ISSN 2079-3316 PROGRAM SYSTEMS: THEORY AND APPLICATIONS vol. 9, No 4(39), pp. 93–116

CSCSTI 27.41.41
UDC 519.652.3

S. V. Znamenskij

Numerical evaluation of the interpolation accuracy of simple elementary functions

Abstract. The paper contains a comparison of the accuracy of the restoration of elementary functions by the values in the nodes for algorithms of low-degree piecewise-polynomial interpolation. The test results demonstrate in graphical form the advantages and disadvantages of the widely used cubic interpolation splines.

The comparison revealed that, contrary to popular belief, the smoothness of the interpolant is not directly related to the accuracy of the approximation. In the 20 different examples considered, the piecewise quadratic interpolation is rarely and only slightly inferior in the form of the used classical cubic splines, often by orders of magnitude better than many of them.

In several examples, the high interpolation error of simple functions on a fixed grid appears to be almost independent of the degree of the algorithm and the smoothness of the interpolant. The piecewise-linear interpolation unexpectedly appeared the most accurate in one of the examples.

A new problem arises: to find a local interpolation algorithm, accurately restoring any rational functions of the second order.

Key words and phrases: local interpolation, spline interpolation, convexity preserving, recovery precision.

2010 *Mathematics Subject Classification*: 65D07; 41A10, 41A15

Introduction

The problem of recovering a piecewise analytic on the segment $[a, b]$ real function $f(x)$ by the vector $\vec{y} = (y_0, \ldots, y_n)$ from its $n+1$ values $y_i = f(x_i)$ at the points $x_0 < \cdots < x_n, \ldots, n$ of the real line has a multi-millennial history [1] and various generalizations and applications. In the course of diverse studies, a great number of convincingly harmonious interpolation theories (see, for example, [2–6]), oriented on optimal algorithms for numerous applications, has grown.

10.25209/2079-3316-2018-9-4-93-116

We focus on the interpolation algorithms available with the source code, capable to recover simple dependencies by a few specified values. This context naturally arises in the analysis of scarce experimental data. We beleive that identifying the weak points of algorithms from this point of view can lead to the formulation of new problems, the solution of which could improve the quality of interpolation in other applications.

This investigation focuses on two practically significant interpolation quality criteria:

(1) *Locality* — dependence of interpolant only on values in the nearest l nodes on the right and l nodes on the left.

(2) *Accuracy of restoring the most simple and common functions.*

Locality is important when restoring piecewise-analytic functions of a single variable or functions with singular points on a grid near the interpolation segment, since it reliably localizes recovery errors caused by irregular behavior near singular points. In image processing, only locality provides clean scaling of pictures in which a high-contrast object is located against a pale background. When processing images, locality provides accurate scaling of photos, while streaming the signal, the radius of locality determines the delay time. When restoring from values on the grid, the piecewise-analytical functions of one variable or functions with singularities near the interpolation segment, it prevents the loss of restoration accuracy outside of singularities neighbourhoods.

The accuracy of the restoration of the most simple functions is an objective and verifiable quality attribute of interpolation. However, this attribute conceals a pitfall. Kolmogorov complexity obviously depends on a programming language, and there is no convincingly unambiguous answer to the questions, which of functions is more simple and commonly used:

— A polynomial or a linear-fractional function?

— Cosine or density of normal probability distribution?

Overlaying one graph on the other one and selecting the intersection points of the graphs, we obtain the initial data of the sample interpolation problem, which has two sharply different perfectly accurate solutions. no distinct best solution.

We see that, demanding from the interpolating algorithm a qualitative reconstruction of a simple in the intuitive understanding of the function, it is necessary to take into account similar ambiguous statements of the interpolation problem.

To identify the effect of the interpolant's smoothness, we consider a simple local quadratic continuous (not necessarily smooth) interpolating spline formula and evaluate its interpolation accuracy on the examples of simple elementary functions, analyzing specific situations arising during comparison.

1. Formula of piecewise-quadratic interpolation

Formulas of piecewise quadratic interpolation between nodes with given values differ in the method of obtaining the coefficient for the highest degree k_i in the Newton interpolation formula

$$P_i(x) = y_i + (x - x_i)(m_i + k_i(x - x_{i+1})),$$

where $m_i = \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$ is the first divided difference. The coefficient k_i is usually taken to be equal to the second divided difference $\Delta_i = \frac{m_{i+1} - m_i}{x_{i+2} - x_i}$, but more accurately can be found from the calculated interpolant value $y_m i = y_i + l_i(m_i + k_i l_i)$ in the middle of the segment $[x_i, x_i + 1]$, where $l_i = \frac{x_{i+1} - x_i}{2}$ is half the length of segment. Using the Lagrange interpolation formula over the four nearest nodes for getting the value, we obtain very simple piecewise quadratic interpolation formula:

$$f(x) = y_i + (x - x_i)\left(m_i + k_i(x - x_{i+1})\right), \tag{1}$$

where

$$k_i = \begin{cases} \Delta_0 + \dfrac{x_2 - l_0}{x_3 - x_0}(\Delta_1 - \Delta_0), & i = 0, \\ \Delta_i + \dfrac{l_i - x_{i+2}}{x_{i+2} - x_{i-1}}(\Delta_i - \Delta_{i-1}), & 0 < i < n - 1, \\ \Delta_{n-2} + \dfrac{l_{n-1} - x_{n-2}}{x_n - x_{n-3}}(\Delta_{n-3} - \Delta_{n-2}) & i = n - 1. \end{cases}$$

Although no exact references could be found, these simple formulas have undoubtedly already been used in some form over the thousand-year history of quadratic interpolation.

2. Shape preserving quadratic interpolation

The squareness of the interpolant provides easily verifiable exact conditions for the shape preservation.

REMARK 1. The method does not always preserve convexity, but the convexity preservation is easy to ensure by putting $k_i = 0$ for those $0 < i < n-1$, for which $\Delta_i \Delta_{i+1} \geq 0$, but computed k_i has a sign opposite to $\Delta_i + \Delta_{i+1}$.

REMARK 2. Non-conservation of monotony or sign is revealed by the exact condition $|2A_i^0| \leqslant \left|\frac{m_i}{x_{i+1}-x_i}\right|$. The joint preservation of the sign and monotony can be easily guaranteed by replacing, where this condition is violated, of the calculated value of A_i^0 to the value $\frac{\pm m_i}{2(x_{i+1})-x_i)}$ taken with the same sign.

REMARK 3. The preservation of the upper and lower bounds inside the segment of the image $[a, b]$ is controlled by the exact condition

$$-\left(\frac{\sqrt{b-y_i}+\sqrt{b-y_{i+1}}}{x_{i+1}-x_i}\right)^2 \leqslant 2A_i^0 \leqslant \left(\frac{\sqrt{y_i-a}+\sqrt{y_{i+1}-a}}{x_{i+1}-x_i}\right)^2$$

and is provided by replacing (just for segments where the condition is violated) of the calculated value of A_i^0 by the value of the nearest constraint.

Although these simple form preservation restrictions are accurate, and their corresponding code additions are simple, they are not included in the tests of this paper. In the considered statement of the problem, they rarely seem to change something, also the changes usually reduces the accuracy of the recovery.

3. Numerical evaluation of the interpolation quality

Numerical comparison of the results of the elementary functions interpolation was carried out on the maximum, average and average-quadratic error.

The integrals in the standard formulas for the mean and mean square were calculated using the trapezoid formula with 12 equally-spaced intermediate nodes on each interpolation segment, and the graphs were plotted using the same values.

Each drawing consists of two. The graphs of the function and its interpolants are on the upper half, and the graphs of interpolation errors on a scale allowing them to be seen and compared are lower.

A program that calculates deviations and draws graphics in LaTeX 2_ε TikZ/PGF is implemented in Perl and attached to the article.

For comparison with the described quadratic spline, all the most accessible implementations of cubic splines were used. The comparison includes practical algorithms implemented in actively developing libraries: open GSL (GNU Scientific Library) and multilingual Russian AlgLib, on which a variety of modern software is based.

Of these, only Steffen spline [9] (S) and non-smooth cubic spline (C), see, for example, [10] has the same locality radius $l = 2$ as tested quadratic Q, and only piecewise linear interpolation L has a smaller radius of locality.

For comparison with the results for non-local natural cubic spline (two types of boundary values were used for non-periodic interpolation problems — parabolic P, which exactly restores quadratic polynomials, and natural N, with zero values of the derivative at the ends).

The implemented in GSL algorithm of H. Akima (A) [8] is also included in the comparison. Other local splines using data from more than two adjacent nodes on each side, such as [7, 11–13], are not considered, since according to the results of testing in [14], the weakening of locality to $l = 3$ predictably allows for higher precision of interpolation of analytic functions.

3.1. Experiments on a uniform grid

The purpose of experiments on a uniform grid of interpolation nodes was to compare the behavior of different splines of the locality radius $l = 2$ on functions of different behavior, from simple, without local extremes and kinks, and ending with more complex, alternating descending-increasing sections, and then alternating directions of convexity.

3.1.1. *Functions without inflection points*

The interpolation errors of the exponent are visible on Figure 1, of the logarithm on Figure 2, of the simplest rational function $y = {}^{1}\!/_{x}$ on Figure 3. The three examples shows that in the absence of local extremum points and inflection points, the errors of A and Q almost coincide and is more than three times lower than that of the other splines.

Exactly the same situation is repeated with a function that has a semi-circle plot on Figure 4, except for the boundary sections where all interpolation formulas work roughly due to infinite derivatives at the extreme nodes.

It is important to note here that a natural cubic spline probably could demonstrate better results with a good choice of boundary conditions. It is not known, it is possible to do this without knowing the function itself. In

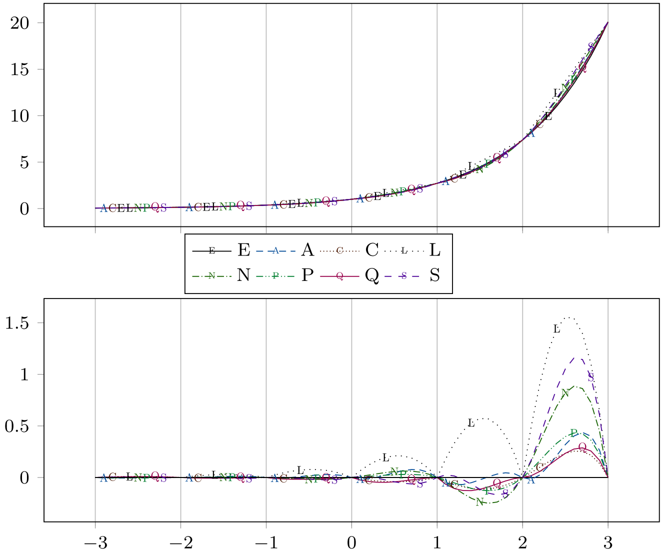

FIGURE 1. Interpolation of e^x on $[-3,3]$

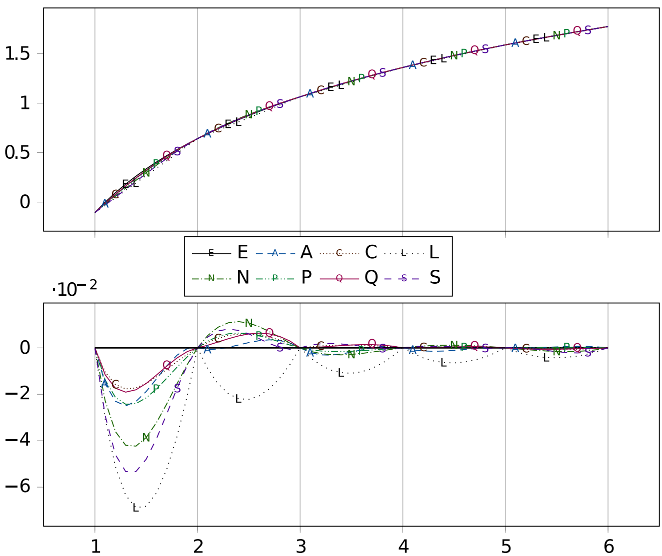

FIGURE 2. Interpolation of $\ln(x-0.1)$ on $[1,6]$

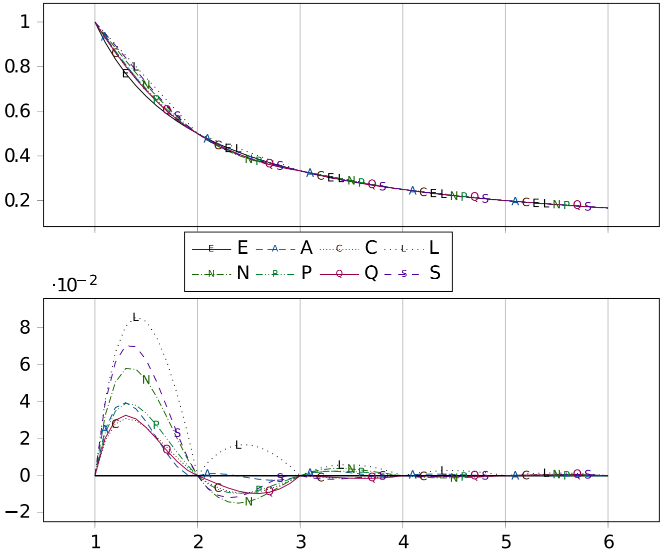

FIGURE 3. Interpolation of $1/x$ on $[1, 6]$

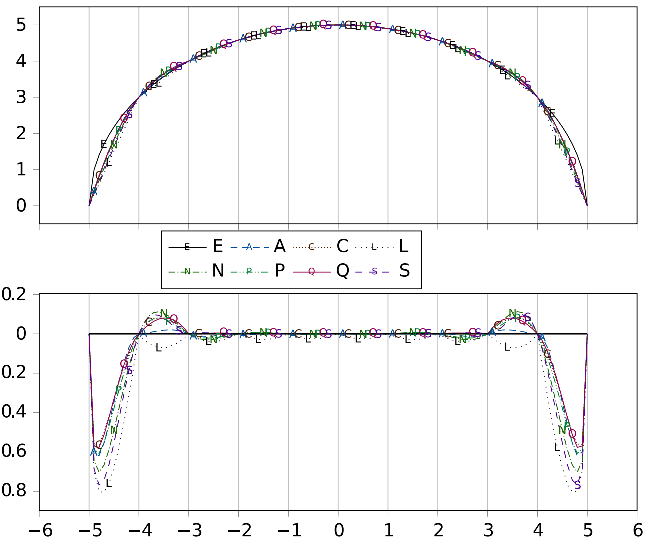

FIGURE 4. Interpolation of $\sqrt{25 - x^2}$ on $[-5, 5]$

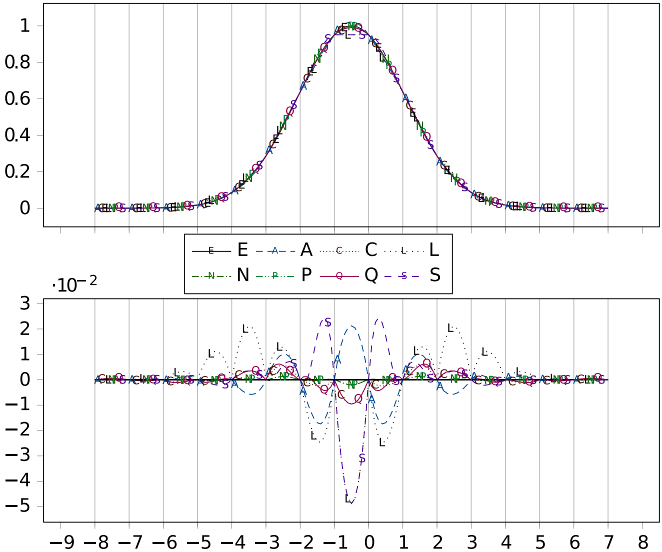

FIGURE 5. Interpolation of $e^{-0.1(x+0.5)^2}$ on $[-8, 7]$

the current version of GSL, the second derivative at the ends is set to zero (N), but a much better result is obtained with the parabolic task $f''(x_0) = \Delta_0$ and $f''(x_n) = \Delta_{n-2}$ at the three extreme points. Note that Q at the edges is determined by four points and has a visible advantage.

3.1.2. *Interpolation of density of normal distribution*

The classic test case for interpolation algorithms is the probability density function of the normal distribution. The overall picture is represented by the graphs on Figure 5–9 and significantly depends on the grid of interpolation nodes.

When the maximum point does not fall into the interpolation node (S on Figure 5), the methods aimed at preserving monotony sharply lose their accuracy, giving an increased error not only not in the extreme interval, but also at least in the neighboring with it (S on Figure 9). A number of them (for example, S to Figure 6) try to round the graph near the sharp maximum at the node (S to Figure 5).

Even when a maximum is exactly in a knot, the complex dependence of the comparison result on the pitch (Figure 6 versus Figure 7) is striking.

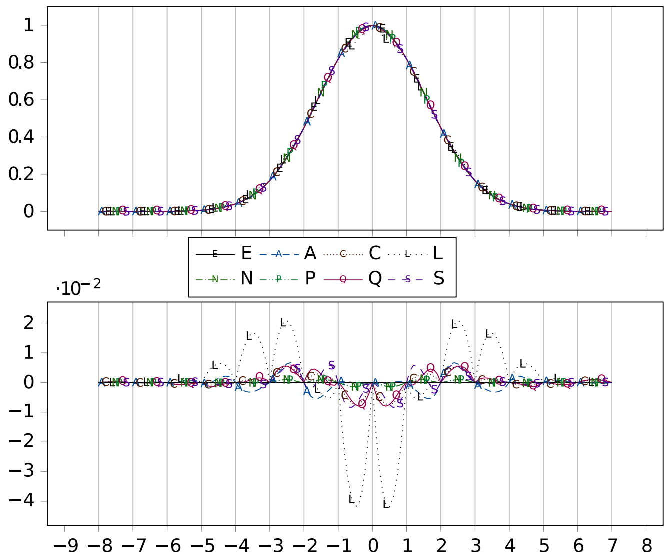

Figure 6. Interpolation of $e^{-0.1x^2}$ on $[-8, 7]$

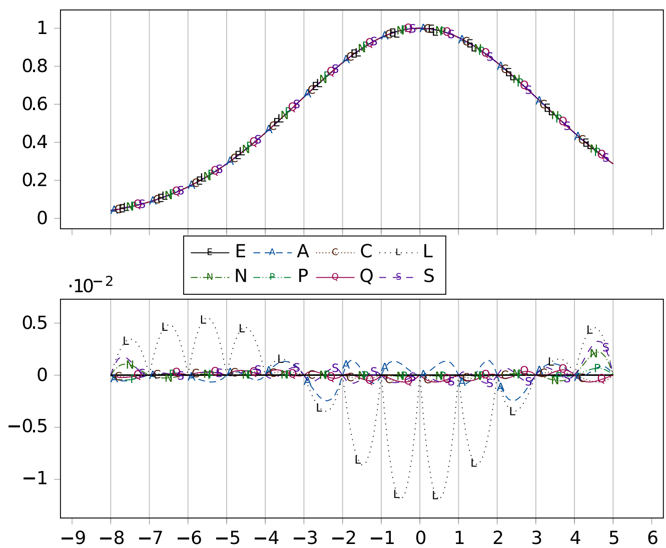

Figure 7. Interpolation of $e^{-0.05x^2}$ on $[-15.5]$

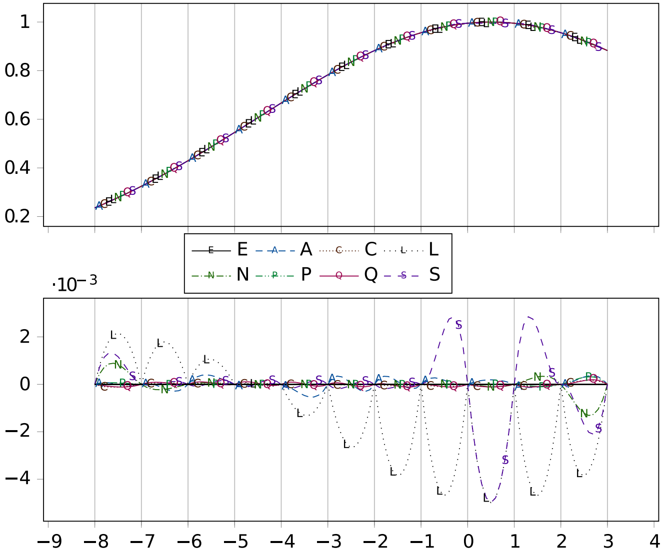

FIGURE 8. Interpolation of $e^{-0.01(x-0.5)^2}$ on $[-8.3]$

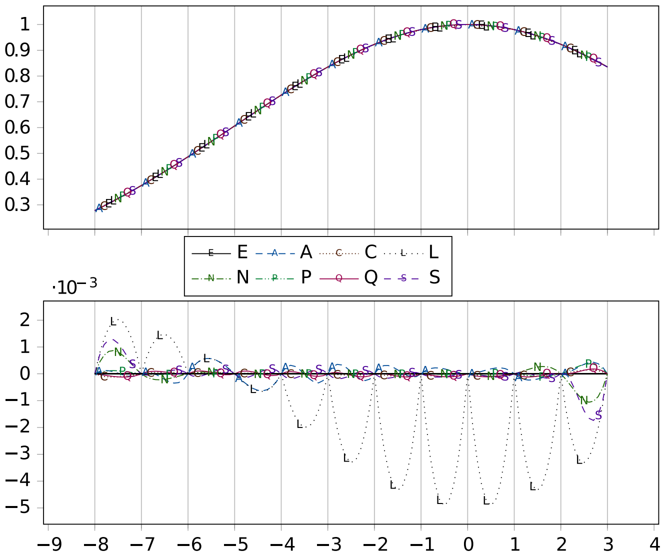

FIGURE 9. Interpolation of $e^{-0.01x^2}$ on $[-8, 3]$

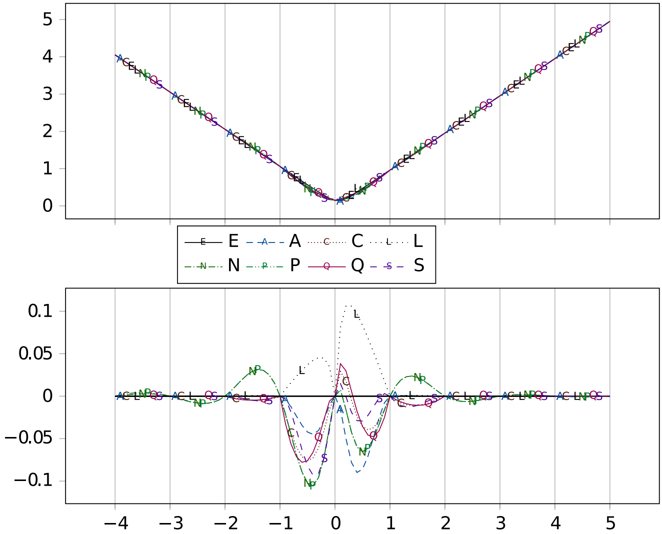

FIGURE 10. Interpolation of $\sqrt{0.02 + (x - 0.05)^2}$ on $[-4, 5]$

When decreasing the step, the interpolation segments become more similar, Figure 9.

In the considered examples, P, Q are also clearly in the lead among local ones, and the natural cubic spline on Figure 5, 6, 8 and 9 also corrupted by zero boundary conditions.

3.1.3. *Other functions with inflection points*

In many of the examples considered, the natural cubic spline with the correct choice of boundary conditions turns out to be much more accurate than local ones. However, accuracy may be lost with sharp bends. In the example Figure 10 the local P, Q are noticeably more accurate.

Unfortunately, this interpolation gain of the locality radius $l = 2$ is unstable to the grid. Significant differences in the ratio of errors for different configurations of nodes are clearly visible on Figure 11.

If the singular points are not close to the nodes, then the positive effect of non-smooth joining in the nodes is compensated by the impossibility of representing non-smoothness outside the nodes and the advantage of non-smooth spline over the natural one is lost on Figure 11.

However, the advantage of non-smooth splines over smooth local ones remains clear. It is also saved for sine, see Figure 12.

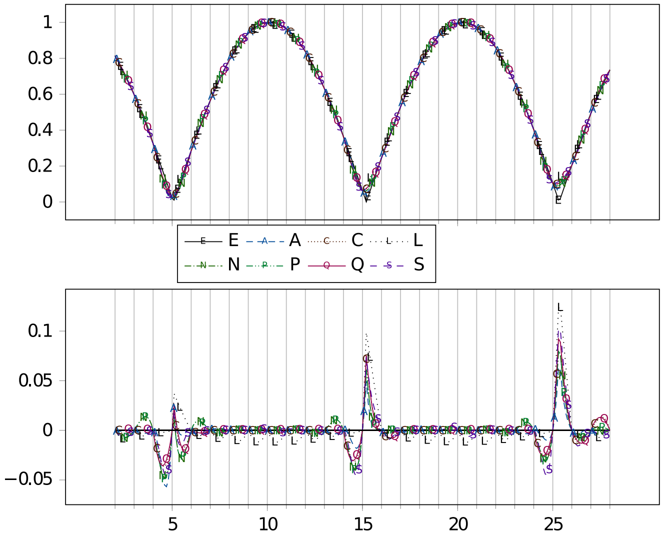

FIGURE 11. Interpolation of $|\cos(0.4x)|$ on $[-15.30]$

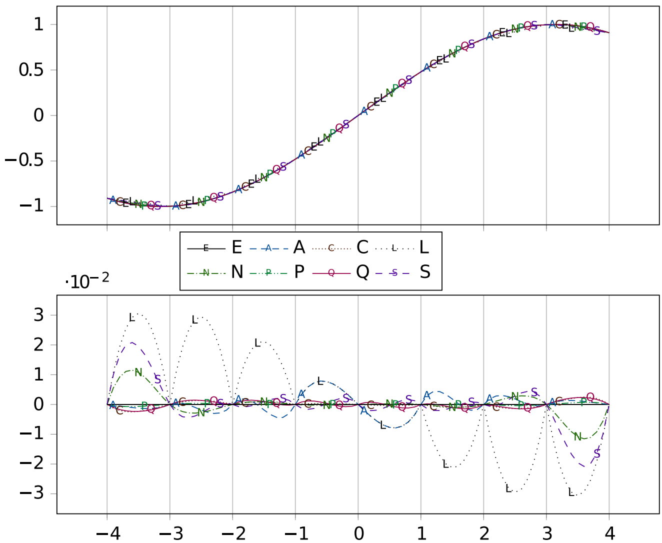

FIGURE 12. Interpolation of $\sin(x/2)$ on $[-4, 4]$

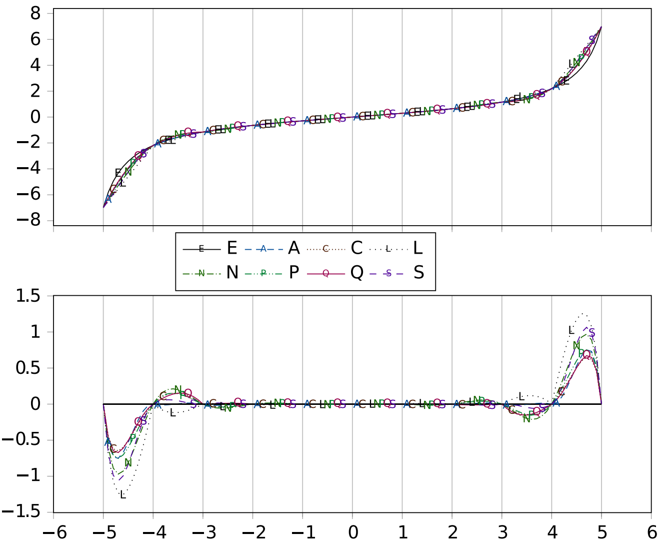

Figure 13. Interpolation of $\mathrm{arctg}(x + 0.7)$ on $[-4, 4]$

3.1.4. *Other functions with inflection points*

In many of the examples considered, the natural cubic spline with the correct choice of boundary conditions turns out to be much more accurate than local ones. However, accuracy may be lost with sharp bends. In the example Figure 10 the local P, Qare noticeably more accurate.

3.2. Interpolation on non-equispaced mesh

The question naturally arose about the specificity of the grid with uniform readings and a different selection of test functions. A search of the previously used set of tests for the accuracy of interpolation of elementary functions in published articles revealed only a test in [11] "Control Data 7600", including an uneven grid with nodes -2.95, -2.6, -2.1, -1.8, -1.4, - 1.0, -0.75, -0.3, -0.05, 0.2, 0.55, 0.9, 1.25, 1.6, 1.7, 2.1, 2.4, 3.0 and 5 tested functions.

The results of this test, presented in Figure 16–20, clearly confirmed the leadership of P, Q among the splines of the locality radius $l < 3$. On this set of examples, they successfully compete even with natural splines, repeatedly confidently surpassing them exactly.

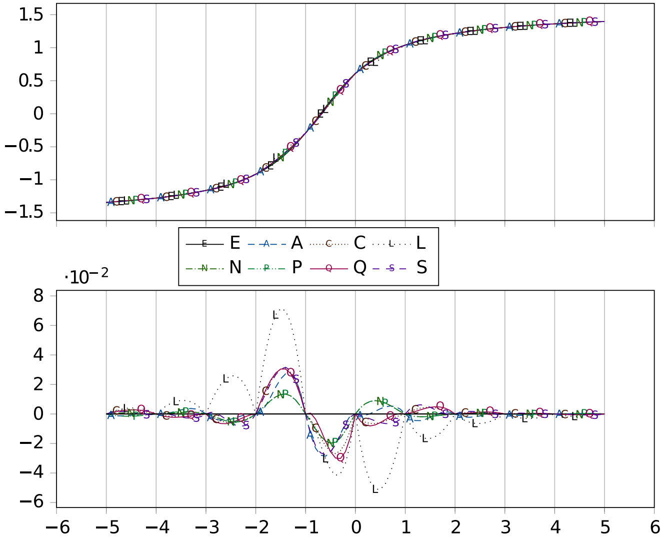

FIGURE 14. Interpolation of $\tan(x/6.5)$ on $[-10, 10]$

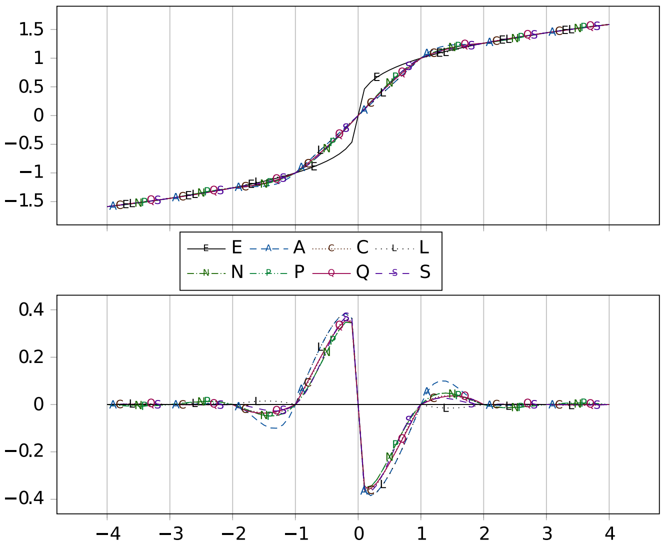

FIGURE 15. Interpolation of $\sqrt[3]{x}$ on $[-4, 4]$

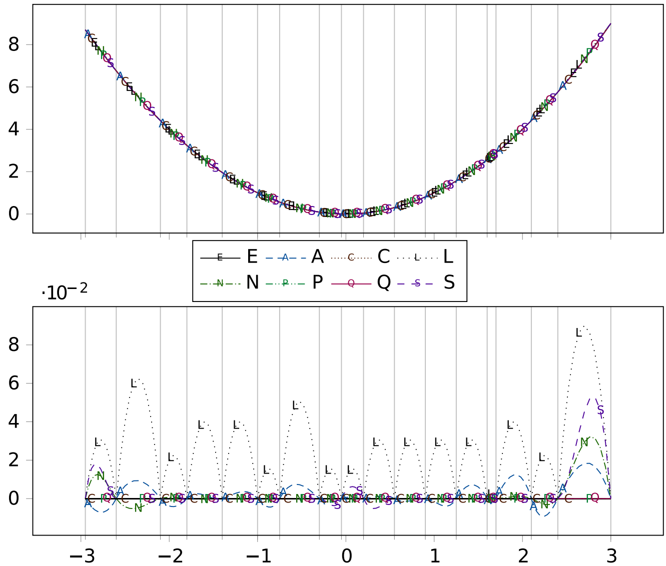

Figure 16. Interpolation of x^2 on CD 7600 mesh

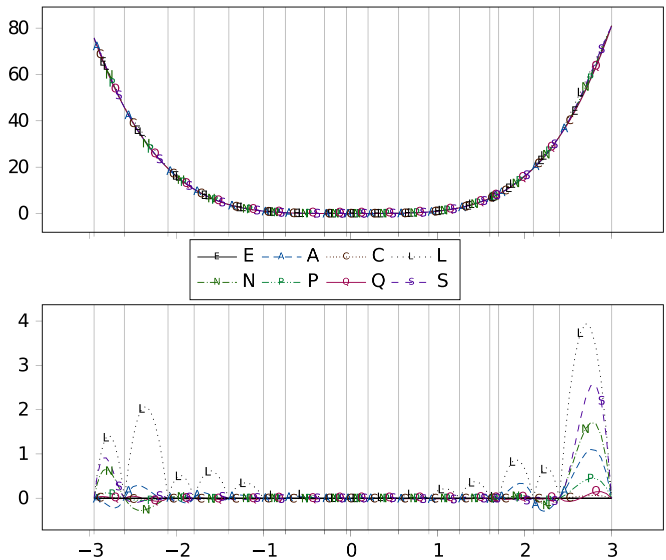

Figure 17. Interpolation of x^4 on the CD 7600 grid

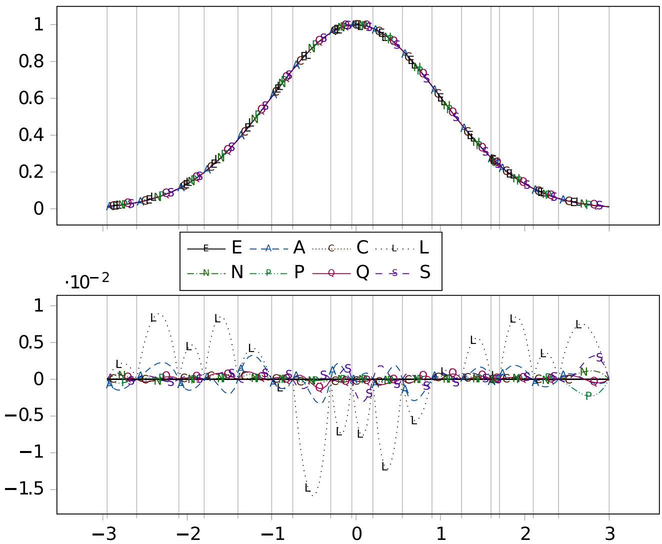

FIGURE 18. Interpolation of $\exp(x^2/2)$ on the grid CD 7600

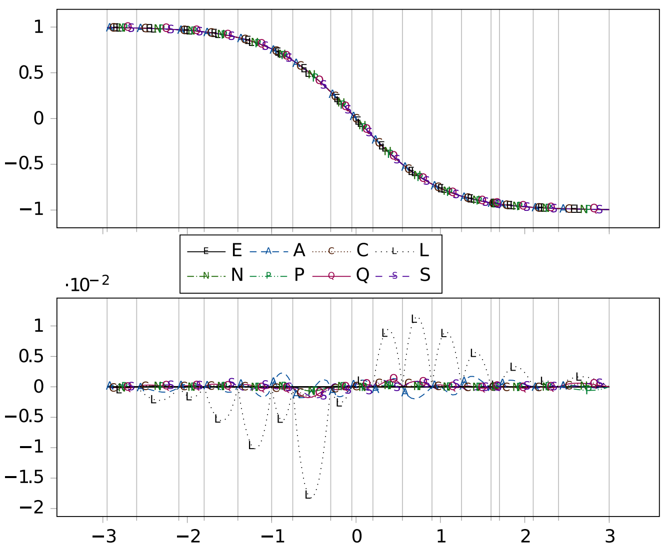

FIGURE 19. Interpolation of $\tanh(x)$ on the CD 7600 grid

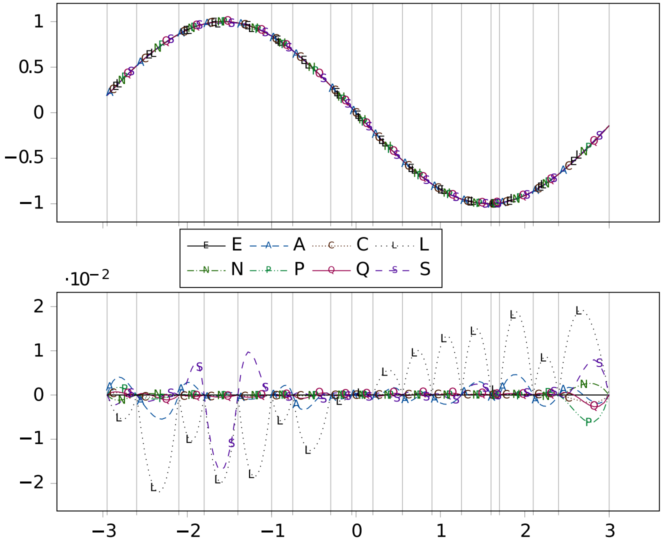

Figure 20. Interpolation of $\sin(x)$ on the CD 7600 grid

3.3. Test Summary

Numerical evaluations of the approximation accuracy achieved in the considered examples are presented in three tables. The maximum percentage errors are given in Table 1, the average percentage errors in Table 2, and the root mean square percentage errors are in Table 3. The color highlights the values **more than twice** superior to the best of the (other) splines of the locality in question: less than twice; worse by 2–5 times; worse by 5–10 times; **worse 10 or more times**.

3.3.1. *Preliminary results*

The tables, especially the last one, show that in any tested situation (grid of nodes, function) the quadratic spline errors have not exceed the doubling of any of the other local splines errors with an accuracy superiority over all including the non-local natural spline in more than 20% of cases. In most tests at least a twofold superiority is shown of P and Q over the other splines of the locality radius $l < 3$.

TABLE 1. Maximum percentage errors of interpolation

mesh	formula	P	N	A	S	L	C	Q
-3..3	e^x	2.38	4.87	2.41	6.37	8.56	1.41	1.58
1..6	$\ln(x-0.1)$	1.38	2.41	1.42	3.03	3.92	1.01	1.09
1..6	$1/x$	3.91	5.79	3.95	7.03	8.57	3.09	3.26
-5..5	$\sqrt{16-x^2}$	12.14	14.04	12.38	15.36	16.12	11.28	11.53
-8..7	$e^{-0.1(x+0.5)^2}$	**0.20**	**0.20**	2.13	4.88	4.88	0.96	0.96
-8..7	$e^{-0.1x^2}$	**0.15**	**0.15**	0.66	0.83	4.19	0.74	0.80
-8..5	$e^{-0.05x^2}$	0.07	0.22	0.25	0.33	**1.20**	0.07	0.07
-8..3	$e^{-0.01(x-0.5)^2}$	0.03	0.13	0.05	**0.50**	**0.50**	0.02	0.02
-8..3	$e^{-0.01x^2}$	0.04	0.11	0.06	**0.17**	**0.49**	0.01	0.02
-4..5	$\sqrt{0.02+(x-0.05)^2}$	2.17	2.17	1.85	1.92	2.18	1.57	1.60
2..28	$\lvert\cos(0.4x)\rvert$	7.97	7.96	6.37	10.23	12.41	9.07	9.22
-4..4	$\sin(x/2)$	**0.07**	1.15	0.79	2.08	**3.06**	0.25	0.23
-5..5	$\tan(x/3.3)$	10.78	13.89	10.77	15.24	17.96	9.30	9.62
-5..5	$\arctan(x+0.7)$	1.42	1.42	2.06	2.27	5.13	2.21	2.18
-4..4	$\sqrt[3]{x}$	21.91	21.90	24.24	23.27	24.24	22.75	22.38
CD	x^2	0	**0.37**	**0.21**	**0.61**	**1.03**	0	0
CD	x^4	0.59	**2.26**	**1.45**	**3.38**	**5.21**	0.08	0.20
CD	$\exp(x^2/2)$	0.23	0.11	0.33	0.32	**1.58**	0.08	0.13
CD	$\tanh(x)$	**0.07**	**0.07**	0.23	0.18	**1.84**	0.17	0.18
CD	$\sin(x)$	0.62	0.26	0.55	**1.70**	**2.21**	0.17	0.27
-5..5	$\frac{x}{1+5x^2}$	47.85	47.85	59.03	50.71	59.03	52.31	51.83
-15..15	$\sqrt[3]{\cos x}$	75.06	71.73	75.91	70.88	63.22	75.72	76.07

Table 2. Mean percentage errors of interpolation

mesh	formula	P	N	A	S	L	C	Q
-3..3	e^x	1.96	3.95	1.66	4.35	8.53	1.46	1.46
1..6	$\ln(x-0.1)$	0.37	0.67	0.31	0.70	1.36	0.29	0.29
1..6	$1/x$	1.71	2.63	1.28	2.73	4.04	1.41	1.41
-5..5	$\sqrt{16-x^2}$	2.00	2.51	1.67	2.54	3.23	1.79	1.79
-8..7	$e^{-0.1(x+0.5)^2}$	**0.10**	**0.10**	1.54	1.78	3.23	0.61	0.64
-8..7	$e^{-0.1x^2}$	**0.11**	**0.11**	0.56	0.61	3.03	0.63	0.65
-8..5	$e^{-0.05x^2}$	**0.01**	0.04	0.12	0.09	**0.66**	0.04	0.05
-8..3	$e^{-0.01(x-0.5)^2}$	0.00	0.02	0.02	**0.11**	**0.25**	0.01	0.01
-8..3	$e^{-0.01x^2}$	0.01	0.02	0.03	0.03	**0.25**	0.01	0.01
-4..5	$\sqrt{0.02+(x-0.05)^2}$	0.72	0.71	0.35	0.36	0.46	0.39	0.42
2..28	$\lvert\cos(0.4x)\rvert$	1.23	1.22	0.71	1.31	1.66	1.09	1.10
-4..4	$\sin(x/2)$	**0.03**	0.34	0.34	0.61	**2.07**	0.12	0.13
-5..5	$\tan(x/3.3)$	8.27	11.03	6.02	9.25	13.52	7.05	7.05
-5..5	$\arctan(x+0.7)$	0.32	0.31	0.41	0.48	1.42	0.48	0.49
-4..4	$\sqrt[3]{x}$	4.90	4.83	6.25	4.51	5.34	4.89	4.89
CD	x^2	0	**0.10**	**0.15**	**0.14**	**0.88**	0	0
CD	x^4	0.24	**0.97**	**0.81**	**1.26**	**3.77**	0.06	0.11
CD	$\exp(x^2/2)$	0.05	0.03	0.25	0.15	**1.07**	0.05	0.07
CD	$\tanh(x)$	**0.01**	**0.01**	0.08	0.04	**0.49**	0.04	0.04
CD	$\sin(x)$	0.08	0.04	0.23	0.31	**1.27**	0.04	0.05
-5..5	$\frac{x}{1+5x^2}$	15.15	15.05	18.95	13.39	15.92	15.15	15.15
-15..15	$\sqrt[3]{\cos x}$	15.41	15.03	14.27	12.13	16.62	15.50	15.70

TABLE 3. Root mean square percentage of interpolation errors

mesh	formula	P	N	A	S	L	C	Q
-3..3	e^x	0.23	**0.48**	0.21	**0.60**	**0.91**	0.15	0.16
1..6	$\ln(x-0.1)$	0.06	0.11	0.06	0.14	**0.20**	0.05	0.05
1..6	$1/x$	0.29	0.44	0.27	0.52	0.68	0.24	0.24
-5..5	$\sqrt{16-x^2}$	0.43	0.52	0.42	0.57	0.65	0.40	0.40
-8..7	$e^{-0.1(x+0.5)^2}$	**0.01**	**0.01**	0.16	**0.25**	**0.31**	0.06	0.06
-8..7	$e^{-0.1x^2}$	**0.01**	**0.01**	0.06	0.07	**0.31**	0.06	0.06
-8..5	$e^{-0.05x^2}$	**0.00**	0.01	**0.01**	0.01	**0.07**	0.00	0.00
-8..3	$e^{-0.01(x-0.5)^2}$	0.00	**0.00**	0.00	**0.02**	**0.03**	0.00	0.00
-8..3	$e^{-0.01x^2}$	0.00	**0.00**	**0.00**	**0.01**	**0.03**	0.00	0.00
-4..5	$\sqrt{0.02+(x-0.05)^2}$	0.11	0.11	0.08	0.08	0.10	0.08	0.08
2..28	$\lvert\cos(0.4x)\rvert$	0.22	0.22	0.17	0.27	0.31	0.23	0.23
-4..4	$\sin(x/2)$	**0.00**	**0.05**	0.04	**0.10**	**0.23**	0.01	0.01
-5..5	$\tan(x/3.3)$	1.17	1.54	1.08	1.58	2.07	1.01	1.03
-5..5	$\arctan(x+0.7)$	**0.05**	**0.05**	0.08	0.09	0.21	0.08	0.09
-4..4	$\sqrt[3]{x}$	0.93	0.93	1.13	0.97	1.10	0.99	0.98
CD	x^2	0	**0.02**	**0.02**	**0.03**	**0.08**	0	0
CD	x^4	**0.04**	**0.15**	**0.10**	**0.22**	**0.42**	0.01	0.01
CD	$\exp(x^2/2)$	0.01	0.01	**0.02**	**0.02**	**0.11**	0.01	0.01
CD	$\tanh(x)$	**0.00**	**0.00**	0.01	0.01	**0.07**	0.01	0.01
CD	$\sin(x)$	**0.02**	0.01	**0.03**	**0.06**	**0.15**	0.01	0.01
-5..5	$\frac{x}{1+5x^2}$	2.68	2.68	3.45	2.76	3.30	2.93	2.92
-15..15	$\sqrt[3]{\cos x}$	2.60	2.53	2.63	2.38	2.58	2.64	2.64

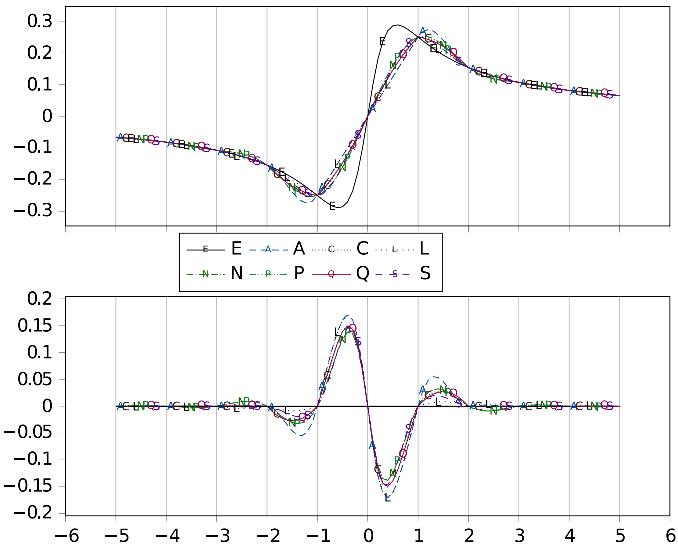

FIGURE 21. Interpolation of $\frac{x}{1+5x^2}$

A close examination of the graphs shows that if there are a few nodes (or the function changes frequently), then the precision of the piecewise-quadratic interpolant differs slightly from the accuracy of smooth natural splines.

At the same time, the areas where simple functions interpolate extremely roughly are highlighted. In the examples considered, these are neighborhoods of singular points of a function and a place where the one-side or second derivative many times exceeds the mean values or is infinite (knee or vertical tangent on the graph). These are friendly bursts of error in the extreme areas in the examples Figure 4, 13 and inside in the examples Figure 13–17.

Note that in this study only linear interpolation algorithms on a polynomial basis were considered. There was a hypothesis that interpolation by such algorithms is difficult at these situations. To test it, consider more expressive functions Figure 21 and 22.

It is clearly seen that the almost identical errors of all algorithms reached and exceeded 50%. Worse, in the last figure, the smallest error (both maximum and root-mean-square) was shown by piecewise linear interpolation. This shows that the considered formulation of the

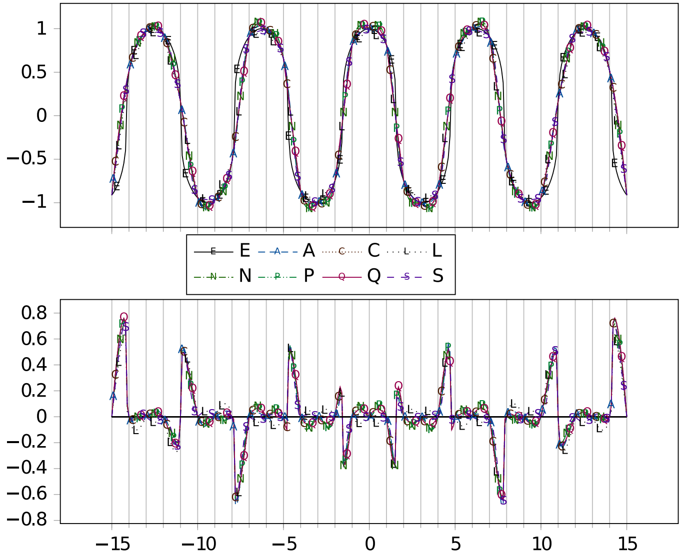

FIGURE 22. Interpolation of $\sqrt[3]{\cos(x)}$

interpolation problem will not solve linear algorithms on a polynomial basis.

An interpolation algorithm is required that accurately restores all (more precisely, almost all) rational functions of the second degree. It cannot be linear, since the linear span of the set of rational functions of the second (and even of the first degree) is infinite-dimensional. In publications on the theory of interpolation, the ideas of such an algorithm are not visible. It could affect the efficiency of various applications of approximation theory methods, including nonlinear regression and extrapolation.

This idea indirectly confirms the well-known possibility of significantly more accurate than polynomials approximation of algebraic and transcendental functions by rational functions. Such a possibility is effectively illustrated, for example, in [**15**].

Conclusions

1. The accuracy of recovery in areas with large derivatives is extremely low and does not depend on the complexity of the considered algorithms. The search for an algorithm that perfectly restores rational functions of the second degree is an interesting new problem.

2. The smoothness of the interpolants and the preservation of monotonicity (convexity, sign) do not guarantee increased interpolation accuracy. The tasks of interpolation, in which these qualities are not required, can be solved in simpler and more economical ways.
3. With high requirements for spline locality (when the radius of locality is less than 3), the considered simple quadratic spline is comparable in accuracy to cubic ones.

References

[1] E. Meijering. "A chronology of interpolation: from ancient astronomy to modern signal and image processing", *Proceedings of the IEEE*, **90**:3 (2002), pp. 319–342. URL doi ↑93

[2] P. Thévenaz, T. Blu, M. Unser. "Interpolation revisited", *IEEE Transactions on Medical Imaging*, **19**:7 (2000), pp. 739–758. URL doi ↑93

[3] B. I. Kvasov. "Monotone and convex interpolation by weighted cubic splines", *Comput. Math. Math. Phys.*, **53**:10 (2013), pp. 1428–1439. doi ↑93

[4] K. Yu. Osipenko. "On optimal recovery methods in Hardy–Sobolev spaces", *Sb. Math.*, **192**:2 (2001), pp. 225–244. doi ↑93

[5] M. Fuentes. "A high frequency kriging approach for non-stationary environmental processes", *Environmetrics*, **12**:5 (2001), pp. 469–483. doi ↑93

[6] N. C. Gabrielides, N. S. Sapidis. "C^1 sign, monotonicity and convexity preserving Hermite polynomial splines of variable degree", *Journal of Computational and Applied Mathematics*, **343** (2018), pp. 662–707. doi ↑93

[7] H. Akima. "A method of univariate interpolation that has the accuracy of a third-degree polynomial", *ACM Transactions on Mathematical Software*, **17**:3 (1991), pp. 341–366. doi ↑97

[8] H. Akima. *A method of smooth curve fitting*, ESSA Tech. Rep. ERL 101-ITS 73, U.S. Government Printing Office, Washington, D.C., 1969. ↑97

[9] M. Steffen. "A simple method for monotonic interpolation in one dimension", *Astron. Astrophys.*, **239** (1990), pp. 443–450. ↑97

[10] O. Niemitalo. *Polynomial interpolators for high-quality resampling of oversampled audio*, 2001. URL ↑97

[11] T. R. Ellis, D. H. McLain. "Algorithm 514: A new method of cubic curve fitting using local data [E2]", *ACM Transactions on Mathematical Software*, **3**:2 (1977), pp. 175–179. doi ↑97,105

[12] A. D. Maude. "Interpolation—mainly for graph plotters", *The Computer J.*, **16**:1 (1973), pp. 64–65. doi ↑97

[13] J. M. Hyman. "Accurate monotonicity preserving cubic interpolation", *SIAM J. Sci. and Stat. Comput.*, **4**:4 (1983), pp. 645–654. doi ↑97

[14] H. T. Huynh. "Accurate monotone cubic interpolation", *SIAM J. Numer Anal.*, **30**:1 (1993), pp. 57–100. ↑$_{97}$

[15] R. Schaback. "Adaptive rational splines", *Constr. Approx.*, **6**:2 (1990), pp. 167–179. ↑$_{114}$

Received	05.10.2018
Revised	24.10.2018
Published	14.11.2018

Recommended by *prof. A. M. Tsirlin*

Sample citation of this publication:

Sergej Znamenskij. "Numerical evaluation of the interpolation accuracy of simple elementary functions". *Program Systems: Theory and Applications*, 2018, **9**:4(39), pp. 93–116. 10.25209/2079-3316-2018-9-4-93-116

URL http://psta.psiras.ru/read/psta2018_4_93-116.pdf

The same article in Russian: 10.25209/2079-3316-2018-9-4-69-92

About the author:

Sergej Vital'evich Znamenskij

Research interests migrated from research in Functional Analysis, Complex Analysis and finite-dimensional Projective Geometry (analogues of Convexity) to the foundations of Collaborative Software Development, Similarity Theory and Intertpolation Theory

iD 0000-0001-8845-7627

e-mail: svz@latex.pereslavl.ru

Эта же статья по-русски: 10.25209/2079-3316-2018-9-4-69-92