Studies in Mathematical Sciences
Vol. 2, No. 2, 2011, pp. 67-77
www.cscanada.org

ISSN 1923-8444 [Print]
ISSN 1923-8452 [Online]
www.cscanada.net

Range and Domain Partitioning in Piecewise Polynomial Approximation

J. S. C. Prentice[1,*]

Abstract: Error control in piecewise polynomial interpolation of a smooth univariate function f requires that the interval of approximation be subdivided into many subintervals, on each of which an interpolating polynomial is determined. The number of such subintervals is often overestimated through the use of a high-order derivative of f. We report on a partitioning algorithm, in which we attempt to reduce the number of subintervals required, by imposing conditions on f and its relevant higher derivative. One of these conditions facilitates a distinction between the need for absolute or relative error control. Two examples demonstrate the effectiveness of this partitioning algorithm.

Key Words: Piecewise Polynomial; Range Partitioning; Domain Partitioning; Error Control

1. INTRODUCTION

Approximation of smooth univariate functions by means of piecewise interpolatory polynomials of relatively low degree is favoured over approximation with polynomials of high degree. This is due to the fact that determining the coefficients of an interpolating polynomial typically requires the inversion of a Vandermonde system, and such systems tend to become badly conditioned as the degree of the interpolating polynomial is increased [1]. Moreover, a piecewise approach, wherein the degree is fixed, lends itself to error control, while attempting to approximate using a single polynomial of very high degree does not easily admit error control.

The price one pays, however, for using low-degree interpolatory piecewise approximation, is that the interval under consideration must be subdivided into a number of subintervals (greater accuracy requires more subintervals), so that the coefficients of several, and probably many, polynomials must be determined and stored. From an efficiency point of view, then, it seems desirable to try to reduce the number of subintervals needed for a given level of accuracy.

In this paper, we introduce a *partitioning algorithm*, whereby we attempt to subdivide the interval of approximation into contiguous regions (in a principled way), such that piecewise approximation carried out on each region would result in fewer subintervals overall, for a user-defined tolerance on the approximation error, compared to the same problem without partitioning.

[1] Department of Applied Mathematics, University of Johannesburg, South Africa.
[*]Corresponding author.
[†]Received 10 April 2011; accepted 9 May 2011.

2. RELEVANT CONCEPTS

In this section, we briefly describe relevant concepts. The reader is referred to the literature (e.g. [2][3][4][5][6]) for more detail regarding polynomial interpolation.

2.1 Polynomial Interpolation

The polynomial of degree n that interpolates the continuous function $f(x)$ at $n+1$ distinct nodes x_i on an interval $[x_0, x_1, \ldots, x_n]$, known as the *Lagrange interpolating polynomial*, is given by

$$P_n(x) = \sum_{k=0}^{n} \left[\prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} \right] f(x_k) \equiv \sum_{k=0}^{n} L_{n,k}(x) f(x_k),$$

where $L_{n,k}(x)$ is known as the nth *Lagrange coefficient polynomial.*

The pointwise approximation error in the Lagrange polynomial is given by

$$\Delta P_n(x) \equiv f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

where $\xi(x) \in (x_0, x_n)$.

The polynomial of degree $2n+1$ (at most) that interpolates the continuous function $f(x)$, and its first derivative $f'(x)$, at $n+1$ distinct nodes x_i on an interval $[x_0, x_1, \ldots, x_n]$, known as the *Hermite interpolating polynomial*, is given by

$$H_{2n+1}(x) = \sum_{k=0}^{n} \left[1 - 2(x - x_k) L'_{n,k}(x_k)\right] L^2_{n,k}(x) f(x_k) + (x - x_k) L^2_{n,k}(x) f'(x_k),$$

and the pointwise approximation error in the Hermite polynomial is given by

$$\Delta H_{2n+1}(x) \equiv f(x) - H_{2n+1}(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} \prod_{i=0}^{n} (x - x_i)^2,$$

where $\xi(x) \in (x_0, x_n)$.

These interpolating polynomials are unique, provided the nodes $[x_0, x_1, \ldots, x_n]$ are distinct. Also, we have implicitly assumed that $f(x)$ is suitably differentiable, so that the higher derivatives in the error expressions exist.

2.2 Piecewise Interpolation with Error Control

Consider the task of approximating $f(x)$ by means of a Lagrange polynomial $P_n(x)$, where n is fixed *a priori*, on an interval $[a, b]$ subject to a user-imposed tolerance ε on the pointwise error $\Delta P_n(x)$.

We subdivide $[a, b]$ into N subintervals, and we find a $P_n(x)$ on each subinterval such that

$$|\Delta P_n(x)| \leqslant \varepsilon \qquad (1)$$

for all x on each subinterval.

The number of subintervals necessary is determined by the condition

$$|\Delta P_n(x)| = \left| \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i) \right| \leqslant \varepsilon$$

which leads to

$$N = \left\lceil \frac{(b-a)}{n} \left(\frac{S \max_{[a,b]} \left| f^{(n+1)} \right|}{\varepsilon (n+1)!} \right)^{\frac{1}{n+1}} \right\rceil \tag{2}$$

where

$$S \equiv \max_{s \in [0,n]} \left| \prod_{i=0}^{n} (s - i) \right|,$$

and $\lceil \cdots \rceil$ indicates 'round up to nearest integer'. Here, we have assumed that the $n+1$ interpolation nodes on each subinterval are equispaced, with spacing h. In terms of h, we can write

$$|\Delta P_n(x)| \leqslant \frac{h^{n+1} S \max_{[a,b]} \left| f^{(n+1)} \right|}{(n+1)!}.$$

It is important at this juncture to appreciate that, since $N \propto \max_{[a,b]} \left| f^{(n+1)} \right|$, we would usually overestimate the number of subintervals needed. The partitioning algorithm that is the subject of this work is motivated, in part, by the desire to reduce the extent of this overestimation.

The tolerance imposed in (1) corresponds to *absolute* error control. We can choose to control the *relative* error, as in

$$\frac{|\Delta P_n(x)|}{|f(x)|} \leqslant \varepsilon \Rightarrow |\Delta P_n(x)| \leqslant \varepsilon |f(x)|.$$

This amounts to replacing the tolerance ε in (2) with the effective tolerance $\varepsilon \min_{[a,b]} |f(x)|$, since

$$|\Delta P_n(x)| \leqslant \varepsilon \min_{[a,b]} |f(x)| \Rightarrow |\Delta P_n(x)| \leqslant \varepsilon |f(x)|$$

for all x on $[a,b]$. Note that, from (2), if $\min_{[a,b]} |f(x)| < 1$, then N would be larger for relative error control, than for absolute error control. Ultimately, we seek to limit the number of subintervals needed, and so we propose that relative error control is used only when $\min_{[a,b]} |f(x)| \geqslant 1$. If $|f(x)| < 1$ anywhere on $[a,b]$, we impose absolute error control.

For Hermite interpolation we have a similar expression for N, as in

$$\begin{aligned} N &= \left\lceil \frac{(b-a)}{n} \left(\frac{S \max_{[a,b]} \left| f^{(2n+2)} \right|}{\varepsilon (2n+2)!} \right)^{\frac{1}{2n+2}} \right\rceil \\ S &\equiv \max_{s \in [0,n]} \left| \prod_{i=0}^{n} (s-i)^2 \right|. \end{aligned} \tag{3}$$

3. THE PARTITIONING ALGORITHM

3.1 Range Partitioning

From (2) we see that the approximation error is proportional to $\max_{[a,b]} \left| f^{(n+1)} \right|$. Now consider the case where

$$\left| f^{(n+1)}(a) \right| < \left| f^{(n+1)}(c) \right| < \left| f^{(n+1)}(b) \right| \tag{4}$$

where $a < c < b$. Then, if we partition $[a, b]$ by means of c, we have

$$\begin{aligned} N_1 &= \left\lceil \frac{(c-a)}{n} \left(\frac{S \max_{[a,c]} \left| f^{(n+1)} \right|}{\varepsilon (n+1)!} \right)^{\frac{1}{n+1}} \right\rceil \\ N_2 &= \left\lceil \frac{(b-c)}{n} \left(\frac{S \max_{[c,b]} \left| f^{(n+1)} \right|}{\varepsilon (n+1)!} \right)^{\frac{1}{n+1}} \right\rceil \end{aligned}$$

and

$$N_1 + N_2 \leqslant N.$$

In other words, the partition could result in fewer subintervals being needed for the approximation.

The partition induced by c is based on the condition (4), and we refer to this type of partition as a *range partition* (since c defines a point at which the range of $\left| f^{(n+1)} \right|$ is subdivided). It should be clear that since $c - a$ and $b - c$ are both smaller than $b - a$, and if $\max_{[a,c]} \left| f^{(n+1)} \right| < \max_{[c,b]} \left| f^{(n+1)} \right|$, then we will definitely have $N_1 + N_2 < N$. The condition $\max_{[a,c]} \left| f^{(n+1)} \right| < \max_{[c,b]} \left| f^{(n+1)} \right|$ can be guaranteed if the *monotonicity* of $\left| f^{(n+1)} \right|$ can be identified - we say more about this in the next section. Obviously, range partitioning helps to reduce the overestimation of N inherent in (2), by creating regions of $[a, b]$ on which $\max \left| f^{(n+1)} \right|$ differs - rather than using a single value of $\max \left| f^{(n+1)} \right|$, which is a larger-than-necessary value on many parts of $[a, b]$.

Another type of range partitioning is motivated by the need to choose between absolute and relative error control. Here, we identify those points on $[a, b]$ where $|f(x)| = 1$. These points will define a partition of $[a, b]$ into regions where $|f(x)| < 1$ and regions where $|f(x)| \geqslant 1$. On the former, we impose absolute error control; on the latter, we impose relative error control. Effectively, these points partition $[a, b]$ into regions where the range of $|f(x)|$ is larger or smaller than unity.

3.2 Domain Partitioning

Now, assume that $\left| f^{(n+1)} \right|$ is monotonically increasing on $[a, b]$. This implies, in terms of the stepsize h,

$$\begin{aligned} \frac{h^{n+1} S \left| f^{(n+1)}(a) \right|}{(n+1)!} &\leqslant |\Delta P_n(x)| \leqslant \frac{h^{n+1} S \left| f^{(n+1)}(c) \right|}{(n+1)!} \quad \text{on} \quad [a, c] \\ \frac{h^{n+1} S \left| f^{(n+1)}(c) \right|}{(n+1)!} &\leqslant |\Delta P_n(x)| \leqslant \frac{h^{n+1} S \left| f^{(n+1)}(b) \right|}{(n+1)!} \quad \text{on} \quad [c, b], \end{aligned}$$

so that the lower and upper bounds on the error on the partitions $[a, c]$ and $[c, b]$ occur at the endpoints of the partitions. The only way to ensure that $\left| f^{(n+1)} \right|$ is monotonic on a given subinterval is to find all roots and stationary points of $\left| f^{(n+1)} \right|$. These points then define a partition of $[a, b]$ such that $\left| f^{(n+1)} \right|$ is monotonic on each region of the partition.

Since we also need to find $\min_{[a,b]} |f(x)|$ for the sake of relative error control, it makes sense to ensure that $|f(x)|$ is also monotonic on any given subinterval. In such case, $\min_{[a,b]} |f(x)|$ occurs at one of the endpoints of the subinterval. Hence, we also find the roots and stationary points of $|f(x)|$ on $[a, b]$.

Note that the stationary points of $|f(x)|$ and $\left| f^{(n+1)}(x) \right|$ can be found by locating the roots of $|f'(x)|$ and $\left| f^{(n+2)}(x) \right|$. Since the roots are the points where the relevant function touches or crosses the x-axis, we refer to this type of partitioning as *domain partitioning*.

3.3 Approximation with Partitioning

We now turn to the task of approximating $f(x)$ on $[a, b]$ by means of a piecewise Lagrange interpolating polynomial of degree n, using partitioning.

We begin by performing domain partitioning. This yields a partition of $[a, b]$, which we will denote $[a, z_1, z_2, \ldots, b]$. The z_j subdivide $[a, b]$ into contiguous regions; on each of these regions, both $|f(x)|$ and $\left|f^{(n+1)}(x)\right|$ are monotonic (we use the term *region* when referring to the partition of $[a, b]$, to avoid confusion with the term *subinterval,* which we will use in reference to subdivisions of a region, as will be encountered later).

Next, we carry out range partitioning with respect to $\left|f^{(n+1)}(x)\right|$ on each region of $[a, z_1, z_2, \ldots, b]$. However, we do not simply introduce a single additional node in each region; rather, we attempt to partition the range of $\left|f^{(n+1)}(x)\right|$ on each region in such a way that successive values of $\left|f^{(n+1)}(x)\right|$ in this partitioned range differ from each other by a constant user-imposed factor θ. Such a partition will generally result in the introduction of several nodes in the region. We give a detailed description as follows:

1) Say we are working on the region $\left[z_j, z_{j+1}\right]$. By construction of $\left[z_j, z_{j+1}\right]$, $\left|f^{(n+1)}(x)\right|$ is monotonic on this region. Without loss of generality, assume that

$$M_1 < M_2,$$

where

$$\begin{aligned} M_1 &\equiv \min_{\left[z_j, z_{j+1}\right]} \left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \left|f^{(n+1)}\left(z_j\right)\right|^{\frac{1}{n+1}} \\ M_2 &\equiv \max_{\left[z_j, z_{j+1}\right]} \left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \left|f^{(n+1)}\left(z_{j+1}\right)\right|^{\frac{1}{n+1}}. \end{aligned}$$

We consider the $(n+1)$th root here, since N is proportional to $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}}$, and it is also understood that it is the positive real root.

2) If $M_2 \leqslant 1$, we make no partition of $\left[z_j, z_{j+1}\right]$.

3) If $M_1 < 1$ and $M_2 > 1$, we form the set

$$\Theta \equiv \left\{M_1, 1, \theta, \theta^2, \theta^3, \ldots, \theta^p, M_2\right\},$$

where

$$p = \left\lceil \frac{\ln M_2}{\ln \theta} - 1 \right\rceil.$$

4) If $M_1 \geqslant 1$, we form the set

$$\Theta \equiv \left\{M_1, \theta M_1, \theta^2 M_1, \theta^3 M_1, \ldots, \theta^p M_1, M_2\right\},$$

where

$$p = \left\lceil \frac{\ln\left(\frac{M_2}{M_1}\right)}{\ln \theta} - 1 \right\rceil.$$

5) For the most part, any two successive members of these sets differ in magnitude by a factor of θ. The exceptions are possibly $\{M_1, 1\}$ and $\{\theta^p, M_2\}$ in #3, and $\{\theta^p M_1, M_2\}$ in #4.

6) The choice of p is such that $\theta^{p+1} \geqslant M_2$ in #3, and $\theta^{p+1} M_1 \geqslant M_2$ in #4.

7) We then find the roots of
$$\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \Theta.$$
See 'Comments' later in this section, and the Appendix.

8) These roots form the desired partition of $\left[z_j, z_{j+1}\right]$. If $\left|f^{(n+1)}(x)\right|$ is decreasing on $\left[z_j, z_{j+1}\right]$, we still have
$$M_1 < M_2,$$
but
$$\begin{aligned} M_1 &\equiv \min_{\left[z_j,z_{j+1}\right]} \left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \left|f^{(n+1)}\left(z_{j+1}\right)\right|^{\frac{1}{n+1}} \\ M_2 &\equiv \max_{\left[z_j,z_{j+1}\right]} \left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \left|f^{(n+1)}\left(z_j\right)\right|^{\frac{1}{n+1}}. \end{aligned}$$
(i.e. interchange z_j and z_{j+1}). The rest of the procedure is the same.

9) This procedure is applied to each region in $[a, z_1, z_2, \ldots, b]$. The result is a partition of $[a, b]$ wherein, on a generic region $[\alpha,\beta]$, we have
$$\begin{aligned} M_1 &< 1 \\ &\text{or} \\ 1 \leqslant \left|f^{(n+1)}(\alpha)\right|^{\frac{1}{n+1}} &< \left|f^{(n+1)}(\beta)\right|^{\frac{1}{n+1}} = \theta\left|f^{(n+1)}(\alpha)\right|^{\frac{1}{n+1}} \\ &\text{or} \\ 1 \leqslant \left|f^{(n+1)}(\alpha)\right|^{\frac{1}{n+1}} &< M_2 \end{aligned}$$
if $\left|f^{(n+1)}(x)\right|$ is increasing on $[\alpha,\beta]$, and
$$\begin{aligned} M_1 &< 1 \\ &\text{or} \\ 1 \leqslant \left|f^{(n+1)}(\beta)\right|^{\frac{1}{n+1}} &< \left|f^{(n+1)}(\alpha)\right|^{\frac{1}{n+1}} = \theta\left|f^{(n+1)}(\beta)\right|^{\frac{1}{n+1}} \\ &\text{or} \\ 1 \leqslant \left|f^{(n+1)}(\beta)\right|^{\frac{1}{n+1}} &< M_2 \end{aligned}$$
if $\left|f^{(n+1)}(x)\right|$ is decreasing on $[\alpha,\beta]$.

10) Essentially, we have formed a base-θ logarithmic partition of the range of $\left|f^{(n+1)}(x)\right|$ on $\left[z_j, z_{j+1}\right]$, and the nodes corresponding to this range partition have been used to partition $\left[z_j, z_{j+1}\right]$.

Lastly, we find the roots of $|f(x)| = 1$ to complete the partitioning process. The end result of this entire procedure is a partition of $[a, b]$ such that, on each region, both $|f(x)|$ and $\left|f^{(n+1)}(x)\right|$ are monotonic, $|f(x)| < 1$ or $|f(x)| \geqslant 1$, and $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}}$ does not vary by a factor of more than θ.

We now determine the number of subintervals N required on each region consistent with the user-imposed tolerance ε, with absolute error control applied on regions with $|f(x)| < 1$, and relative error control applied on regions with $|f(x)| \geqslant 1$. The quantities $\min|f(x)|$ and $\max\left|f^{(n+1)}(x)\right|$ needed in determining N on each region are easily obtained from the endpoints of the region in question, since $|f(x)|$ and $\left|f^{(n+1)}(x)\right|$ are monotonic on each region.

This partitioning algorithm can also be used for Hermite interpolation: we use $\left|f^{(2n+2)}(x)\right|$, $\left|f^{(2n+2)}(x)\right|^{\frac{1}{2n+2}}$ and (3); all other aspects are the same.

3.4 Refinement

On a generic region $[\alpha,\beta]$ of the final partition of $[a,b]$, we might find

$$\frac{(\beta-\alpha)}{n}\left(\frac{S\max_{[\alpha,\beta]}\left|f^{(n+1)}\right|}{\varepsilon(n+1)!}\right)^{\frac{1}{n+1}} \ll 1 \tag{5}$$

but because of the roundup, as in (2), this still represents $N = 1$ on the region. It may be better to simply absorb this region into a neighbouring region on which $N \geqslant 1$. This is achieved by simply deleting one of the endpoints of the region. Our strategy in this regard is to identify all those regions for which (5) is true, and to delete the right endpoint of each such region (except for the rightmost region of $[a,b]$, which we never delete). This results in a modified partition of $[a,b]$, containing fewer regions, and on this modified partition we then determine new, and final, values of N.

3.5 Comments

A few comments regarding the previous are in order:

1) We have assumed that the roots of various functions can be found. Of course, if these functions are smooth, as is usually the practical case, then simple roots can be found easily using the bisection method [7]. Although the bisection method is only linearly convergent, it is robust for simple roots. For compound roots (of multiplicity greater than one), Newton's method would be preferred. In our own work, we have found that computer algebra software (CAS) can also be used; however, not all functions can be solved using CAS, so that numerical methods are probably the most reliable, generally speaking. It is not our intention to report on such methods - the reader is referred to the extensive literature on this subject - rather, we make the assumption that potential users of our partitioning algorithm are suitably familiar with root-finding techniques.

2) What is a suitable choice for the scale parameter θ? If θ is too close to unity, the resulting partition could have regions that are so small that they are ultimately deleted by the refinement procedure. On the other hand, if θ is too large, the resulting large regions would have larger N than desired, thus defeating the purpose of the partitioning algorithm. We suggest that

$$2 \leqslant \theta \leqslant 5.$$

Of course, this is our own subjective opinion, based on our experience with our own approximation work. However, it must be remembered that the number of subintervals on each region can be determined, for a given θ, before the approximation polynomial coefficients are computed. Hence, one could experiment to some extent with various values of θ, before deciding on a suitable value. There is no doubt that a suitable value of θ for a given approximation problem is problem-specific and, ultimately, such value of θ is the subjective, yet informed, choice of the user.

3) It is also possible to select a different value of θ for different regions, depending on the behaviour of $\left|f^{(n+1)}\right|$ on each region. This, however, is merely a refinement to the algorithm, and we have not considered this approach in our numerical examples.

4. EXAMPLES

We use partitioning in approximating

$$f(x) = e^{x} - \frac{1}{2}$$

on $[0, 15]$, and

$$f(x) = \frac{10}{10x^2 + 1}$$

on $[-5, 5]$, using piecewise Lagrange and Hermite interpolation.

The first of these examples is a rapidly increasing function with no roots or stationary points on the interval of approximation. Consequently, only range partitioning is relevant. The second example, a variation of Runge's function, has an oscillating higher derivative, and both domain and range partitioning are appropriate. In both examples, there are regions where absolute error control is appropriate, and regions where relative error control can be applied.

4.1 Exponential Function

We find the number of subintervals needed to approximate

$$f(x) = e^x - \frac{1}{2}$$

on $[0, 15]$, using piecewise Lagrange interpolation. In the tables below, the symbol N denotes the number of subintervals without applying the partitioning algorithm; the symbol N_p denotes the number of subintervals needed after applying partitioning. We consider two values for ε : a moderate value of 10^{-6}, and a stringent value of 10^{-12}. Furthermore, we find N_p for several values of $\theta \in [2, 5]$. We also consider two values of n. Results are shown in Table 1. For this example,

$$\frac{1}{2} \leqslant \left|\frac{d^4 f}{dx^4}\right|, \left|\frac{d^8 f}{dx^8}\right| \leqslant e^{15} - \frac{1}{2} \approx 3.269 \times 10^6.$$

Table 1: N_p **and** N **(Lagrange, exponential example)**

n	3				7			
θ	2	3	4	5	2	3	4	5
$N_p\left(\varepsilon = 10^{-6}\right)$	142	200	257	319	16	20	26	33
$N\left(\varepsilon = 10^{-6}\right)$	3038				47			
$N_p\left(\varepsilon = 10^{-12}\right)$	4359	6277	8073	10005	78	107	136	178
$N\left(\varepsilon = 10^{-12}\right)$	96056				264			

In all cases, N_p is smaller than N, and in some cases, considerably so. This demonstrates the efficacy of partitioning. Also, N_p increases as θ increases, showing how the scale of the range partitioning can influence N_p. For the higher-order approximation ($n = 7$), far fewer subintervals are needed - due to the effect of the factorial in the denominator in (2).

In Table 2, we show results for Hermite interpolation with $n = 3$. The same qualitative behaviour is apparent.

4.2 Runge's Function

Runge's function is often used as an example in favour of piecewise polynomial approximation, wherein error control is effective, as opposed to approximating the function by means of a single very high degree polynomial [5]. Such a polynomial is likely to exhibit oscillatory behaviour, completely at odds with the character of Runge's function.

Table 2: N_p **and** N **(Hermite, exponential example)**

n	3			
θ	2	3	4	5
$N_p\left(\varepsilon = 10^{-6}\right)$	34	47	59	78
$N\left(\varepsilon = 10^{-6}\right)$	113			
$N_p\left(\varepsilon = 10^{-12}\right)$	184	254	322	424
$N\left(\varepsilon = 10^{-12}\right)$	632			

We find the number of subintervals needed to approximate

$$f(x) = \frac{10}{10x^2 + 1}$$

on $[-5, 5]$, using piecewise Lagrange interpolation. In Tables 3 and 4 below, the symbols have the same meaning as described previously. For this example,

$$\begin{aligned} 0 &\leqslant \left|\frac{d^4 f}{dx^4}\right| \leqslant 24000 \\ 0 &\leqslant \left|\frac{d^8 f}{dx^8}\right| \leqslant 4.032 \times 10^9. \end{aligned}$$

Table 3: N_p **and** N **(Lagrange, Runge's example)**

n	3				7			
θ	2	3	4	5	2	3	4	5
$N_p\left(\varepsilon = 10^{-6}\right)$	168	169	169	168	29	29	29	29
$N\left(\varepsilon = 10^{-6}\right)$	890				114			
$N_p\left(\varepsilon = 10^{-12}\right)$	5052	5089	5142	5117	133	134	134	134
$N\left(\varepsilon = 10^{-12}\right)$	28118				641			

The behaviour of N_p here is similar to that in the previous example.

In Table 4, we show results for Hermite interpolation with $n = 3$. Again, the same qualitative behaviour is apparent.

Table 4: N_p **and** N **(Hermite, Runge's example)**

n	3			
θ	2	3	4	5
$N_p\left(\varepsilon = 10^{-6}\right)$	60	60	60	60
$N\left(\varepsilon = 10^{-6}\right)$	274			
$N_p\left(\varepsilon = 10^{-12}\right)$	310	314	314	315
$N\left(\varepsilon = 10^{-12}\right)$	1539			

5. CONCLUSION

We have developed a partitioning algorithm, designed to reduce the number of subintervals required for piecewise polynomial interpolation of degree n of a smooth function $f(x)$. The algorithm is based on *domain partitioning*, where the roots and stationary points of $|f(x)|$ and $\left|f^{(n+1)}(x)\right|$ are used to define regions upon which these functions are monotonic. So-called *range partitioning* is then used to subdivide each of these regions into smaller regions, upon which $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}}$ does not vary by more than a user-imposed scale factor. Range partitioning is also used to find points at which $|f(x)| = 1$. On each region defined by these partitions, piecewise polynomial interpolation is performed, subject to a tolerance imposed on the absolute/relative error. The nett result is that the total number of subintervals required for such interpolation (and, hence, the number of polynomials needed in the piecewise approximation) is reduced, perhaps greatly so, compared to piecewise approximation without partitioning (in which the number of subintervals is usually greatly overestimated). Two simple but meaningful examples have demonstrated the beneficial effects of this partitioning algorithm, particularly for Lagrange interpolation of relatively low degree and high accuracy.

REFERENCES

[1] Higham, N.J. (2002). *Accuracy and Stability of Numerical Algorithms, 2nd ed.*. Philadelphia: SIAM.

[2] Burden, R.L., and Faires, J.D. (2011). *Numerical Analysis, 9th ed.*. Brooks/Cole.

[3] Kincaid, D., and Cheney, W. (2002). *Numerical Analysis: Mathematics of Scientific Computing 3rd ed.*. Pacific Grove: Brooks/Cole.

[4] Mhaskar, H.N., and PAi, D.V. (2000). *Fundamentals of Approximation Theory*. Boca Raton: CRC Press.

[5] Isaacson, E., and Keller, H.B. (1994). *Analyisis of Numerical Methods*. New York: Dover.

[6] Hamming, R.H. (1986). *Numerical Methods for Scientists and Engineers*. New York: Dover.

[7] Gerald, C.F., and Wheatley, P.O. (1984). *Applied Numerical Analysis, 3rd ed.*. Massachusetts: Addison-Wesley.

A. APPENDIX

A.1 Roots of $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \Theta$

Here, we discuss the use of the bisection method to find the roots of $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \Theta$ on a region. To begin with, we clarify our notation: by $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \Theta$ we imply the set of equations

$$\begin{aligned}
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= M_1 \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= 1 \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= \theta \\
&\vdots \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= \theta^p \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= M_2,
\end{aligned}$$

or

$$\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = M_1$$

$$\begin{aligned}
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} &= \theta M_1 \\
&\vdots \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} - \Theta &= \theta^p M_1 \\
\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} - \Theta &= M_2,
\end{aligned}$$

depending on which set Θ, as described earlier, is relevant.

To begin with, $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}}$ is monotonic on the region - a consequence of domain partitioning. Therefore, the endpoints of the region are the roots of the first and last of the equations in either set. This means that the remaining equations have simple roots - roots where $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} - \Theta$ crosses the x-axis - on the region. In other words, $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} - \Theta$ changes sign on the region. This means that the region brackets the root of each equation, and so the region can be used as the starting interval for the bisection method. Consequently, the bisection method can be used to find the roots of $\left|f^{(n+1)}(x)\right|^{\frac{1}{n+1}} = \Theta$ and, what is more, these roots are guaranteed to be found.

Incidentally, the same reasoning holds for solving $|f(x)| = 1$ on any region. Domain partitioning ensures that $|f(x)|$ is monotonic on any region. Hence, one of the endpoints is the root or the root lies within the region and is a simple root, so that the bisection method is appropriate, and the region serves as the starting interval.