Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 273

Newton Interpolation with Extremely High Degrees by Leja Ordering and Fast Leja Points

Michael Breuß, Oliver Vogel and Kai Uwe Hagenburg

Saarbrücken 2010

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 273
submitted: August 31, 2010

Newton Interpolation with Extremely High Degrees by Leja Ordering and Fast Leja Points

Michael Breuß
Saarland University
Faculty of Mathematics and Computer Science
P.O. Box 15 11 50
66041 Saarbrücken
Germany
breuss@mia.uni-saarland.de

Oliver Vogel
Saarland University
Faculty of Mathematics and Computer Science
P.O. Box 15 11 50
66041 Saarbrücken
Germany
vogel@mia.uni-saarland.de

Kai Uwe Hagenburg
Saarland University
Faculty of Mathematics and Computer Science
P.O. Box 15 11 50
66041 Saarbrücken
Germany
hagenburg@mia.uni-saarland.de

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax: + 49 681 302 4443
e-Mail: preprint@math.uni-sb.de
WWW: http://www.math.uni-sb.de/

Abstract

In this paper we perform a numerical study of Newton polynomial interpolation. We explore the Leja ordering of Chebyshev knots and the Fast Leja knots introduced by Reichel. In all previous publications we are aware of, the degree of interpolation polynomials in use is in the order of a few hundreds. We show that it is possible to employ degrees of up to one million or higher without a numerical stability problem or excessive computation times. We also show experimentally that Leja ordering and Fast Leja points enable stable and meaningful interpolation of functions that are just continuous or even discontinuous.

Keywords. Polynomial interpolation, Newton interpolation, interpolation knots, Leja ordering, Fast Leja points

1 Introduction

Polynomial interpolation (PI) is one of the fundamental tasks in numerical analysis, see e.g. the textbooks [1, 4, 10, 13, 18] for accounts on the subject. The goal of PI is to compute an approximation of an unknown function $f(x)$ over some onedimensional real interval of interest $[a, b]$. This is to be done at hand of a set of given numbers $f(x_j)$ at interpolation knots x_j, $j = 1, \ldots, n$, making use of a polynomial meeting exactly the data $(x_j, f(x_j))$. A popular format of the unique interpolation polynomial is the Newton form as its computation and evaluation can be done in an efficient way. The computation of high-degree interpolation polynomials for large numbers of given knots is a non-trivial task since the process easily becomes numerically unstable, see e.g. [21] for a useful discussion.

It is well-known that the location of the interpolation knots x_j influences the quality of the computed interpolant. For instance, a subject often discussed in textbooks such as those cited above is that the use of Chebyshev knots minimises the amplitude of the knot polynomial $k_n(x) := \Pi_{j=1}^{n}(x - x_j)$ and hence the error of PI. In a different setting utilising complex domains, the Leja points studied first in [7, 17] are known to be an optimal choice, cf. [3, 19]. In the complex setting also the Fejér points have favorable properties [9, 21], and it can be shown that these can be reduced to Chebyshev knots when considering a real interval [9].

However, not only the location but also the ordering of the knots can greatly affect the solution accuracy. Important works discussing this issue are the papers of Reichel [19] and Calvetti and Reichel [6]. While the Leja ordering

of points is developed mainly for use in the complex domain (and in general the ordering is not unique), also a version for use with real intervals can be infered heuristically. The resulting ordering scheme has been used in several fields of scientific computing, cf. [2, 5, 8]. It puts a fixed set of given points into a certain ordering, for instance it can be applied as in this paper at a set of predetermined Chebychev knots. As shown in [15] the Leja ordering is related to a partial pivoting of the matrix arising in Newton PI.
While it has been proven in the mentioned works that the Leja ordering can be useful, there is the undesirable property that each time one increases or decreases the number of knots, the Leja ordering has to be computed completely anew. As a remedy to this, the Fast Leja (FL) points were proposed in [3]. Inspired by the Leja point construction, the procedure of constructing the FL knots involves selecting the location out of a finite set of knot candidates that is constructed before the selection step. It also involves a fast way to determine the ordering during the selection step. In the FL process, increasing the number of points means to inherit the previously computed ones in the already determined order.

Our contribution. While it has been shown in previous papers that the Leja ordering and FL points have some benefits especially with respect to accuracy, the true potential of these techniques especially for Newton PI has not been pointed out. In fact, all examples we have seen in previous papers are just toy examples in comparison to what is possible. Even papers concerned specifically with high-degree Newton interpolation do not consider examples with more than a few hundred knots, compare e.g. [19, 21]. In this paper we show that one can easily use up to one million interpolation knots, or even more. In the thorough numerical study which is the subject of our work, we also show that it is experimentally possible to interpolate functions that are just continuous or even discontinuous with high precision. Let us note in this context that especially with FL points, the computational times stay very reasonable so that the effort can readily be justified.

Paper Organisation. In Section 2 we give a brief account on the aspects of the Newton PI problem, and we recall the Leja ordering and the FL points. This is followed by a presentation of our computational study in Section 4. The paper is finished by a conclusion.

2 Mathematical Basis

The purpose of this section is to recall for the convenience of the reader facts and methods from the already cited literature as employed in this work.

2.1 Newton Polynomial Interpolation

Given a discrete data point set $\{(x_j, f_j)\}$ for data points at position x_j with function value $f_j \equiv f(x_j)$ and $j \in \{1, \ldots, n\}$, a polynomial $P_n(x) \in \Pi_n$ is sought. This polynomial should satisfy the interpolation condition

$$P_n(x_j) = f_j \tag{1}$$

in all data points. Gathering the interpolation conditions from all points, a linear system of equations arises.
It is well known that it is of importance in which basis the interpolation polynomial is expressed. The *Newton basis* consists for n interpolation points of the polynomials

$$\mathsf{n}_k(x) \;:=\; \Pi_{i=1}^{k-1}\,(x - x_i) \qquad k = 1, \ldots, n \tag{2}$$

The benefit of using this special basis is that the matrix arising in the linear system of equations that gathers the interpolation conditions is of a simple structure, more precisely it is of lower triangular type:

$$\begin{bmatrix} 1 & & & & 0 \\ 1 & x_2 - x_1 & & & \\ 1 & x_3 - x_1 & (x_3 - x_1)(x_3 - x_2) & & \\ \vdots & \vdots & & \ddots & \\ 1 & x_n - x_1 & \ldots & \ldots & \prod_{i=1}^{n-1}(x_n - x_i) \end{bmatrix} \begin{bmatrix} a_1 \\ \\ \vdots \\ \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \\ \vdots \\ \\ y_n \end{bmatrix} \tag{3}$$

This system of equations can easily be solved via

$$N(x_j) \;:=\; \sum_{i=1}^{j} \mathsf{n}_i(x_j) a_i \;=\; y_j \qquad \text{iterating as by} \qquad j = 1, \ldots, n \tag{4}$$

Moreover, for the evaluation of the Newton interpolation polynomial $N(x)$ the efficient *Horner scheme* [16] can be used. The latter utilises the construction of $N(x_j)$, rewriting it as by

$$N(x_j) \;=\; a_1 + (x_j - x_1)\{a_2 + (x_j - x_2)[\ldots a_{n-1} + (x_j - x_{n-1})a_n]\}$$

This allows to use a stable, recursive $\mathcal{O}(n)$-algorithm to evaluate the polynomial:

$$\text{Set} \quad p := a_n \tag{5}$$

$$\text{for} \quad k = n-1, n-2, \ldots, 0: \; p := a_k + (x - x_k)p \tag{6}$$

and set p as the result for $N(x)$.

2.2 Chebyshev Knots

So far, we did not explicitly mention how the interpolation points x_j could be chosen. The most simple choice of equidistantly spaced points is well known to lead to numerical instability. One way to cope with this is to consider the roots of Chebyshev polynomials defined as

$$T_m(t) \;=\; \cos(m \arccos(t/b)) \qquad t \in [-b, b] \tag{7}$$

with $m = 0, 1, \ldots$, and their roots are then given by

$$t_k^m := b \cos\left(\frac{(2k-1)\pi}{2m}\right) \qquad k = 1, 2, \ldots, m \tag{8}$$

This choice of interpolation points appears to reduce the numerical error problem. See e.g. [1, 4, 18] for some more information on the mentioned aspects.

2.3 Leja Ordering

For m given interpolation knots in the set S_m, where S_m may for instance be obtained via (8), one determines the *Leja ordering* of the knots x_j via

$$|x_1| \;:=\; \max_{x \in S_m} |x| \tag{9}$$

$$\underbrace{\prod_{k=1}^{j-1} |x_j - x_k|}_{\text{'multiplicative distance'}} \;=\; \max_{j \le l \le m} \prod_{k=1}^{j-1} |x_l - x_k| \;, \quad 2 \le j \le m \,. \tag{10}$$

In order to implement this ordering, one simply starts with the largest point in the set of points. Then, one selects the point that has the largest distance to this point. Usually, these two points will be the endpoints of the interpolation interval $[a, b]$. Of the remaining knots in the set to be sorted, one tests all for their multiplicative distances to the points selected so far. Then one chooses the one with the largest multiplicative distance as the next in the order. This process is repeated until all points are in order.
Note, that sorting the points has a computational complexity worse than quadratic, but not worse than cubic. We consider as candidate points for the Leja ordering the Chebyshev knots.

2.4 Fast Leja (FL) Points

FL points [19] represent an alternative to a Leja ordering of Chebyshev knots. The algorithm to compute the FL points can be described in a simple way.

One starts with the limits a and b of the interpolation interval $[a, b]$, these are the first two points. Then, one chooses the midpoint between these two points as the first candidate point. Now, one recursively adds the candidate point with the largest multiplicative distance to all the previous points to the set of FL points, and one adds the midpoints between this point and its neighbours to the set of candidate points. Selecting the candidate point as FL point is done in a similar fashion as for the Leja ordering.
This procedure recursively gives the FL points in their correct order. We implemented this method in C-code. A working Matlab version of the code and details on the method can be found in [19].

3 Numerical Experiments

In this section, we will demonstrate that very high degree polynomial interpolation can be done using the numerical techniques presented earlier in this paper. We will put this in contrast to the naïve way of ordering the interpolation knots increasingly.

Technical remarks. For the evaluation of Newton interpolation polynomials of very high degree the size of the computational domain may be point of concern. It can be shown that a scaling of the x-domain to the length four can be favorable in such a setting since this may prevent some numerical inaccuracies [21]. Accordingly, we will employ such a scaling in our experiments.
Also, it is necessary to employ high-precision datatypes. All computations are done in the highest precision floating-point type available with most C-compilers on standard platforms, which is an 80-bit floating point number.

The interpolated functions. The numerical experiments are done on four different functions, all on the interval $[-2, 2]$:

1. The Runge function [20]

$$f_1(x) = \frac{1}{1 + 6.25x^2} \tag{11}$$

   which is a classic example for a function that is difficult to interpolate with high degree.

2. The Heaviside function

$$f_2(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0\,, \end{cases} \tag{12}$$

   which has a discontinuity at the origin.

3. A sawtooth function
$$f_3(x) = x - [x] \tag{13}$$
with several discontinuities.

4.
$$f_3(x) = \sqrt{|x|} \tag{14}$$
which is not differentiable at the origin and has a pole in its first derivative.

See Figure 1 for corresponding plots.

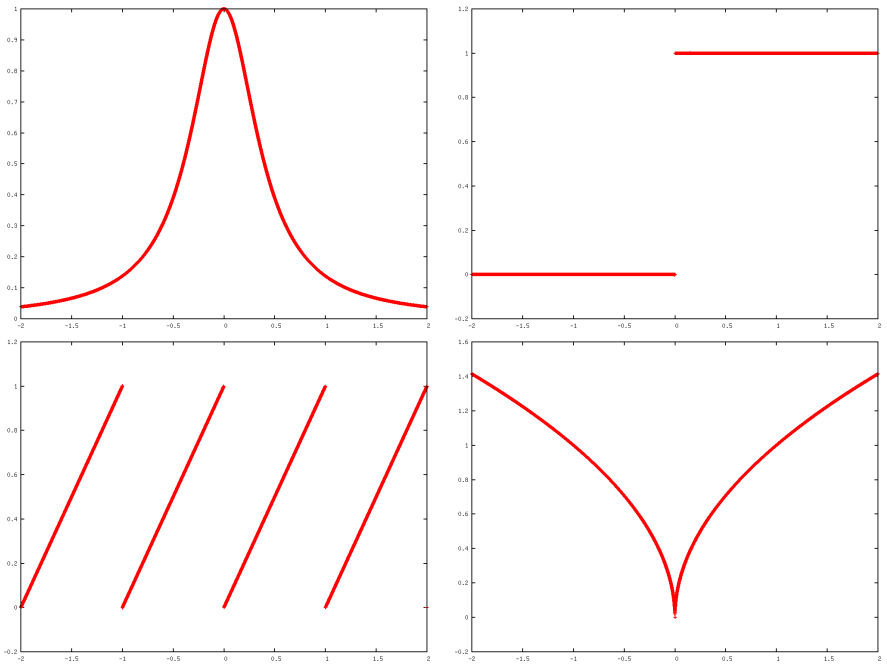

Figure 1: Input functions: Runge function, Heaviside function, sawtooth function, and $\sqrt{|x|}$.

Evaluation. To evaluate the numerical accuracy of the interpolation, we sample the interval with m equidistant points and evaluate the polynomial in each of these points. Then, we take the L_2 and L_∞ distances between the interpolated points and the exact function values at these sample points. For most experiments Chebyshev knots are used, as they result in much more accurate and stable reconstructions than equidistant knots. The only exception is the part concerned with FL points that are constructed algorithmically.

The error computations are done in the same precision as the interpolation itself.

3.1 Naïve Ordering

In the first experiment, we use the most simple approach for evaluating the interpolation polynomial, ordering the interpolation points from left to right. The main advantage of this approach is that it is free, i.e. no computational effort is necessary to produce this ordering.
As expected, despite the high accuracy of the used data structure and the favourable choice of Chebyshev knots, the process quickly becomes numerically unstable. As Table 1 demonstrates, the result starts to become unstable at polynomial degrees of 50–80 already, where the Sawtooth experiment proves to be the most challenging. Before even reaching degree 100, though, all experiments 'explode' numerically.

3.2 Leja Ordering

The Leja ordering has been stated to be stable at higher polynomial degrees [19]. However, the computations in the latter work were performed just for up to one hundred points, and the most difficult test function used there was the Runge function.
For this paragraph we take the Chebyshev knots as before, Table 2 shows the errors up to degree 10000. As we can observe, the process remains numerically stable up to this degree, which is a massive improvement compared to the naïve approach we used before.
Discussion. A major drawback of this approach is computational complexity. Ordering n points is worse than quadratic in n. Furthermore, having sorted the Chebyshev knots for degree n does not help at computing the order for degree $n+1$, these have to be computed separately. Consequently, this means that for being able to use polynomial reconstructions of arbitrary degrees, it is necessary to precompute a database of orderings for all possible degrees. As a guideline, with our implementation on standard desktop hardware, 1000 points can be ordered in several seconds, and 10000 points take about an hour.

3.3 Fast Leja Points

As indicated, the FL points introduced in [3] can be computed recursively, and the FL points are automatically ordered. The values of the first n FL points suffice as a database for all lower amounts of FL points. With our

implementation, it is possible to compute the first million FL points in less than half an hour. Note that this is roughly the same time as for computing the Leja ordering of a few more than 10000 Chebychev knots. This important detail has not been pointed out in earlier works where only examples of up to a few hundred degrees are used. Note also, that the FL points can be simply saved to a file for later use.

The interpolation results using $m = 10001$ sample points are summarised in Figure 2 and Table 3, respectively. It is quite impressive that for very high degrees, even the interpolation of the sawtooth function is perfect in this resolution. By Table 3 we observe the reconstruction errors of the four experiments up to polynomial degree 1000000. Clearly, all interpolations remain numerically stable.

Further discussion. Finally, let us discuss the effect of discontinuities. For our standard sampling of $m = 10001$ points we notice the L_∞ difference to decrease towards zero with increasing degree. Naturally, the interpolating polynomial does have an actual L_∞-error of at least one for the Heaviside function, and the lower value we obtain is caused by the sparse sampling of our test domain. Table 4 shows the L_∞-errors of the reconstruction of the Heaviside function for finer samplings, and indeed they approach 1 once the sampling is fine enough. L_∞ errors larger than one can be explained by Gibbs' phenomenon, not by numerical instabilities.

It is clear that for every polynomial degree, there is a sampling fine enough so that the L_∞-error is at least close to 1. In the opposite direction, it can be assumed that for every sampling of the test domain the L_∞-error approaches zero for very high polynomial degrees. However, as Table 4 demonstrates, this requires extremely high degrees, already at 1000001 samples, a degree of one million is necessary to get below an L_∞-error of 0.5. In Figure 3 we demonstrate the behaviour of the Newton interpolation polynomial for a sequence of finer sampling rates.

4 Conclusion

Both the Leja ordering and the Fast Leja points enable impressive, extremely high-degree Newton interpolation results. We think that these techniques should be a topic mentioned in standard numerical analysis books.

References

[1] K. Atkinson, An Introduction to Numerical Analysis, 2nd ed., Wiley, 1989.

[2] J. Baglama, D. Calvetti, L. Reichel, Iterative methods for the computation of a few eigenvalues of a large symmetric matrix, BIT 36, (1996), 400-421.

[3] J. Baglama, D. Calvetti, L. Reichel, Fast Leja Points, Electronic Transactions on Numerical Analysis 7, (1998), 124-140.

[4] R. Bulirsch, J. Stoer, Introduction to Numerical Analysis, 3rd ed., Springer, Berlin, 2002.

[5] D. Calvetti, L. Reichel, Adaptive Richardson iteration based on Leja points, Journal of Computational and Applied Mathematics 71, (1996), 267-286.

[6] D. Calvetti, L. Reichel, On the evaluation of polynomial coefficients, Numerical Algorithms 33, (2003), 153-161.

[7] A. Edrei, Sur les déterminants récurrents et les singularités d'une fonction donnée par son developpement de Taylor, Composito Mathematica 7, (1939), 20-88.

[8] A. Eisinberg, G. Fedele, On the inversion of the Vandermonde matrix, Applied Mathematics and Computation 174, (2006), 1384-1397.

[9] B. Fischer, L. Reichel, Newton Interpolation in Fejér and Chebyshev Points, Mathematics of Computation 53, No. 187, (1989), 265-278.

[10] W. Gautschi, Numerical Analysis: An Introduction, Birkhäuser, Boston, 1997.

[11] I. Gohberg, V. Olshevsky, The Fast Generalized Parker-Traub Algorithm for Inversion of Vandermonde and Related Matrices, Journal of Complexity 13, No. 2, (1997), 208-234.

[12] G. H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[13] M. Hanke-Bourgeois, Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens, B.G. Teubner, Stuttgart, 2002.

[14] N. Higham, A Survey of Condition Number Estimation for Triangular Matrices, SIAM Review 29, Issue 4, (1987), 575-596.

[15] N. Higham, Stability analysis of algorithms for solving confluent Vandermonde-like systems, SIAM Journal on Matrix Analysis and Applications 11, No. 1, (1990), 23-41.

[16] W. G. Horner, A new method of solving numerical equations of all orders, by continuous approximation, Philosophical Transactions of the Royal Society of London (1819), 308-335.

[17] F. Leja, Sur certaines suits liées aux ensemble plan et leur application à la representation conforme, Annales Polonici Mathematici 4, (1957), 8-13.

[18] G. Opfer, Numerische Mathematik für Anfänger, 5th ed., Vieweg + Teubner, Wiesbaden, 2008.

[19] L. Reichel, Newton interpolation at Leja points, BIT 30, Issue 2, (1990), 332-346.

[20] C. Runge, Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten, Zeitschrift für Mathematik und Physik 46, (1901), 224-243.

[21] H. Tal-Ezer, High Degree Polynomial Interpolation in Newton Form, SIAM Journal on Scientific and Statistical Computing 12, Issue 3, (1991), 648-667.

Table 1: Average numerical errors of the interpolation with Chebyshev interpolation knots, ordered from left to right, using $m = 10001$ sample points for the evaluation.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	5.5e-03	1.7e-02	1.1e-01	4.7e-03
20	9.7e-05	8.6e-03	4.2e-02	1.2e-03
30	1.8e-06	5.8e-03	1.6e-02	5.3e-04
40	3.4e-08	4.3e-03	2.0e-02	3.0e-04
50	6.5e-10	3.4e-03	2.2e-02	1.9e-04
60	1.3e-09	1.1e-02	6.0e+07	1.6e-04
70	2.5e-05	1.6e+04	1.1e+15	1.0e+02
80	1.2e+01	3.1e+10	5.6e+23	1.5e+09
90	2.6e+06	4.4e+17	2.4e+33	4.8e+15
100	5.5e+12	2.1e+24	1.8e+41	4.4e+23

L_2-errors.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	2.6e-01	5.5e-01	1.0e+00	4.7e-01
20	3.6e-02	5.2e-01	8.9e-01	3.4e-01
30	5.0e-03	5.1e-01	5.4e-01	2.7e-01
40	6.8e-04	5.1e-01	8.7e-01	2.4e-01
50	9.3e-05	5.1e-01	1.6e+00	2.1e-01
60	6.8e-04	1.3e+00	9.3e+04	1.9e-01
70	1.3e-01	2.2e+03	8.4e+08	2.8e+02
80	7.8e+01	4.4e+06	1.4e+13	1.0e+06
90	4.3e+04	1.6e+10	7.1e+17	1.5e+09
100	5.3e+07	4.0e+13	9.2e+21	1.0e+13

L_∞-errors.

Table 2: Average numerical errors of the interpolation with Chebyshev interpolation knots, ordered by Leja ordering, using $m = 10001$ sample points for the evaluation.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	5.4e-03	1.7e-02	1.1e-01	4.7e-03
20	1.0e-04	8.6e-03	4.2e-02	1.2e-03
30	1.9e-06	5.8e-03	1.6e-02	5.4e-04
40	2.1e-08	4.3e-03	2.0e-02	3.0e-04
50	1.3e-09	3.4e-03	1.6e-02	1.9e-04
60	7.6e-12	2.9e-03	8.1e-03	1.3e-04
70	1.6e-10	2.4e-03	1.2e-02	9.9e-05
80	6.9e-10	2.2e-03	1.0e-02	7.6e-05
90	8.4e-11	1.9e-03	5.2e-03	6.0e-05
100	1.7e-11	1.7e-03	8.0e-03	4.9e-05
1000	3.3e-12	1.7e-04	8.4e-04	5.6e-07
10000	9.9e-13	5.5e-05	4.3e-03	3.7e-08

L_2-errors.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	2.6e-01	5.5e-01	1.0e+00	4.7e-01
20	3.7e-02	5.2e-01	8.9e-01	3.4e-01
30	5.0e-03	5.1e-01	5.4e-01	2.8e-01
40	5.4e-04	5.1e-01	8.7e-01	2.4e-01
50	6.7e-05	5.1e-01	8.6e-01	2.1e-01
60	1.1e-05	5.0e-01	5.6e-01	1.9e-01
70	1.8e-05	5.0e-01	8.9e-01	1.8e-01
80	3.7e-05	5.0e-01	8.7e-01	1.7e-01
90	1.3e-05	5.0e-01	5.1e-01	1.6e-01
100	5.9e-06	5.0e-01	8.6e-01	1.5e-01
1000	2.6e-06	5.0e-01	8.6e-01	4.8e-02
10000	1.4e-06	5.1e-01	8.2e-01	1.5e-02

L_∞-errors.

Table 3: Interpolation errors with Fast Leja points as interpolation knots, using $m = 10001$ sample points for the evaluation.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	5.3e-03	5.4e-02	2.0e-01	2.0e-02
100	1.6e-18	5.9e-03	3.7e-02	2.3e-04
1000	3.7e-36	8.2e-04	3.6e-03	6.6e-06
10000	2.1e-35	3.7e-05	3.2e-04	4.1e-08
100000	5.5e-34	2.2e-05	2.2e-04	3.0e-09
1000000	3.4e-32	6.9e-10	1.9e-04	6.0e-06

L_2-errors.

degree	Runge	Heaviside	Sawtooth	$\sqrt{\lvert x \rvert}$
10	1.6e-01	1.0e+00	1.0e+00	3.0e-01
100	3.5e-09	1.0e+00	1.0e+00	1.0e-01
1000	2.4e-17	9.8e-01	1.0e+00	5.8e-02
10000	5.0e-17	4.3e-01	1.0e+00	1.7e-02
100000	3.5e-16	2.8e-01	1.0e+00	3.3e-03
1000000	7.8e-15	2.0e-03	1.0e+00	6.1e-15

L_∞-errors.

Table 4: Average interpolation errors for the Heaviside function using fast Leja points, using $m = 10001$, $m = 100001$, and $m = 1000001$ sample points for the evaluation.

	$m = 10001$		$m = 100001$		$m = 1000001$	
degree	L_2	L_∞	L_2	L_∞	L_2	L_∞
10	5.4e-02	1.0e+00	5.4e-02	1.0e+00	5.4e-02	1.0e+00
100	5.9e-03	1.0e+00	6.0e-03	1.0e+00	6.0e-03	1.0e+00
1000	8.2e-04	9.8e-01	8.6e-04	1.0e+00	8.7e-04	1.0e+00
10000	3.7e-05	4.3e-01	6.8e-05	9.1e-01	7.2e-05	9.9e-01
100000	2.2e-05	2.8e-01	2.4e-05	4.5e-01	2.8e-05	9.5e-01
1000000	6.9e-10	2.0e-03	6.5e-09	1.9e-02	2.5e-07	3.3e-01

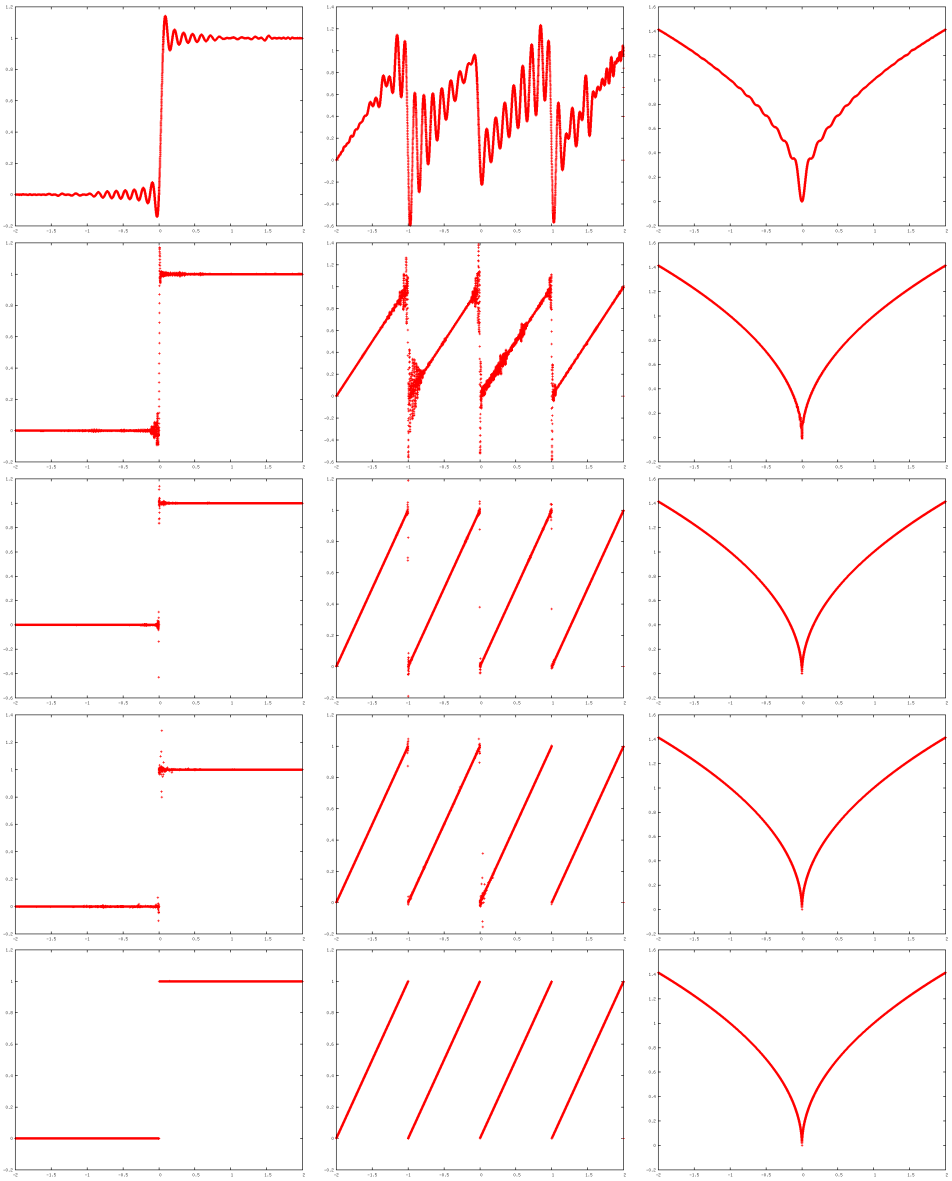

Figure 2: Polynomial reconstructions with FL points for the Heaviside, sawtooth, and $\sqrt{|x|}$ functions, using $m = 10001$ samples. From top to bottom: Polynomial degree $n = 100, 1000, 10000, 100000, 1000000$.

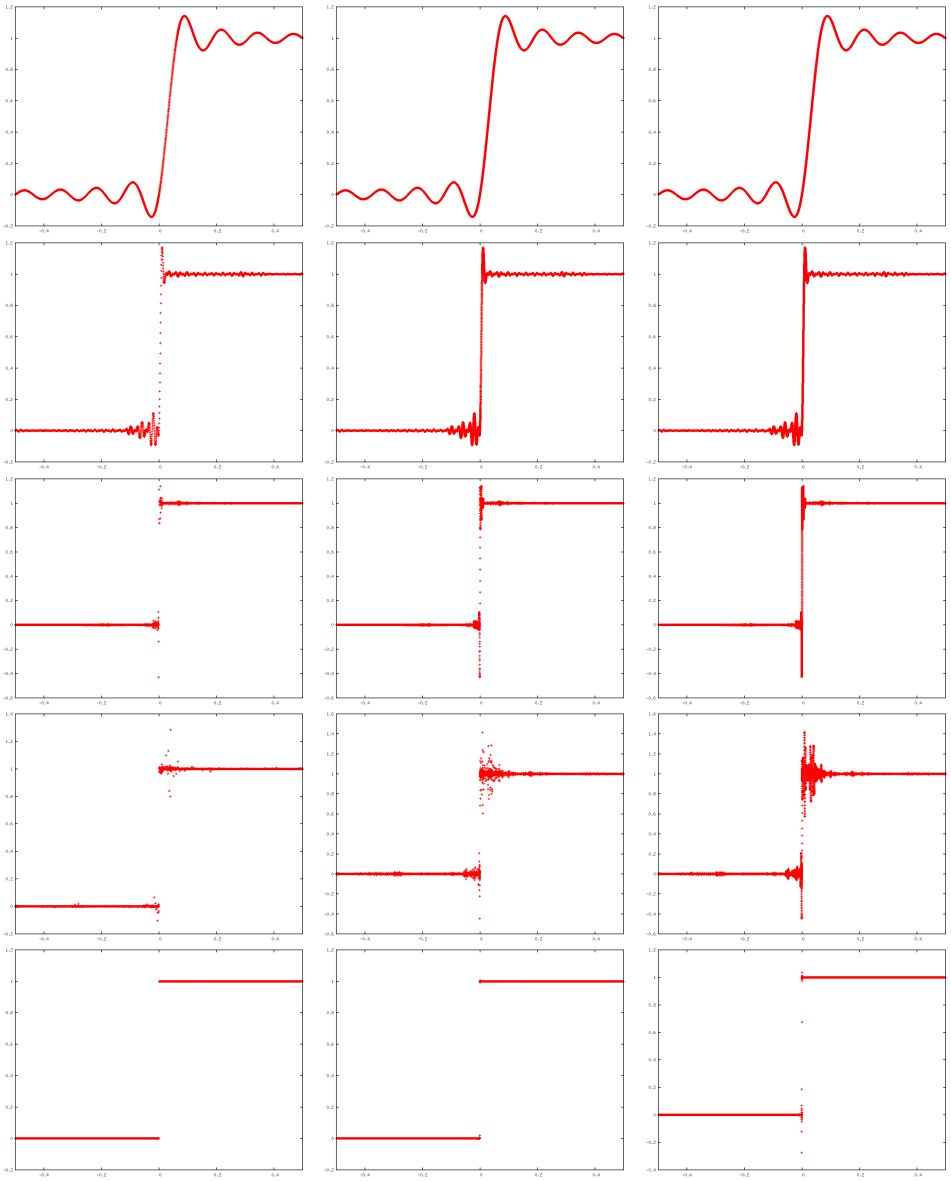

Figure 3: Polynomial reconstructions of the Heaviside function with different samplings. From left to right: $m = 10001, 100001, 1000001$ samples. From top to bottom: Polynomial degree $n = 100, 1000, 10000, 100000, 1000000$. All plots are given for zooming into the interval $x \in [-0.5, 0.5]$.