importance of choosing appropriate $\tilde{d}$ for extended interpolants. It also shows that it is pointless to increase $d$ when $d > \tilde{d}$. Large values of $d$ have a devastating effect on usual Floater-Hormann interpolants, and this basic fact is mentioned explicitly in the documentation of libraries that implement these interpolants [1]. Consequently, there is little to be learned from comparisons of extended and usual Floater-Hormann interpolants which fix $\tilde{d}$ at convenient values for extended interpolants and then raise $d$ to large values. Such choices of $d$ and $\tilde{d}$ have no practical motivation for extended interpolants and are notoriously unfavorable to usual Floater-Hormann interpolants.

Fig. 6: Log10 of the error for $f(t) = \sin(20t)$ and $t \in [-1,1]$, with $n = 200$, $\tilde{n} = \tilde{d} = 8$ and $d$ varying from 1 to 35. Note that by simply increasing $d$, with an inappropriate $\tilde{d}$, we may obtain inaccurate results for extended interpolants. In this example, increasing $d > \tilde{d} = 8$ has no effect on the accuracy of the interpolants.

In this section we consider the case $d = \tilde{d} = \tilde{n}$ because it is the minimal one resulting in an approximation of order $h^d$ for extended interpolants. Our experiments lead us to believe that this is the most reasonable choice in general, and should be the first one to be considered in practice. We note, however, that we will gladly change our point view in the face of objective data showing that a better choice is available, especially if this choice does not require hours of tweaking and searching for the parameters $d$, $\tilde{n}$ and $\tilde{d}$ for each particular function we want to interpolate.

As in [10], here we consider the interpolation of $f(t) = \sin(20t)$ for $t \in [-1,1]$ with $n = 200$. We start with Figure 7 which compares usual Floater-Hormann interpolants, extended interpolants with $\tilde{y}_i$ computed in double precision and extended interpolants with precise $\tilde{y}_i$. By precise we mean that these $\tilde{y}_i$ were computed using the MPFR library [16], with floating point numbers with a mantissa of 320 bits, from $y_i$ computed with the same high precision.

By error in our plots we mean the maximum difference between the numerically evaluated interpolant and the original function at $10^7$ equally spaced sample points in the interval $[-1,1]$. For each interpolant, its barycentric formula ((3) or (7)) was evaluated in double precision ($\epsilon \approx 10^{-16}$), using straightforward C++ code. The $y_i$ and $\tilde{y}_i$ computed using multiple precision were rounded to double precision and the barycentric formula corresponding to them was also evaluated in double
precision. In other words, the case precise \( \tilde{y}_i \) differs from the other cases only by the precision of the \( \tilde{y}_i \), and not by the precision used to evaluate the barycentric formulae (3) and (7).

Figure 7 shows that, when the \( \tilde{y}_i \) are evaluated in double precision, extended interpolants are not significantly more accurate than usual ones, and their accuracy deteriorates sharply as \( d = \tilde{d} = \tilde{n} \) grows. By contrast, extended interpolants with precise \( \tilde{y}_i \) are remarkably accurate, even for large values of \( d = \tilde{d} = \tilde{n} \). This is strong evidence that the inaccuracy of the \( \tilde{y}_i \) is the cause of the low accuracy of the extended interpolants for large \( d = \tilde{d} = \tilde{n} \).

The combination of Figures 4 and 7 leads to a surprising observation: according to Figure 4, the Lebesgue constant of the usual interpolants in our experiments is about 100 times larger than the Lebesgue constant of the corresponding extended interpolants for large \( d = \tilde{d} = \tilde{n} \), yet Figure 7 shows that the usual interpolants are much more accurate for such \( d, \tilde{n} \) and \( \tilde{d} \). The data in Figures 4 and 7 are combined in Figure 8.

Figure 8: Log10 of (error divided by the Lebesgue constant) for \( f(t) = \sin(20t) \), \( t \in [-1, 1] \) and \( n = 200 \).

Figure 8 highlights three important points for \( d = \tilde{n} = \tilde{d} \geq 15 \):
The numerical stability of extended Floater Hormann interpolants

- For fixed $n$, the error incurred by usual Floater-Hormann interpolants is of order $\epsilon A_{x,d}$.
- For fixed $n$, the error incurred by extended interpolants grows faster than $\epsilon A_{x,d,\tilde{n},\tilde{d}}$.
- The effects of the large Lebesgue constant are reduced significantly when the $\tilde{y}_i$ are precise. In this case, numerical errors occur mostly in the evaluation of the barycentric formula, and the argument following [11]'s equation (3.2) applies and explains the errors of order $\epsilon$ for the extended interpolants with precise $\tilde{y}_i$ in Figure 7.

The data for the extended interpolants with double precision $\tilde{y}_i$ in Figure 8 for $d \geq 30$ indicate the existence of another source of numerical instability for these interpolants, in addition to the large Lebesgue constants. This extra source of instability are the enormous entries of the matrices $a_{ij}$ and $b_{ij}$ in Lemma 1, which are defined explicitly in (24)–(25). In fact, Figure 9 shows that the $a_{ij}$ and $b_{ij}$ grow exponentially with $d = \tilde{d} = \tilde{n}$.

![Log10:max(a_ij)](image)

Fig. 9: $\log_{10}(\max |a_{ij}|) = \log_{10}(\max |b_{ij}|)$ for $n = 200$.

In view of the remarkable accuracy of the extended interpolants with precise $\tilde{y}_i$, it makes sense to consider the possibility of using $\tilde{y}_i$ evaluated in multiple precision from the double precision $y_i$ which are usually available. These $\tilde{y}_i$ are not as precise as the ones obtained from high precision $y_i$ using multiple precision arithmetic. Figure 10 illustrates, however, that this strategy improves the accuracy of the extended interpolants, at a relatively low cost when $d$, $\tilde{d}$ and $\tilde{n}$ are small compared to $n$ and we want to evaluate the interpolants for many values of $t$.

In the case considered in this section, Figure 11 shows that the $\tilde{y}_i$ evaluated using multiple precision, from double precision $y_i$, lead to overall numerical errors of order $\epsilon A_{x,d,\tilde{n},\tilde{d}}$, which are about 100 times smaller than the errors incurred by the usual Floater-Hormann interpolants in our experiments for large $d = \tilde{n} = \tilde{d}$.

7 The importance of the cancellation of rounding errors

Here we show that the accuracy of extended interpolants is sensitive to the way we implement the extrapolation step. We also show that accurate implementations of
Fig. 10: Log10 of the error for $f(t) = \sin(20t)$, $t \in [-1, 1]$ and $n = 200$.

Fig. 11: Log10 of (error divided by the Lebesgue constant) for $f(t) = \sin(20t)$, $t \in [-1, 1]$ and $n = 200$.

this step depend on the cancellation of the rounding errors in order to achieve their accuracy. In fact, the errors incurred by extended interpolants can be enormous when we use the extrapolation formula proposed in [11] and restated in (4)–(6), and compute the extrapolated $\tilde{y}_i$ according to the following procedure:

(a) If $i$ is even, set the rounding mode upward and evaluate $\tilde{y}_i$ as in (4)–(6).
(b) If $i$ is odd, set the rounding mode downward and evaluate $\tilde{y}_i$ as in (4)–(6).

In this scenario the overall effect of rounding errors can be much larger than what one would expect from the already large Lebesgue constants, as illustrated in Figures 12 and 13. In the plots corresponding to $\tilde{y}_i$ evaluated as in (8)–(9) in these figures the $\tilde{y}_i$ were obtained by matrix multiplication, with $a_{ij}$ and $b_{ij}$ computed in multiple precision and then rounded to double precision i.e., with $a_{ij}$ and $b_{ij}$ as accurate as possible.

We emphasize that the choices of rounding modes in the computation steps (a) and (b) above are not frivolous. Their purpose is to help us understand what can be proved about the numerical stability of extended interpolants, so that we do not try to prove something that cannot be proved. It is unlikely that the $\tilde{y}_i$ will be evaluated as in the steps (a) and (b) when rounding to nearest. In this mode, there is a 50% chance of rounding up in each flop and, under the questionable
Fig. 12: Extended Floater-Hormann. \( \log_{10} \) of error for \( f(t) = \sin(20t) \) for \( t \in [-1, 1] \) and \( n = 200 \).

Fig. 13: Extended Floater-Hormann. \( \log_{10} \) of (error divided by the Lebesgue constant) for \( f(t) = \sin(20t) \), \( t \in [-1, 1] \) and \( n = 200 \).

The hypothesis of independence of the rounding errors, there would be a minuscule probability of \( 2^{-2(d+1)d} \) of having all the intermediate results in the evaluation of \( \tilde{y}_i \), rounded up when the rounding mode is set to nearest. Since in reality the set of floating point numbers is finite, such a coincidence may be impossible. However, our experiments indicate that it is difficult to build a realistic theory on the effects of rounding errors on extended interpolants, because the rounding errors induced by our changes of rounding modes would be allowed under the usual models of floating point arithmetic, with \( \epsilon \) replaced by \( 2\epsilon \). More precisely: when evaluating the \( \tilde{y}_i \), with our choices of rounding modes, we monitored the relative errors

\[
\left| \frac{\tilde{R}(x+y) - (x+y)}{(x+y)} \right| \quad \text{and} \quad \left| \frac{\tilde{R}(x \ast y) - (x \ast y)}{(x \ast y)} \right|
\]

for each operation we performed, and found all of them to be smaller than \( 1.97\epsilon \).

By contrast, under the usual models of floating point arithmetic [5], we already have theories that bound the rounding errors in terms of \( \epsilon \), \( n \) and the Lebesgue constant for other barycentric interpolation schemes, as in [7] [13] [14] [15]. In particular, [4] provides such bounds for usual Floater-Hormann interpolants.
References

1. Bochkanov, S., AlgLib, an open source library for numerical computation, available at www.alglib.net.
2. 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.
3. Brutman, L., Lebesgue functions for polynomial interpolation – a survey. Ann. Numer. Math. 4 (1997) pp. 111–127.
4. de Camargo, A. Pierro, On the numerical stability of the Floater-Hormann interpolants, work in progress.
5. Floater, S., and Hormann, K., Barycentric rational interpolation with no poles and high rates of approximation. Numer. Math. 107 (2) (2007) pp. 315–331.
6. Higham, N. J., Accuracy and Stability of Numerical Algorithms, 2nd ed., SIAM, Philadelphia (2002).
7. Klein, G., An extension of the Floater-Hormann family of barycentric rational interpolants, ICMS Edinburgh, June 24, 2013, 16th IMA Leslie Fox Prize in Numerical Analysis, slides, downloaded from [http://people.maths.ox.ac.uk/kleing/research.html](http://people.maths.ox.ac.uk/kleing/research.html) on aug. 19 2014.
8. Klein, G. and Berrut, J.-P., Linear rational finite differences from derivatives of barycentric rational interpolants. SIAM J. Numer. Anal. 50 (2) (2012) pp. 643–656.
9. Mascarenhas, W. F., The stability of barycentric interpolation at the Chebyshev points of the second kind, Numer. Math., (2014) available online, DOI: 10.1007/s00211-014-0612-6.
10. Mascarenhas, W. F. and de Camargo, A. Pierro, On the backward stability of the second barycentric formula for interpolation, Dolomites Res. Notes Approx. 7 (2014) pp. 1–12.
11. Mascarenhas, W. F. and de Camargo, A. Pierro, The effects of rounding errors in the nodes on barycentric interpolation, [arXiv:1309.7970v2 [math.NA]](http://arxiv.org/abs/1309.7970) 4 Jul 2014, submitted to Numer. Math.
12. Fousse, L., Hanrot, G., Lefèvre, V., Pelissier P., and Zimmermann, P., MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding. ACM Trans. Math. Softw. 33(2), (2007) article 13.
13. Trefethen, L. N., and Weideman, J., Two results on polynomial interpolation in equally spaced points J. Approx. Theory 65 (1991) pp. 247–260.