Computer Science • 21(4) 2020 https://doi.org/10.7494/csci.2020.21.4.3932

Roman Dębski

REAL-TIME INTERPOLATION OF STREAMING DATA

Abstract *One of the key elements of the real-time C^1-continuous cubic spline interpolation of streaming data is an estimator of the first derivative of the interpolated function that is more accurate than those based on finite difference schemas. Two such greedy look-ahead heuristic estimators, based on the calculus of variations (denoted as MinBE and MinAJ2), are formally defined (in closed form), along with the corresponding cubic splines that they generate. They are then comparatively evaluated in a series of numerical experiments involving different types of performance measures. The presented results show that the cubic Hermite splines generated by heuristic MinAJ2 significantly outperformed those that were based on finite difference schemas in terms of all of the tested performance measures (including convergence). The proposed approach is quite general. It can be directly applied to streams of univariate functional data like time-series. Multi-dimensional curves that are defined parametrically (after splitting) can be handled as well. The streaming character of the algorithm means that it can also be useful in processing data sets that are too large to fit in the memory (e.g., edge computing devices, embedded time-series databases).*

Keywords streaming algorithm, online algorithm, spline interpolation, cubic Hermite spline

Citation Computer Science 21(4) 2020: 513–532

1. Introduction

Cubic splines are in widespread use in numerical analysis, statistics, computer graphics, computer-aided design (CAD), and many other disciplines of science and engineering. A significant number of their applications are related to interpolation of sets of discrete data. Well-known natural cubic splines (and B-splines) have proven effective in many of these scenarios, yet they are not always applicable. This is often because of their *global* character – they require the solution of a system of linear equations whose coefficients depend on a whole interpolated data set.

However, access to a whole data set is impossible in a number of cases (e.g., streaming data, big data sets). This scenario is becoming increasingly frequent in contemporary systems, as many sources (e.g., sensor networks, scientific data) produce data continuously; often, the data should be processed in an online fashion. Examples include real-time animation (in computer games or scientific simulations), real-time control, and decision support systems.

Taking this into account, effective streaming interpolators should be entirely local and operate in an online fashion. A classic example of such a local interpolant is a cubic Hermite spline, which is made up of cubic polynomial segments that are joined together with common slopes at the data points. If the slopes are not given, they must be estimated – usually with the use of finite difference schemas. Since the accuracy of this approximation method may be insufficient in some cases, a more robust method is desirable.

The aim of this paper is to introduce such a method as the foundation of the real-time algorithm that constructs C^1-continuous cubic splines in an *online* fashion. The proposed approach is quite general. It can be directly applied to streams of univariate functional data (like time-series). Multi-dimensional curves that are defined parametrically (after splitting) can be handled as well. The streaming character of the algorithm means that it can also be useful in processing data sets that are too large to fit in the memory (e.g., edge computing devices, embedded time-series databases).

The main contributions of this paper are the following:

- two greedy look-ahead heuristic estimators of the first derivative of the interpolated function (Sections 4.1 and 4.2);
- a streaming interpolator that maps a stream of data points to a stream of cubic spline segments (using one of these heuristic estimators), which form a C^1-continuous interpolant (Section 4.3);
- numerical results that demonstrate the effectiveness of the proposed heuristics as well as the algorithm itself (Section 5).

The remainder of this paper is organized as follows. In the next section, the problem statement is given. Following this, the proposed solution is described. Then, the solution is evaluated, and the obtained results are presented and discussed. The last two sections contain a related work overview and the conclusions of the study, respectively.

2. Related work

There is an extensive amount of literature on interpolating splines[1] that covers a number of disciplines of science and engineering. To understand the recent developments in this field, some historical context (given in a number of seminal papers) is helpful.

Historical context. The term *interpolation*[2], according to [26], was initially used in the mathematical sense in [35]. Among the first contributors to the topic of interpolation were Newton, Lagrange, Euler, Gauss, Laplace, and Cauchy. *Hermite interpolation* (finding a polynomial of which also the first few derivatives assume pre-specified values at given points) was first discussed in [19] and then generalized by [5]. A more thorough treatment of the history of interpolation theories and techniques from the earliest times to the present day can be seen in an excellent review paper by [26].

According to [15], the first reference to a *spline* (a flexible and mechanical strip of wood that is used to draw smooth curves) appears to be [28]. The mathematical equivalent of a mechanical spline is a *spline curve*. At first, this corresponded to a curve that minimized a specific functional (elastic bending energy); however, spline curves are generally regarded as piece-wise polynomial curves with certain smoothness properties these days. This new idea of a spline curve (and a basic k^{th}-order spline curve, nowadays known as a *B-spline*) was presented in [30]. This seminal work laid the mathematical foundations for spline approximation and interpolation. More information on splines can be found in [7, 21, 31–33], for example.

Selected recent developments: computer-aided geometric design. The generalization of B-splines to *nonuniform rational B-splines* (NURBS) has become the standard curve form in the CAD/CAM industry [34]. A special case of NURBS is given by *rational Bézier curves* [16]. The concepts of *Bézier curves* [3, 17] and *Casteljau's curves* [10] are based on the use of *control polygons*, which involve both points on the curve and the points close to it. Catmull-Rom splines (as local interpolants) and blending functions are described in [9], for instance. Another piece-wise-defined curve scheme (*biarc*) was proposed in [6]. Biarcs are piece-wise circular arcs pieced together with tangent continuity; hence, these can be thought of as *circle splines*. More information about curves in computer-aided geometric design can be found in [15].

Selected recent developments: real-time applications and trajectory optimization. An approximation of a linearly varying curvature by three cubic curve segments (between four points) was described in [27]. The authors named their approach *the curvic interpolation*. A similar representation of a curve but used in an online algorithm for the generation of minimum time joint trajectories for industrial manipulators was presented in [2]. Their approach, which they named the *3-Cubic (method)*, gives continuity of acceleration at the junction points by imposing zero acceleration (at the starting and terminating ends of the segment).

[1] And curve fitting.
[2] From the Latin *interpolare*.

Interpolating splines in the context of real-time/online applications were also researched in [8, 14, 18], and [29]. Online trajectory generation methods in robotic systems are described in [22].

An excellent overview of trajectory-planning methods can be seen in [4]. Among other topics, the authors describe many representations of trajectories: elementary ones like polynomial (of different degrees), trigonometric, or exponential as well as their different compositions and multi-point variants (including different splines). An extension of the spline-based representation to a family of trajectories (*multi-spline*) used in a parallel dynamic programming-based optimization algorithm was proposed in [12]. This can be treated as an extension of the method presented in [11, 13]. Higher-order splines (of a degree of five or seven) are detailed in an interpolation context in [20, 23–25] and in a trajectory optimization context (*3-4-5* and *4-5-6-7* interpolation polynomials) in [1].

3. Problem statement

Consider a stream S_p (finite or infinite) of data points $P_0, P_1, \ldots, P_k, \ldots$ that arrive (or are accessed) sequentially and describe an underlying signal $\mathbf{F}(q)$, $q \in \mathbb{R}$. This signal can represent a trajectory or a curve that is defined parametrically in the following way[3]:

$$\mathbf{F}(q) = (F_1(q), F_2(q), \ldots, F_n(q)), \ q \in \mathbb{R} \tag{1}$$

In such a case, each data point P_k can take the following form:

$$P_k = (q_k, F_1(q_k), \ldots, F_n(q_k)) \tag{2}$$

where $k = 0, 1, 2 \ldots$, and it is assumed that $q_k < q_{k+1} < q_{k+2} \ldots$ If $n > 1$, input stream S_p can be split (as a prepossessing step) into n streams, $S_p^{(1)}, \ldots, S_p^{(n)}$, each describing the characteristic of the interpolated trajectory/curve in one dimension (see Figure 1) and defined as $S_p^{(i)} = P_0^{(i)}, P_1^{(i)}, \ldots, P_k^{(i)}, \ldots$, where $P_k^{(i)} = (q_k, F_i(q_k))$, $i = 1, \ldots, n$ and $k = 0, 1, 2 \ldots$

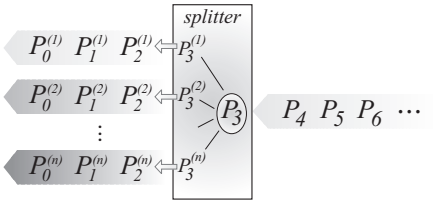

Figure 1. Stream splitter as interpolation pre-processor that changes original multi-dimensional problem to series of one-dimensional problems, which can then be solved independently

[3] $n = 3$ for the 3D space.

Following this, each of these new streams can be interpolated independently (possibly in parallel); finally, the results can be combined.

Remark 1. (Problem reduction). The original multi-dimensional problem can be reduced to a one-dimensional problem.

Let $S_D = (D_0, D_1, D_2, \dots)$ be one of these new streams, and assume that each data point D_k takes the following form:

$$D_k = (q_k, f(q_k)) \tag{3}$$

where $k = 0, 1, 2 \dots$ and $f(\cdot)$ stands for the component function that corresponds to stream S_D (i.e., one of $F_1, F_2, \dots$). Then, the problem statement can be formulated in the following manner:

Problem. (statement). Given a stream of data points $S_D = (D_0, D_1, D_2, \dots)$ where $D_k = (q_k, f_k))$, $q_k < q_{k+1} < q_{k+2} \dots$, and $k = 0, 1, 2 \dots$, find the corresponding stream of cubic spline segments that form a C^1-continuous interpolant of S_D (see Figure 2).

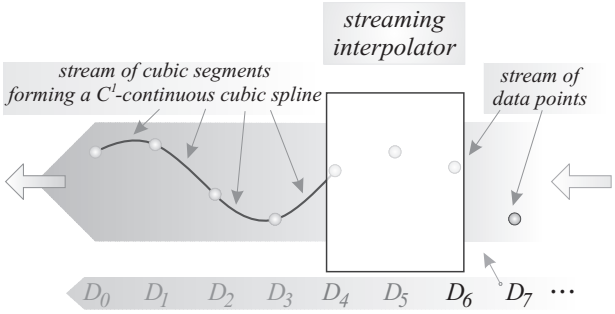

Figure 2. Conceptual diagram of problem statement: streaming interpolator maps *stream of data points* to *stream of cubic spline segments*, which form C^1-continuous interpolant

Remark 2. (Functional programming context). The streaming interpolator can be considered to be a *stream mapper* (from a stream of data points to a stream of spline segments).

4. Proposed solution

The streaming algorithm proposed in this paper is based on the specific use of a cubic Hermite spline segment – a building block of an "on the fly"-constructed spline interpolant. Since the first derivatives of the interpolated function are not given, they must be estimated. Instead of classic approximations with finite difference schemas, two specific heuristics (based on the calculus of variations) are used. Because of the

online characteristic of the interpolation algorithm, they have to be local and are based on a greedy strategy as a result.

The key element of the proposed approach is a specific form of looking ahead from the current point (state), x_{k-1} – not just one level (i.e., x_k, as in the classic greedy approach), but two levels (i.e., x_k and x_{k+1}) to extend the planning horizon by one data point[4]. These two heuristics are briefly described in the following two subsections.

Remark 3. (Notation). To simplify the formulae, it is assumed that the current state is k, which corresponds to x_A (i.e., $x_{k-1} = x_A$). Similarly, for states k and $k+1$, we have $x_k = x_B$ and $x_{k+1} = x_C$, respectively.

4.1. Heuristic *MinBE*: spline of *minimum elastic bending energy*

Given (in a local frame – see Fig. 3):

$$x_A = 0 : \begin{cases} s_k(x_A) = s_{k-1}(x_A) = f_A \\ s'_k(x_A) = s'_{k-1}(x_A) = s'_A \end{cases} \quad x_B : \Big\{ s_k(x_B) = f_B \quad x_C : \Big\{ s_{k+1}(x_C) = f_C \tag{4}$$

and assuming that, at x_B:

$$\begin{cases} s'_k(x_B) = s'_{k+1}(x_B) \\ s''_k(x_B) = s''_{k+1}(x_B), \end{cases} \tag{5}$$

the value of s'_B that minimizes the functional:

$$J[s] = \int_{x_A}^{x_B} \left[s''_k(x)\right]^2 dx + \int_{x_B}^{x_C} \left[s''_{k+1}(x)\right]^2 dx \tag{6}$$

is defined as follows:

$$s'_B = \frac{As_A + Bs'_A + Cs_B + Ds_C}{E}, \tag{7}$$

where:

$$\begin{cases} A = -6(x_C - x_B)^2 \\ B = -2x_B(x_C - x_B)^2 \\ C = 3(2x_C^2 - 4x_B x_C + x_B^2) \\ D = 3x_B^2 \\ E = x_B(x_C - x_B)(4x_C - x_B). \end{cases} \tag{8}$$

Hence, having found the missing value of s'_B, we have:

$$x_A = 0 : \begin{cases} s_k(x_A) = s_A = f_A \\ s'_k(x_A) = s'_A = f'_A \end{cases} \quad x_B : \begin{cases} s_k(x_B) = s_B = f_B \\ s'_k(x_B) = s'_B, \end{cases} \tag{9}$$

[4]As when using the central finite difference approximation.

and the k^{th} segment of the cubic spline interpolant in global coordinate system Oqy (with q corresponding to x) is defined as follows[5]:

$$s_k(q) = \sum_{j=1}^{4} a_j (q - q_{k-1})^{4-j}, \tag{10}$$

where:

$$\begin{cases} a = (\Delta x)^{-3} \left[\Delta x(s'_B + s'_A) - 2(s_B - s_A)\right] \\ b = (\Delta x)^{-2} \left[3(s_B - s_A) - (s'_B + 2s'_A)\Delta x\right] \\ c = s'_A \\ d = s_A \end{cases} \tag{11}$$

assuming that $\Delta x = x_B - x_A$.

4.2. Heuristic *MinAJ2*: spline of *minimum accumulated squared jerk*

Given (in a local frame – see Fig.3):

$$x_A = 0 : \begin{cases} s_k(x_A) = s_{k-1}(x_A) = f_A \\ s'_k(x_A) = s'_{k-1}(x_A) = s'_A \end{cases} \quad x_B : \begin{cases} s_k(x_B) = f_B \end{cases} \quad x_C : \begin{cases} s_{k+1}(x_C) = f_C \end{cases} \tag{12}$$

and assuming that, at x_B:

$$\begin{cases} s'_k(x_B) = s'_{k+1}(x_B) \\ s''_k(x_B) = s''_{k+1}(x_B) \end{cases} \tag{13}$$

the value of s'_B that minimizes the functional:

$$J[s] = \int_{x_A}^{x_B} \left[s'''_k(x)\right]^2 dx + \int_{x_B}^{x_C} \left[s'''_{k+1}(x)\right]^2 dx \tag{14}$$

is defined as:

$$s'_B = \frac{As_A + Bs'_A + Cs_B + Ds_C}{E} \tag{15}$$

where:

$$\begin{cases} A = -(x_C - x_B)^2(2x_C^2 + 2x_B x_C - x_B^2) \\ B = -x_B x_C^2 (x_C - x_B)^2 \\ C = x_C(2x_C^3 - 2x_B x_C^2 - 3x_B^2 x_C + 2x_B^3) \\ D = x_B^3(2x_C - x_B) \\ E = x_B x_C (x_C - x_B)(x_C^2 + x_B x_C - x_B^2) \end{cases} \tag{16}$$

then, the k^{th} segment of the interpolating spline can be found by using Equation 10 and 11.

[5]The monomial basis $\{1, x, x^2, x^3\}$, $x = q - q_{k-1}$, is used – see also Figure 3.

4.3. Algorithm

The conceptual diagram of the algorithm is shown in Figure 3, and its pseudo-code is evident in Algorithm 1.

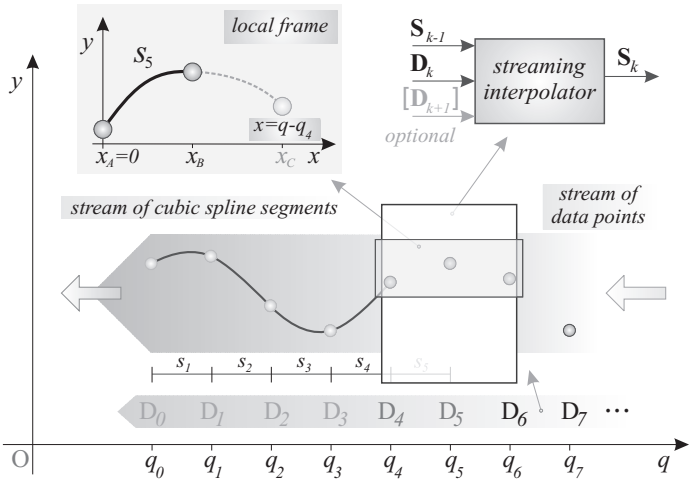

Figure 3. Conceptual diagram of the proposed algorithm: the streaming interpolator maps (in *online way*) incoming data points to cubic spline segments using only data points D_k and D_{k+1} (if available) and the previous spline segment data, s_{k-1}

The data points from the input stream shown in Figure 1 are mapped in an *online way* to cubic spline segments and put into the output stream. At any step k where $k = 1, 2, 3 \ldots$, the streaming interpolator uses only the following:

- currently processed data point D_k as well as D_{k+1} (if available);
- previous spline segment data s_{k-1} (needed to calculate initial conditions for current segment s_k).

For each data point D_k, the interpolator creates a cubic spline segment s_k (see Equation 10), which is at once put into the output stream.

The first and last segments of the output stream are treated in a specific way. At the beginning of the stream, the missing value of s'_0 must be estimated; for instance, in the following way:

$$s'_0 = \frac{(f_1 - f_0)(x_C - x_A)^2 + (f_1 - f_2)(x_B - x_A)^2}{(x_B - x_A)(x_C - x_A)(x_C - x_B)} \tag{17}$$

The coefficients of the last segment (i.e., at the end of the stream) can be computed using Equations 10 and 11 with the missing s'_B equal to:

$$s'_B = \begin{cases} -0.5\ (\Delta x)^{-1}\ (s'_A \Delta x - 3f_B + 3s_A) & \text{for } \mathit{MinBE} \\ -(\Delta x)^{-1}\ (s'_A \Delta x - 2f_B + 2s_A) & \text{for } \mathit{MinAJ} \end{cases} \tag{18}$$

Algorithm 1 Real-time/streaming interpolation using cubic splines

Require:
1) S_{in} {non-empty input stream of data points (at least three data points required)},
2) S_{out} {output stream of cubic spline segments},
3) I_H {interpolation heuristic (strategy)}
{1. Initialization phase}
S_{in}.open()
S_{out}.open()
$dp_0 \leftarrow S_{in}$.nextElem() {dp - data point}
$dp_1 \leftarrow S_{in}$.nextElem()
$dp_2 \leftarrow S_{in}$.nextElem()
$s_{ic} \leftarrow$ initCondsFrom(dp_0, dp_1, dp_2) {$s_{ic} = (s_0, s_0')$ – see Eq.17}
$s \leftarrow$ cubicSegmFor(s_{ic}, dp_1, dp_2, I_H) {the first segment}
S_{out}.put(s)
$dp_{k+1} \leftarrow dp_2$
$s_{ic} \leftarrow$ initCondsFrom(s)
{2. Streaming processing}
while S_{in}.hasNext() **do**
  $dp_k \leftarrow dp_{k+1}$
  $dp_{k+1} \leftarrow S_{in}$.nextElem()
  $s \leftarrow$ cubicSegmFor(s_{ic}, dp_k, dp_{k+1}, I_H) {the k-th segment}
  S_{out}.put(s)
  $s_{ic} \leftarrow$ initCondsFrom(s)
end while
{3. Finalization phase}
$s \leftarrow$ cubicSegmFor(s_{ic}, dp_{k+1}, I_H) {the last (special) segment – see Eq.18}
S_{out}.put(s)
S_{in}.close()
S_{out}.close()

Remark 4. (Algorithm extensibility). Since Algorithm 1 is parametrized by a interpolation heuristic (strategy), it specifies a one-parameter family of streaming interpolation algorithms.

Remark 5. (Algorithm complexity). The streaming (online) character of the presented algorithm means that it requires $O(n)$ time and $O(1)$ space, where n stands for the length of S_{in} (potentially $n \to \infty$).

5. Results and discussion

To evaluate the proposed heuristics, a series of numerical experiments was carried out in the form of a comparative analysis. The interpolants were examined, along with their first three derivatives. As a point of reference, a *cubic Hermite spline*[6] was used.

[6]The classic solution to the problem of (local) interpolation in which the first derivatives of the interpolated function are given (or can be estimated by *usually using finite difference schemas*).

A brief summary of the evaluation process used is given in Subsection 5.1, and the results of the experiments are presented in Subsections 5.2–5.4.

5.1. Evaluation process overview

The key aspects of the evaluation process – *test functions* and *algorithm performance descriptors* – are briefly described in this section, along with some remarks about the auxiliary notation used.

Test functions. The following four test functions:

$$\begin{cases} f_1(x) = e^{-x^2} \sin x \ , \ -3 \le x \le 3 & \text{(19a)} \\ f_2(x) = \dfrac{\log x}{\sqrt{x}} \sin x \ , \ 1 \le x \le 5 & \text{(19b)} \\ f_3(x) = \dfrac{1}{1+e^{-x}} \ , \ -2 \le x \le 2 & \text{(19c)} \\ f_4(x) = \dfrac{36x^7 - 229x^5 + 25x^3}{36} \ , \ -1 \le x \le 1 & \text{(19d)} \end{cases}$$

were used to generate the streams of data points to be interpolated. The graphs of these functions are shown in Figure 4.

Performance descriptors. Each of the compared interpolants was evaluated by using the following measures:

- Mean Absolute Error:

$$MAE(g_1, g_2) = \frac{1}{n} \sum_{i=1}^{n} |g_1(x_i) - g_2(x_i)| \tag{20}$$

- Root Mean Squared Error:

$$RMSE(g_1, g_2) = \left\{ \frac{1}{n} \sum_{i=1}^{n} [g_1(x_i) - g_2(x_i)]^2 \right\}^{1/2} \tag{21}$$

- Normalized Root Squared Error:

$$NRSE(g_1, g_2) = \left\{ \frac{\sum_{i=1}^{n} [g_1(x_i) - g_2(x_i)]^2}{\sum_{i=1}^{n} [g_2(x_i)]^2} \right\}^{1/2} \tag{22}$$

- Mean Absolute Error Quotient:

$$Q_{\mathrm{MAE}}(g_1, g_2)|_f = \frac{MAE(g_1, f)}{MAE(g_2, f)} \tag{23}$$

- Root Mean Squared Error Quotient:

$$Q_{\mathrm{RMSE}}(g_1, g_2)|_f = \frac{RMSE(g_1, f)}{RMSE(g_2, f)} \tag{24}$$

- Absolute Error (AE):

$$e_{\mathrm{a}}(x)|_{g_2}^{g_1} = |g_1(x) - g_2(x)| \tag{25}$$

and its maximum value (L_∞-norm):

$$||e_{\mathrm{a}}(x)|_{g_2}^{g_1}||_\infty \tag{26}$$

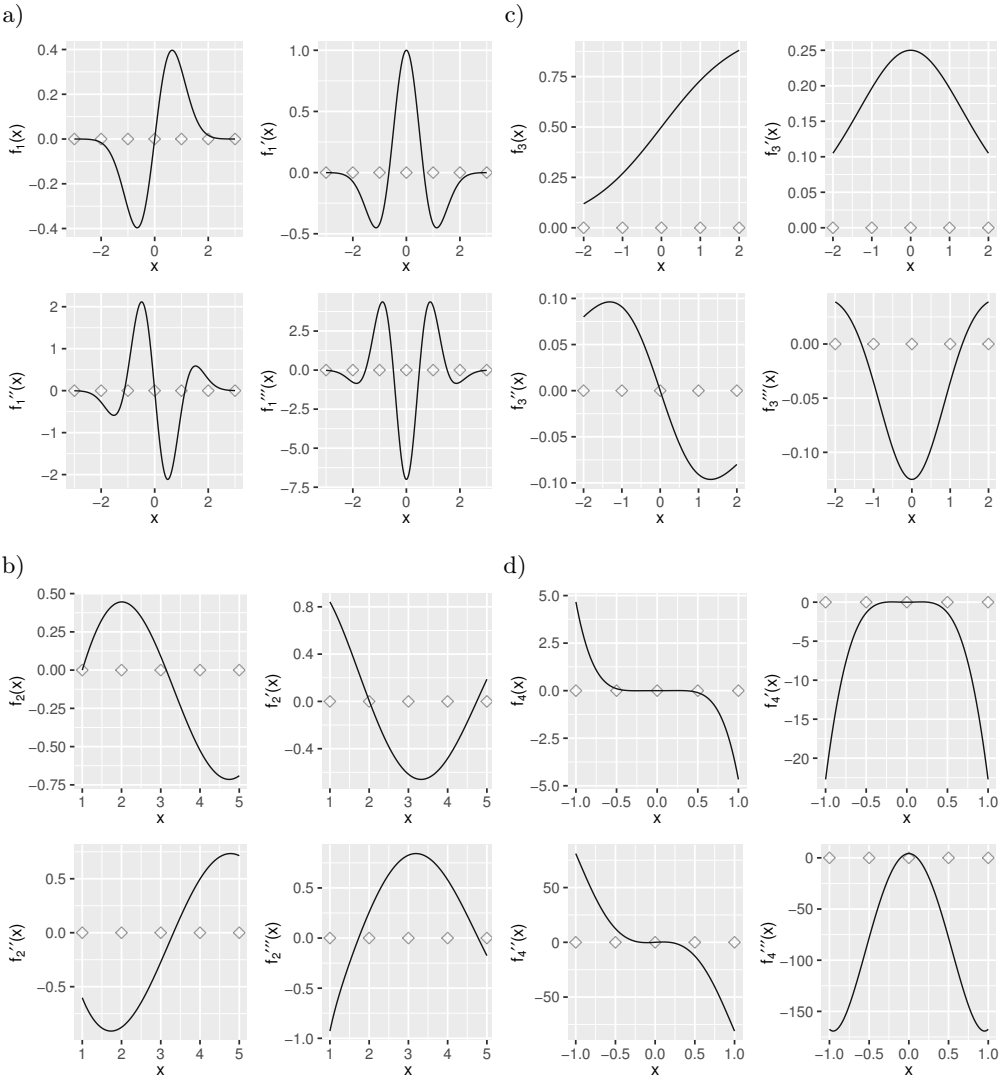

Figure 4. Test functions and their first three derivatives (together with interpolation nodes used in evaluation): a) $f_1(x)$; b) $f_3(x)$; c) $f_2(x)$; d) $f_4(x)$

Auxiliary notation. The auxiliary definitions and naming conventions used in this section are as follows:

- $s_E(x)$ or s_E – a cubic spline generated by heuristic *MinBE*;
- $s_J(x)$ or s_J – a cubic spline generated by heuristic *MinAJ2*;
- cubic spline of the T type – a spline generated by heuristic $T \in \{$*MinBE*, *MinAJ2*$\}$;
- $h(x)$ or h – a cubic Hermite spline;
- $\{D_k\}_f,\ k = 0, 1, 2, \ldots$ – a data stream generated by function f;
- $f'(x)$, $f''(x)$, and $f'''(x)$ – the first, second, and third derivatives of f;
- $\mathbf{e}_a[\ g_1, g_2, \ldots\ ]_f(x) = [\ e_a(x)|_f^{g_1}, e_a(x)|_f^{g_2}, \ldots\ ]$;
- $\mathbf{e}_a[\ g_1, g_2, \ldots\ ]_f = [\ e_a|_f^{g_1}, e_a|_f^{g_2}, \ldots\ ]$ – a special (point-free) case of the above used for axis titles;
- $\mathbf{e}_{nrs}[g_1, g_2, \ldots]_f = [NRSE(g_1, f), NRSE(g_1, f), \ldots]$.

Example 1. $s_J'''(x)$ stands for the third derivative of $s_J(x)$, where $s_J(x)$ is a cubic spline generated by the *MinAJ2* heuristic.

Example 2. $\mathbf{e}_a[s_J, h]_{f_1}(x) = [\ e_a(x)|_{f_1}^{s_J}, e_a(x)|_{f_1}^{h}\ ]$ stands for a pair of absolute error functions ($e_a(x)|_{f_1}^{s_J}$ and $e_a(x)|_{f_1}^{h}$), where $f_1(x)$ is the reference function, $s_J(x)$ is a cubic spline generated by the *MinAJ2* heuristic, and $h(x)$ is a cubic Hermite spline.

5.2. Streaming interpolator using heuristic *MinBE*

Remember that each element of the incoming stream takes the form of $D_k = (x_k, f_k))$, $k = 0, 1, 2\ldots$ The missing initial value of the first derivative (s_0') was calculated by using Equation 17. Following this, the coefficients defining the current interpolating segment were computed at each interpolation step k by using Equations 7, 8, 10, and 11. The end of the stream was handled according to Equations 10, 11, and 18.

The performance of the *MinBE* interpolator, expressed in terms of the descriptors listed in Subsection 5.1, is summarized in Figure 5 and Table 1. A *cubic Hermite spline* is treated as a point of reference.

Local performance measures. Figure 5 presents the absolute interpolation error distributions for the cubic splines of the *MinBE* type and the *cubic Hermite splines*, which both interpolate the data streams generated by the f_1–f_4 test functions. The interpolants were examined, along with their first three derivatives, and the cubic spline of the *MinBE* type outperformed the Hermite spline (significantly so in some cases).

Global/integral performance measures. Table 1 gives the values of the *Mean Absolute Error Quotient* (Q_{MAE}) and the *Root Mean Squared Error Quotient* (Q_{RMSE}) for the cubic spline of the *MinBE* type as compared to the *cubic Hermite spline*. The two-column blocks ($(\cdot)$, $(\cdot)''$, ...) correspond to the interpolants and their first three derivatives.

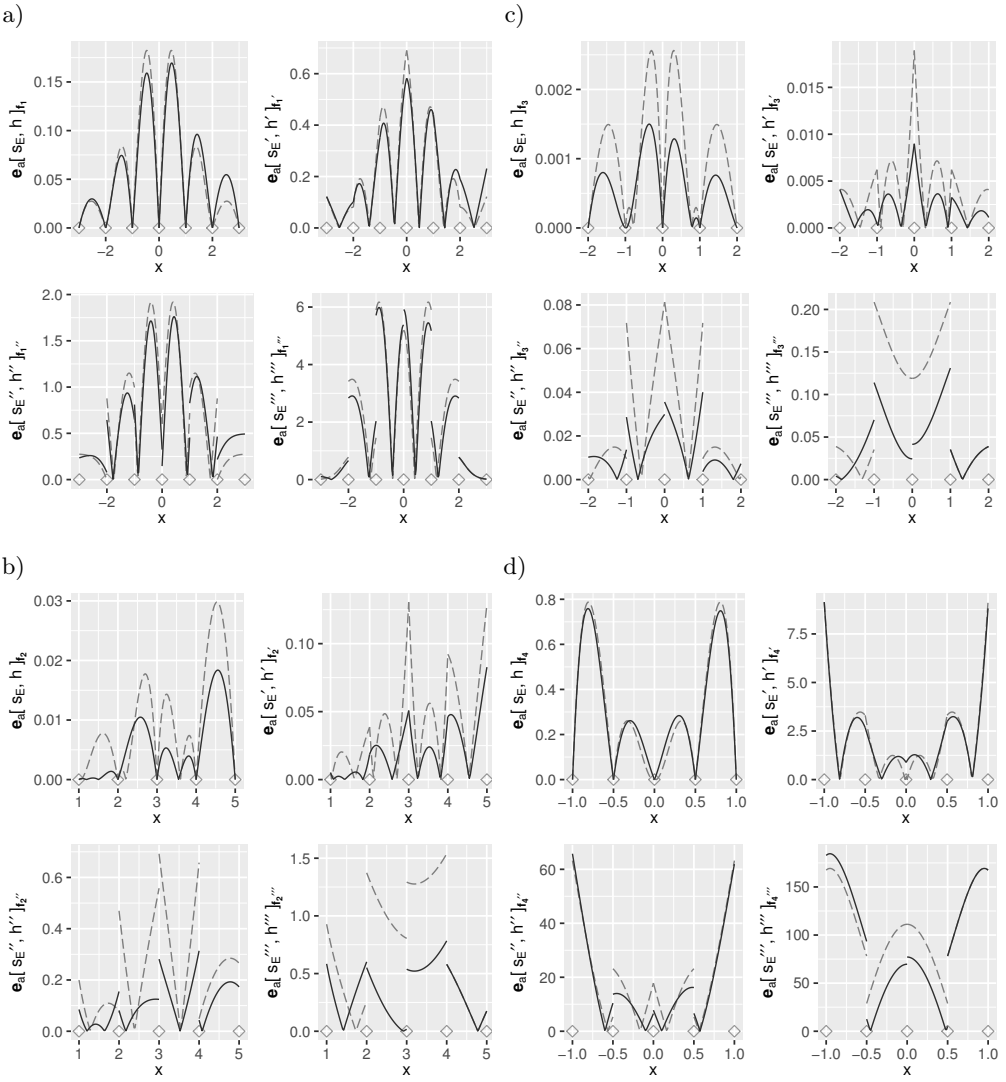

Figure 5. Cubic spline of *MinBE* type (solid line) vs. *cubic Hermite spline* (dashed line) in interpolation of test streams: absolute errors for interpolants and their first three derivatives. Interpolation nodes are as shown in Figures 4a–4d: a) stream $\{D_k\}_{f_1}$ (generated by f_1); b) stream $\{D_k\}_{f_3}$ (generated by f_3); c) stream $\{D_k\}_{f_2}$ (generated by f_2); d) stream $\{D_k\}_{f_4}$ (generated by f_4)

Table 1

Cubic splines of *MinBE* type vs. *cubic Hermite spline* in the interpolation of $\{D_k\}_{f_i}$: quotients of errors for the interpolants and their first three derivatives. Q_{MAE} and Q_{RMSE} computed for $g_1 = s_E$, $g_2 = h$. The interpolation nodes as shown in Figures 4a–4d

f	$(\cdot)(x)$		$(\cdot)'(x)$		$(\cdot)''(x)$		$(\cdot)'''(x)$	
	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}
f_1	1.00	0.944	0.995	0.938	0.946	0.905	0.932	0.943
f_2	0.545	0.590	0.508	0.536	0.441	0.447	0.454	0.448
f_3	0.546	0.552	0.518	0.512	0.498	0.468	0.515	0.500
f_4	0.998	0.965	0.978	0.965	0.904	0.958	0.870	0.954

Aside from the first table entry, all of the values are less than one, which means that the cubic spline of the *MinBE* type outperformed the Hermite cubic spline.

Remark 6. (Cubic spline of *MinBE* type's performance). In almost all cases, the streaming interpolator based on the cubic spline of the *MinBE* type outperformed (sometimes significantly) its classic alternative – the *cubic Hermite spline*.

5.3. Streaming interpolator using heuristic *MinAJ2*

As in the previous case, the missing initial value of the first derivative (s'_0) was calculated by using Equation 17. Following this, the coefficients that define the current interpolating segment were computed at each interpolation step k by using Equations 15, 16, 10, and 11. The end of the stream was handled according to Equations 10, 11, and 18.

The performance of the interpolator that was based on heuristic *MinAJ2* (expressed in terms of the descriptors listed in Subsection 5.1) is summarized in Figure 6 and Table 2. As before, a *cubic Hermite spline* is treated as a point of reference.

Local performance measures. Figure 6 presents the absolute interpolation error distributions for the cubic splines of the *MinAJ2* type and the *cubic Hermite splines* interpolating the data streams that were generated by test functions f_1–f_4. The interpolants were examined, along with their first three derivatives, and the cubic spline of the *MinAJ2* type outperformed the Hermite spline (significantly so in some cases).

Global/integral performance measures. The values of Q_{MAE} and Q_{RMSE} for the cubic spline of the *MinAJ2* type as compared to the *cubic Hermite spline* are given in Table 2. The two-column blocks ($(\cdot)$, $(\cdot)''$, ...) correspond to the interpolants and their first three derivatives.

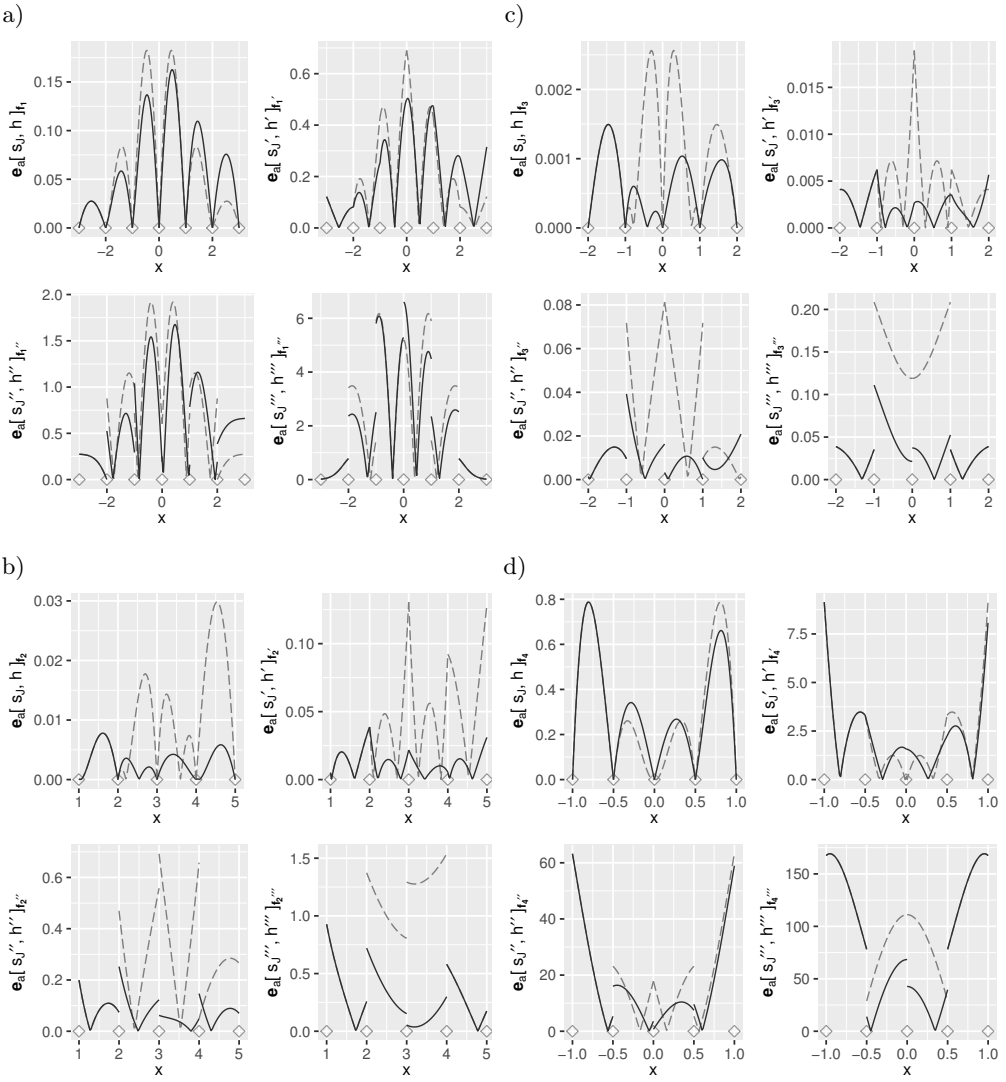

Figure 6. Cubic spline of *MinAJ2* type (solid line) vs. *cubic Hermite spline* (dashed line) in interpolation of test streams: absolute errors for interpolants and their first three derivatives. Interpolation nodes are as shown in Figures 4a–4d: a) stream $\{D_k\}_{f_1}$ (generated by f_1); b) stream $\{D_k\}_{f_3}$ (generated by f_3); c) stream $\{D_k\}_{f_2}$ (generated by f_2); d) stream $\{D_k\}_{f_4}$ (generated by f_4)

Table 2

Cubic splines of *MinAJ2* type vs. *cubic Hermite spline* in interpolation of $\{D_k\}_{f_i}$: quotients of errors for interpolants and their first three derivatives. Q_{MAE} and Q_{RMSE} are computed for $g_1 = s_J$, $g_2 = h$. Interpolation nodes are as shown in Figures 4a–4d

f	$(\cdot)(x)$		$(\cdot)'(x)$		$(\cdot)''(x)$		$(\cdot)'''(x)$	
	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}	Q_{MAE}	Q_{RMSE}
f_1	0.985	0.909	0.974	0.901	0.918	0.864	0.905	0.924
f_2	0.301	0.283	0.301	0.286	0.308	0.303	0.367	0.387
f_3	0.573	0.564	0.505	0.475	0.397	0.358	0.355	0.346
f_4	1.010	0.955	0.979	0.946	0.878	0.921	0.780	0.887

Aside from one table entry, all of values are less than one, which means that the cubic spline of the *MinAJ2* type outperformed the Hermite cubic spline.

Remark 7. (Cubic spline of *MinAJ2* (type performance) In almost all cases, the streaming interpolator based on the cubic spline of the *MinAJ2* type outperformed (sometimes significantly) its classic alternative – the *cubic Hermite spline*.

5.4. Streaming spline interpolation: experimental convergence analysis

The evaluation results presented so far give quite a significant insight into the performance of different streaming interpolators (both the local and global/integral performance measures were considered); however, there is still one key performance aspect that has not been addressed at all – the convergence rate. The *convergence rate* is a measure of how fast the interpolation/approximation error drops to zero as the number of interpolation nodes increases (or the distance between subsequent nodes decreases[7]). This measure is often used in numerical analysis to determine the relative accuracy of different interpolation methods.

For the cubic Hermite spline, it is known (see, for instance, [7]) that, for $x \in [x_A, x_B]$:

$$|f(x) - h(x)| \leq \frac{1}{384}(\Delta x)^4 \max_{x_A \leq z \leq x_B} |f^{(4)}(z)| \tag{27}$$

where f is the interpolated function, h is the cubic Hermite spline interpolant, and Δx stands for the distance between the interpolation nodes

In the following comparative analysis, the cubic Hermite spline is again treated as a point of reference. The results of the experimental analysis of convergence are presented as log-log graphs.

[7]This is the point of view used in FEM analysis.

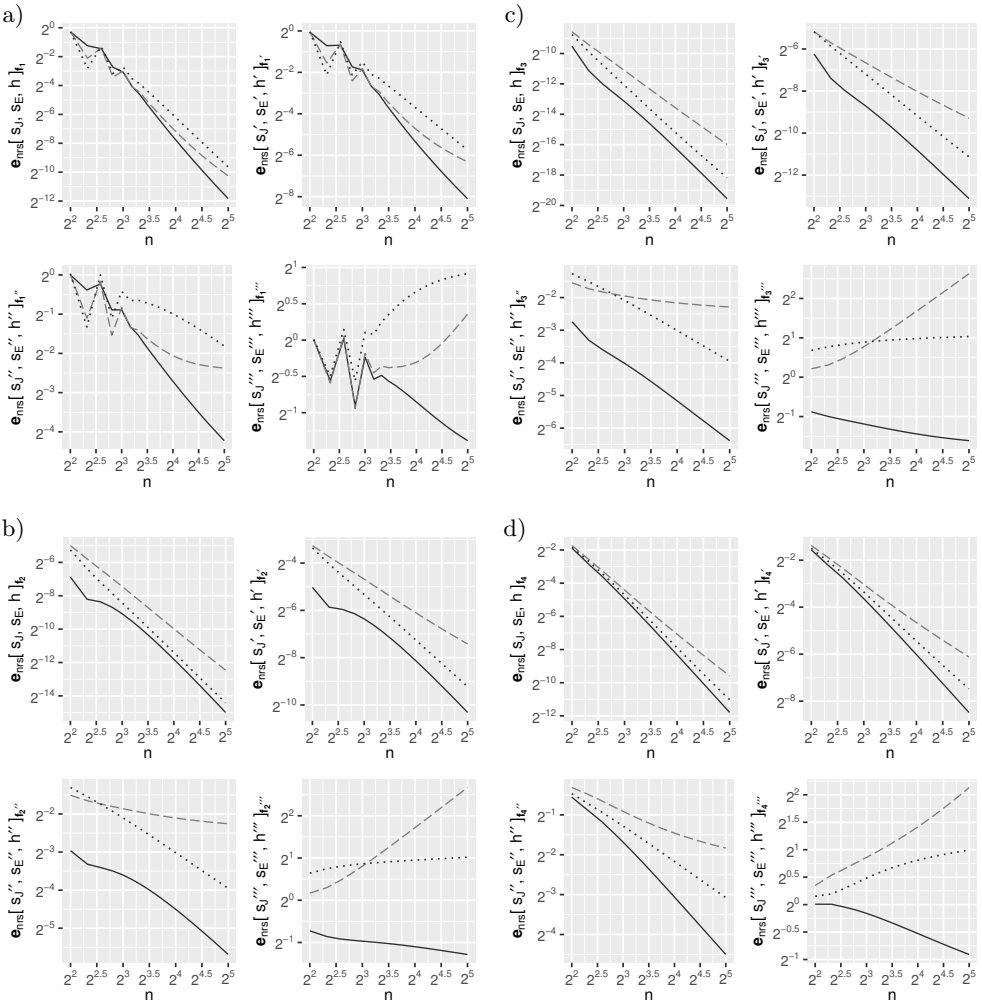

Figure 7. Cubic spline of *MinAJ2* type (solid line) vs. cubic spline of *MinBE* type (dashed line) vs. *cubic Hermite spline* (dotted line) in interpolation of test streams: normalized root squared error as function of number of interpolation intervals n for interpolants and their first three derivatives; a) stream $\{D_k\}_{f_1}$ (generated by f_1); b) stream $\{D_k\}_{f_3}$ (generated by f_3); c) stream $\{D_k\}_{f_2}$ (generated by f_2); d) stream $\{D_k\}_{f_4}$ (generated by f_4)

There are two key observations that are related to Figure 7:

- the splines of the *MinAJ2* type significantly outperformed both the cubic Hermite splines and the splines of the *MinBE* type;
- the convergence rates for the cubic Hermite splines and the splines of the *MinBE* type can be *negative* (in the case of the third derivative)!

Remark 8. (Convergence analysis summary). The cubic spline of the *MinAJ2* type was the best (and the only stable one) among the tested interpolants.

6. Conclusions

It has been shown that the real-time C^1-continuous cubic spline interpolation of streaming data can be effective. The key element in achieving this was to use an estimator of the first derivative of the interpolated function that is more accurate than those that are based on finite difference schemas.

Two greedy look-ahead heuristic estimators (denoted as *MinBE* and *MinAJ2*) have formally been defined (in closed form) along with the corresponding cubic splines that they generate. They were then comparatively evaluated in a series of numerical experiments that involved different types of performance measures. *The cubic Hermite splines that were generated by heuristic MinAJ2 significantly outperformed those that were based on finite difference schemas in terms of all of the tested performance measures (including convergence)* – see Figures 6 and 7.

The proposed approach is quite general. It can be directly applied to streams of univariate functional data like time-series. After splitting, multi-dimensional curves that are defined parametrically can be handled as well. The streaming character of the algorithm means that it can also be useful in processing data sets that are too large to fit in the memory (e.g., edge computing devices, embedded time-series databases).

Future research could concentrate on new interpolation heuristics, their new applications (e.g., in data compression, real-time animations), and auto-adaptation mechanisms for streaming interpolators.

References

[1] Angeles J.: *Fundamentals of Robotic Mechanical Systems*, Springer, 2002.

[2] Bazaz S.A., Tondu B.: Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators, *Robotics and Autonomous Systems*, vol. 29(4), pp. 257–268, 1999.

[3] Bézier P.: Définition numerique de courbes et surfaces I, Automátisme, vol. XI, pp. 625–632, 1966.

[4] Biagiotti L., Melchiorri C.: *Trajectory Planning for Automatic Machines and Robots*, Springer Science & Business Media, 2008.

[5] Birkhoff G.D.: General mean value and remainder theorems with applications to mechanical differentiation and quadrature, *Transactions of the American Mathematical Society*, vol. 7(1), pp. 107–136, 1906.

[6] Bolton K.: Biarc curves, *Computer-Aided Design*, vol. 7(2), pp. 89–92, 1975.

[7] Boor de C.: *A Practical Guide to Splines*, Applied Mathematical Sciences, vol. 27, Springer-Verlag, New York, 1978.

[8] Carvalho de J.M., Hanson J.V.: Real-time interpolation with cubic splines and polyphase networks, *Canadian Electrical Engineering Journal*, vol. 11(2), pp. 64–72, 1986.

[9] Catmull E., Rom R.: A class of local interpolating splines. In: *Computer aided geometric design*, pp. 317–326. Elsevier, 1974.

[10] De Casteljau P.: Courbes et surfaces à pôles. *André Citroën,* Automobiles SA, Paris, 1963.

[11] Dębski R.: High-performance simulation-based algorithms for an alpine ski racer's trajectory optimization in heterogeneous computer systems, *International Journal of Applied Mathematics and Computer Science*, vol. 24(3), pp. 551–566, 2014.

[12] Dębski R.: An adaptive multi-spline refinement algorithm in simulation based sailboat trajectory optimization using onboard multi-core computer systems, *International Journal of Applied Mathematics and Computer Science*, vol. 26(2), pp. 351–365, 2016.

[13] Dębski R.: Simulation-Based Sailboat Trajectory Optimization Using On-Board Heterogeneous Computers, *Computer Science*, vol. 17(4), pp. 461–481, 2016.

[14] Fan W., Lee C.H., Chen J.H.: A realtime curvature-smooth interpolation scheme and motion planning for CNC machining of short line segments, *International Journal of Machine Tools and Manufacture*, vol. 96, pp. 27–46, 2015.

[15] Farin G., Hoschek J., Kim M.S.: *Handbook of computer aided geometric design*, Elsevier, 2002.

[16] Forrest A.R.: *Curves and surfaces for computer-aided design.* Ph.D. thesis, University of Cambridge, 1968.

[17] Forrest A.R.: Interactive interpolation and approximation by Bézier polynomials, *Computer-Aided Design*, vol. 22(9), pp. 527–537, 1990.

[18] Guven O., Eftekhar A., Kindt W., Constandinou T.G.: Computationally efficient real-time interpolation algorithm for non-uniform sampled biosignals, *Healthcare Technology Letters*, vol. 3(2), pp. 105–110, 2016.

[19] Hermite M.C., Borchardt M.: Sur la formule d'interpolation de Lagrange, *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1878(84), pp. 70–79, 1878.

[20] Jüttler B.: Hermite interpolation by Pythagorean hodograph curves of degree seven, *Mathematics of Computation*, vol. 70(235), pp. 1089–1111, 2000.

[21] Knott G.D.: *Interpolating Cubic Splines*, Progress in Computer Science and Applied Logic, vol. 18, Birkhäuser, Boston, MA, 2000.

[22] Kröger T.: *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*, Springer Tracts in Advanced Robotics, vol. 58, Springer-Verlag, Berlin–Heidelberg, 2010.

[23] Li J.: A class of quintic Hermite interpolation curve and the free parameters selection, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 13(1), pp. JAMDSM0011–JAMDSM0011, 2019.

[24] Lu L.: Planar quintic G^2 Hermite interpolation with minimum strain energy, *Journal of Computational and Applied Mathematics*, vol. 274, pp. 109–117, 2015.

[25] Lu L., Jiang C., Hu Q.: Planar cubic G^1 and quintic G^2 Hermite interpolations via curvature variation minimization, *Computer & Graphics*, vol. 70, pp. 92–98, 2018.

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

[27] Milenkovic V., Milenkovic P.H.: Tongue Model for Characterizing Vocal Tract Kinematics. In: *Recent Advances in Robot Kinematics*, pp. 217–224, Springer,1996.

[28] Monceau Du H.L.D.: *Elémens de l'architecture navale ou traité pratique de la construction des vaisseaux*, CA Jombert, 1752.

[29] Ogniewski J.: Cubic Spline Interpolation in Real-Time Applications using Three Control Points. In: *27. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG 2019, Plzen, Czech Republic, May 27–30, 2019*, vol. 2901, pp. 1–10, World Society for Computer Graphics, 2019.

[30] Schoenberg I.: Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions, *Quarterly of Applied Mathematics*, vol. 4(1), pp. 45–99, 1946.

[31] Schoenberg I.J.: *Cardinal spline interpolation*, vol. 12. Siam, 1973.

[32] Schumaker L.: *Spline functions: basic theory*, Cambridge University Press, 2007.

[33] Späth H.: *One Dimensional Spline Interpolation Algorithms*. Ak Peters Series, Taylor & Francis, 1995.

[34] Versprille K.J.: *Computer-Aided Design Applications of the Rational B-Spline Approximation Form*. Ph.D. thesis, Syracuse University, 1975.

[35] Wallis J.: Arithmetica infinitorum, *John Wallis, Operum Mathematicorum*, vol. 2, pp. 1–199, 1656.

Affiliations

Roman Dębski
AGH University of Science and Technology, Department of Computer Science,
al. Mickiewicza 30, 30-059 Krakow, Poland, rdebski@agh.edu.pl,
ORCID ID: https://orcid.org/0000-0003-3283-6032

Received: 28.07.2020
Revised: 19.10.2020
Accepted: 19.10.2020