Author(s): Averbuch, Amir; Neittaanmäki, Pekka; Shefi, Etay; Zheludev, Valery

Title: Local cubic splines on non-uniform grids and real-time computation of wavelet transform

Year: 2017

Version:

Please cite the original version:

Averbuch, A., Neittaanmäki, P., Shefi, E., & Zheludev, V. (2017). Local cubic splines on non-uniform grids and real-time computation of wavelet transform. Advances in Computational Mathematics, 43(4), 733-758. https://doi.org/10.1007/s10444-016-9504-x

Adv Comput Math
DOI 10.1007/s10444-016-9504-x

Local cubic splines on non-uniform grids and real-time computation of wavelet transform

Amir Averbuch[1] · Pekka Neittaanmäki[3] · Etay Shefi[2] · Valery Zheludev[1,3]

Received: 13 December 2015 / Accepted: 23 November 2016

Abstract In this paper, local cubic quasi-interpolating splines on non-uniform grids are described. The splines are designed by fast computational algorithms that utilize the relation between splines and cubic interpolation polynomials. These splines provide an efficient tool for real-time signal processing. As an input, the splines use either clean or noised arbitrarily-spaced samples. Formulas for the spline's extrapolation beyond the sampling interval are established. Sharp estimations of the approximation errors are presented. The capability to adapt the grid to the structure of an object and to have minimal requirements to the operating memory are of great advantages for offline processing of signals and multidimensional data arrays. The designed splines serve as a source for generating real-time wavelet transforms to apply to signals in scenarios where the signal's samples subsequently arrive one after the other at random times. The wavelet transforms are executed by six-tap weighted moving averages of the signal's samples without delay. On arrival of new samples, only a couple of adjacent transform coefficients are updated in a way that no boundary effects arise.

Keywords Local splines · Quasi-interpolating splines · Real-time wavelet transform · Discrete vanishing moments

Communicated by: Arieh Iserles

✉ Valery Zheludev
zhel@post.tau.ac.il

[1] School of Computer Science, Tel Aviv University, Tel Aviv, 69978, Israel

[2] School of Electrical Engineering, Tel Aviv University, Tel Aviv, 69978, Israel

[3] Department of Mathematical Information Technology P.O. Box 35 (Agora), University of Jyväskylä, Jyväskylä, Finland

Published online: 06 December 2016

Mathematics Subject Classification (2010) 41A15 · 42C40

1 Introduction

Since their introduction in [14], splines have become one of the most powerful tools in mathematics and computer aided geometric design. In recent decades, splines have served as a source for the constructions of wavelets, wavelet packets and wavelet frames. Splines and spline- based wavelets, wavelet packets and frames are extensively used in signal and image processing applications (see, for example, [3, 4]).

Interpolating splines possess exclusive approximation properties. In particular, an interpolating spline of order p, which consists of pieces of polynomials of degree $p-1$, restores polynomials of the same degree. Due to this property, these splines generate biorthogonal wavelets with p and, in some cases, $p+1$ vanishing moments ([3, 4]). However a drawback in the design and manipulation of interpolating splines is that, for their computation, a system of equations that involves all the available grid samples has to be solved. This fact prevents the usage of these splines for real-time processing.[1]

Therefore, the idea to have splines that can be designed and manipulated directly without resorting to systems of equations, while their approximation accuracy is close to that of the interpolating splines, is attractive. A method for the design of such splines on a uniform grid was presented in the pioneering spline paper [14]. When the grid is non-uniform, the design and estimation of the approximation properties of such splines is more complicated, especially for higher-order splines. A number of investigations in this field were carried out in the 70's (see, for example, [13]). These splines are called local because the computation of a spline's value at a fixed point requires to use only a few adjacent grid samples. Nevertheless, there exist local splines which provide the same approximation order as the interpolating splines. That is, there exist local splines of any order p which restore polynomials of degree $p-1$. Such local splines are referred to as quasi-interpolating splines.

In this paper, we describe a procedure to design and analyze local cubic quasi-interpolating splines on arbitrary grids. Typically, local splines are designed via their B-spline representation. An alternative approach presented in this paper, which is based on the relation between quasi-interpolating splines and cubic interpolating polynomials, results in a "simple" algorithm for spline computation. When samples are given on a limited interval, which is a typical situation, the spline is extended to the boundaries of the interval without loss of its approximation accuracy. In addition, a method for accurate extrapolation of the spline beyond the sampling interval is presented. Only six adjacent samples are needed to compute the spline's value at a certain point. Therefore, the design of the spline can be implemented in a real-time mode when samples of a signal arrive dynamically and sequentially at random times. Due to the extension formulas, the design can be carried out without delay up to

[1] Under the real-time processing we mean a situation when signal's samples arrive dynamically and sequentially and the computations are executed with a minimal, if any, delay with respect to the samples arrival.

the latest sample arrival. In addition, the extrapolation algorithm can be applied to a prediction-correction processing of signals evolving in time.

Similarly to cubic interpolating splines, the described splines restore cubic polynomials at inner parts of the sampling interval and so also near the boundaries and in the extrapolation process. Moreover, the spline representation via interpolating polynomials provides sharp estimations for the approximation errors.

The described splines are used for the construction of wavelet transforms of arbitrarily sampled signals. Most of the existing wavelet transforms operate on uniformly sampled signals. A few works that describe wavelets on non-uniform grids appeared in recent years. Equidistant wavelets, which are utilized for denoising of non-uniformly sampled signals, appear in [6]. Scaling functions on different levels of enclosed irregular grids, which appear in [7], are designed as limit functions of subdivision and wavelets are designed as their linear combinations. In [11], a local decomposition of the space of splines, which are constructed on a fine irregular grid, onto a coarse-grid spline space and its orthogonal complement was presented. The bases of these subspaces are formed by quasi-interpolating splines and compactly supported pre-wavelets. A general method for constructing wavelets on irregular lattices in $\mathbb{R}^d$ and a general atomic frame decomposition of the space $L^2(\mathbb{R}^d)$ are described in [2].

In the current paper, the quasi-interpolating splines are utilized as a source of finite impulse response "filters" to generate wavelet transforms of discrete-time arbitrarily sampled signals. When sampling is uniform, these are filters in a proper sense of the word. Neither scaling functions nor continuous wavelets are involved.

A natural way to design and implement wavelet transforms of signals is the Lifting Scheme introduced in [18], which consists of subsequent applications of *prediction* and *updating* operators to signal's samples. The scheme was extended in [19] to non-uniformly sampled signals. The lifting wavelet transform on a non-uniform grid, where wavelets have two vanishing moments, is applied in [20] to signal denoising. Our design scheme is based on quasi-interpolating local splines. The idea which, in the case of uniform sampling, is explored in [4], is to predict odd samples of the signal to be transformed by values of a spline constructed on the even samples of the signal. Then, the detail coefficients are derived by subtraction of the predicted odd samples from the original ones. The next step consists of updating the even samples by values of the spline designed on the detail coefficients. Since quasi-interpolating splines restore cubic polynomials, the detail wavelet transform coefficients are zero if the signal is a sampled cubic polynomial (at least locally). In that sense, we claim that the arising discrete-time wavelet transforms have four discrete vanishing moments. Typically, wavelet transforms of signals (images) defined on limited intervals (areas) require an extension of the object beyond its boundaries ([5]), in order to reduce boundary effects. However, by using the definition of the splines near the boundaries of the sampling interval, the wavelet transforms are implemented without this type of extension. The design of splines make them useful for real-time (with no delay) execution of wavelet transforms in the situation when samples of a signal subsequently arrive at random time moments.

A real-time denoising method, which is based on the lifting wavelet transform of uniformly sampled signals, is presented in [12]. The online wavelet transform is implemented using a moving window (see also [21]), which produces some delay

with respect to the sample acquisition and requires an artificial extension of the data beyond the moving window. Our algorithm produces wavelet coefficients with no delay. No extension of the data is needed. Arrival of a new sample leads to the production of a new coefficient and to updating of a couple of already produced coefficients.

The paper is organized as follows. Section 2 introduces the necessary notation and recalls definitions of the divided differences and interpolating polynomials. Section 3 introduces local cubic quasi-interpolating splines, describes algorithms for spline computation on unlimited and finite intervals and an extrapolation method. Approximation properties of the splines are investigated and numerical examples are provided. The Lifting Scheme of the wavelet transforms is presented in Section 4.1. In Section 4.2, spline- based prediction and updating operators are explicitly described. A scheme for the real-time wavelet transforms computation is outlined in Section 4.3 and numerical examples are provided.

In order for the paper to be self-contained, some results about local quasi-interpolating splines, which appeared in [17, 23], are included.

2 Preliminaries

The following notations are used throughout the paper:

The grid on the real axis is denoted by $\mathbf{t} := \{t[k]\}$. The steps of the grid are denoted by $h[k] := t[k+1] - t[k]$.
If $t \in [t[k], t[k+1]]$ then a local variable is $\tau := (t - t[k])/h[k] \in [0, 1]$.
The n-order divided difference (DD) ([1, 9, 16]) of a sequence $\mathbf{f} = \{f[k]\}$ with respect to the grid $\mathbf{t}$ is

$$f\,[k, k+1, k+2, ..., k+n] := \sum_{l=k}^{n} \frac{f[k+l]}{\omega_n'[k](t[k+l])}, \tag{1}$$

where

$$\omega_m[k](t) := (t - t[k])\,(t - t[k+1])\,...\,(t - t[k+m])\,. \tag{2}$$

If a function $f(t)$ belongs to $C^n\,[t[k], t[k+n]]$, then the DD of the sequence $\mathbf{f} := \{f[k+\nu] = f(t[k+\nu])\}\,,\ \nu = 0, ..., n$, is

$$f\,[k, k+1, k+2, ..., k+n] = \frac{f^{(n)}(\theta)}{n!}, \quad \theta \in (t[k], t[k+n])\,. \tag{3}$$

The notation $P(t)[k]$ is used for a cubic polynomial that interpolates a function $f(t)$ at the grid points $\{t[k-1], t[k], t[k+1], t[k+2]\}$. The remainder term of the interpolation is explicitly expressed by

$$R(t) := f(t) - (t) = f[t, k, k+1, ..., k+3]\,\omega_3[k](t). \tag{4}$$

3 Local cubic splines

The cubic B-spline is

$$b(t)[k] = (t[k+4] - t[k]) \sum_{\nu=0}^{4} \frac{(t - t[k+\nu])_+^3}{w_4'[k](t[k+\nu])}. \tag{5}$$

It is supported on the interval $(t[k], t[k+4])$. The following observation will be used for further design.

Proposition 3.1

1. *The values of the B-spline* $b(t)[k-3]$ *at the interval* $[t[k], t[k+1]]$ *do not depend on the location of the grid point* $t[k-3]$.
2. *The values of the B-spline* $b(t)[k]$ *at the interval* $[t[k], t[k+1]]$ *do not depend on the location of the grid point* $t[k+3]$.

Cubic splines on the grid $\mathbf{t} = \{t[k]\}$ are represented via the B-splines ([8, 15]). For $t \in [t[k], t[k+1]]$, a cubic spline $s(t)$ is given by

$$s(t) = \sum_{\nu \in \mathbb{Z}} q[\nu+2]\, b(t)[\nu] = \sum_{\nu=k-3}^{k} q[\nu+2]\, b(t)[\nu], \tag{6}$$

where $\mathbf{q} = \{q[\nu]\}$ is a sequence of real numbers, which determines the spline's properties.

Typically, for a spline that approximates a function $f(t)$, the coefficients $q[\nu]$ are derived from the samples $\{f[k] := f(t[k])\}$ of $f(t)$ on the grid $\mathbf{t}$. In that case, the spline is denoted by $s[f]$. For interpolating splines, the coefficients $q[\nu]$ are obtained by solving a tridiagonal system of equations, which involves all the available samples and some boundary conditions. An alternative is provided by the so-called local splines.

Definition 3.2 If the coefficients $q[\nu]$ in Eq. 6 are finite linear combinations of grid samples $\{f[k]\}$, then the spline $s[f]$ is referred to as local spline.

Definition 3.3 If for any polynomial $f(t) = P^n(t)$ of degree n, a spline satisfies $s[f](t) \equiv f(t)$, then it is said that the spline $s[f]$ restores polynomials of degree n and its approximation order is $n+1$.

It is well-known that cubic interpolating splines restore cubic polynomials, thus having the approximation order 4.

Definition 3.4 A local cubic spline, whose approximation order is 4, is referred to as the quasi-interpolating local spline.

To get the simplest cubic spline on the grid $\mathbf{t}$, which approximates the function $f(t)$ from the samples $\{f[k] := f(t[k])\}$, the coefficients in Eq. 6 should be chosen

to be $q_0[\nu] := f[\nu]$. Thus, the spline, which is denoted as $s_0[f]$, is expressed by $s_0[f](t) = \sum_{\nu=k-3}^{k} f[\nu+2]\, b(t)[\nu]$. Its approximation order is 2, thus it restores only first-degree polynomials. For the computation of the spline's values $s_0[f](t)$ at the interval $[t[k], t[k+1]]$, four grid samples $f[k-1]$, $f[k]$, $f[k+1]$ and $f[k+2]$ are needed. The quasi-interpolation can be achieved by adding two more grid samples into the computational scheme.

Denote by

$$\begin{aligned} \beta_{-1}[k] &:= \frac{-(h[k])^2}{3\,h[k-1]\,(h[k-1]+h[k])}, \quad \beta_1[k] := \frac{-(h[k-1])^2}{3\,h[k]\,(h[k-1]+h[k])}, \\ \beta_0[k] &:= 1 - \beta_{-1}[k] - \beta_1[k], \end{aligned} \tag{7}$$

where $h[k]$ is the grid step.

Proposition 3.5 ([22]) *If the coefficients in* Eq. 6 *are chosen as*

$$q_1[\nu] := \beta_{-1}[\nu]\, f[\nu-1] + \beta_0[\nu]\, f[\nu] + \beta_1[\nu]\, f[\nu+1],$$

then the spline

$$s_1[f](t) = \sum_{\nu\in\mathbb{Z}} q_1[\nu+2]\, b(t)[\nu] \tag{8}$$

restores cubic polynomials. For the computation of the spline's values $s_1[f](t)$, $t \in [t[k], t[k+1]]$, *six grid samples* $f[k-2], \ldots, f[k+3]$ *are needed.*

Proof Proved by direct computation of the spline's values of the functions $f(t) \equiv 1$, $f(t) := t^r$, $r = 1, 2, 3$. □

Remark 3.6 In the rest of the paper, we deal exclusively with the quasi-interpolating splines $s_1[f]$. Therefore, we drop the index $\cdot_1$. Thus, in the sequel $s[f] := s_1[f]$.

3.1 Quasi-interpolating cubic splines

3.1.1 Computation of quasi-interpolating cubic splines

The quasi-interpolating cubic splines $s[f]$ can be represented in an alternative form, which is based on a relation between the splines $s[f]$ and cubic interpolation polynomials. Such form is computationally efficient because it utilizes the well-established algorithms for the computation of interpolation polynomials, and facilitates extension of the spline to the boundaries of the intervals. In addition, that representation makes it possible to derive sharp estimates of the approximat errors for the splines $s[f]$. Originally, this representation was introduced in [23]. For the paper to be self-contained, we place the proof of the following statement into Appendix.

Recall that $P(t)[k]$ denotes the cubic polynomial which interpolates the function $f(t)$ at the grid points $\{t[k-1], t[k], \ldots, t[k+2]\}$, and $f[k] := f(t[k])$.

Theorem 3.7 *For $t = t[k] + h[k]\,\tau,\ \ \tau \in [0, 1]$, the cubic spline $s[f](t)$, which restores cubic polynomials, is expressed by*

$$s[f](t) = P(t)[k] + F[k-1]\,(1-\tau)^3 + F[k]\,\tau^3, \tag{9}$$

where the coefficients $F[k]$ are

$$F[k] := -f[k-1, k, k+1, k+2, k+3]\,\frac{(h[k])^2\,(h[k+1])^2\,(t[k+3]-t[k-1])}{3\,(t[k+2]-t[k])}. \tag{10}$$

Proof In Appendix.

When the grid $\mathbf{t}$ is uniform and $t[k] = hk,\ \ k \in \mathbb{Z}$, the finite differences are used instead of the DDs to express the spline $s[f](t)$. In this case, when $t = h\,(k+\tau),\ \ \tau \in [0, 1]$, the spline is represented explicitly by

$$\begin{aligned} s[f](t) \;=\; & f[k-1] + \Delta[\mathbf{f}][k-1]\,(1+\tau) + \frac{\Delta^2[\mathbf{f}][k-1]}{2}\,(1+\tau)\,\tau \\ & + \frac{\Delta^3[\mathbf{f}][k-1]}{6}\,(1+\tau)\,\tau\,(\tau-1) - \frac{\Delta^4[\mathbf{f}][k-1]\tau^3 + \Delta^4[\mathbf{f}][k-2](1-\tau)^3}{36}. \end{aligned} \tag{11}$$

□

For the computation of the spline's values $s[f](t)$ at the interval $[t[k], t[k+1]]$, six grid samples $f[k-2]$, $f[k-1]$, $f[k]$, $f[k+1]$, $f[k+2]$ and $f[k+3]$, are needed. Thus, the spline can be computed in real time with a delay of two samples.

3.2 Approximation properties of quasi-interpolating cubic splines

Approximation properties of the spline $s[f](t)$ are close to the properties of the cubic spline $s_i[f](t)$ which interpolates the function $f(t)$ on the grid $\mathbf{t}$ such that $s_i[f](t[k]) = f(t[k]) = f[k]$. Like the interpolating splines, the splines $s[f](t)$ restore cubic polynomials. Denote $\bar{h}[k] := \max_{v=k-2,\ldots,k+2} h[v]$ and $I[k] := [t[k-2], t[k+3]]$.

Theorem 3.8 *For $t \in [t[k], t[k+1]]$, the following error estimation holds*

$$\max_{t\in[t[k],t[k+1]]} |f(t) - s[f](t)| \le \frac{35\bar{h}^4}{48} \max_{t\in I[k]} |f[t, k-1, k, k+1, \ldots, k+2]|\,. \tag{12}$$

If the function $f(t)$ belongs to $C^4[I[k]]$, then for $t \in [t[k], t[k+1]]$, we have

$$\max_{t\in[t[k],t[k+1]]} |f(t) - s[f](t)| \le \frac{35\,(\bar{h}[k])^4}{1152} \max_{t\in I[k]} \left|f^{(4)}(t)\right|. \tag{13}$$

If the grid $\boldsymbol{t}$ *is uniform, at least locally (that means that* $h[\nu] = h$ *for* $\nu = k-2, ..., k+2$*), then*

$$\max_{t\in[t[k],t[k+1]]} |f(t) - s[f](t)| \le \frac{35\,h^4}{1152} \max_{t\in I[k]} \left|f^{(4)}(t)\right| \approx 0.0304\,h^4 \max_{t\in I[k]} \left|f^{(4)}(t)\right|. \tag{14}$$

35/1152 is the least possible constant in the inequality at Eq. 14.

Proof In Appendix. □

Remark 3.9 The estimates in Eq. 14 is sharp in the sense that it becomes an identity for the function $f(t) = t^4$ when $t = h(k + 1/2)$.

For comparison, a similar sharp estimate for interpolating splines $s_i[f]$ defined on the interval $I := [a, b]$ is ([10]):

$$\max_{t\in I} |f(t) - s_i[f](t)| \le \frac{5\,\tilde{h}^4}{384} \max_{t\in I} \left|f^{(4)}(t)\right| \approx 0.0130\,\tilde{h}^4 \max_{t\in I} \left|f^{(4)}(t)\right|, \tag{15}$$

where $\tilde{h}$ is the maximal grid step at the interval I.

Remark 3.10 Unlike the estimate in Eq. 15, the estimates in Eqs. 12 – 14 are local. That means that the estimates on a certain interval depend only on the function's behavior in a close vicinity of this interval.

Remark 3.11 The sharp estimate in Eq. 14 for the uniform grid was established in [23] by the technique different from that used in the proof of Theorem 3.8. The inequality in Eq. 12 makes it possible to establish sharp estimates of the approximation errors for functions from the classes $C^m[I[k]]$, $m < 4$. In the case when the available samples $f[k]$ are noisy, the estimation in Eq. 12 enables us to evaluate the noise contribution into the approximation error.

3.3 Quasi-interpolating cubic splines at finite intervals

Assume that only a finite number of samples $f[\nu] = f(t[\nu])$, $\nu = 0, 1, ..., N$, of the function $f(t)$ are available. We construct the spline $\bar{s}[f](t)$ which quasi-interpolates the function $f(t)$ at the interval $[t[0], t[N]]$. As before, $P(t)[k]$ denotes the polynomial which interpolates the function $f(t)$ at the points $\{t[k-1], ..., t[k+2]\}$. At the inner interval $[t[2], t[N-2]]$, the representation of the spline $s[f](t)$ given in Eq. 9 is valid. We denote

$$A_0[f] := -f[0, 1, 2, 3, 4]\,\frac{(h[2])^2\,(t[4] - t[0])}{3\,h[1]\,(t[3] - t[1])},$$

$$A_N[f] := -f[N-4, N-3, N-2, N-1, N]\,\frac{(h[N-3])^2\,(t[N] - t[N-4])}{3\,h[N-2]\,(t[N-1] - t[N-3])}. \tag{16}$$

Theorem 3.12 *The function*

$$\bar{s}[f](t) := \begin{cases} P(t)[1] + A_0[f]\,(t - t[1])_+^3, & \text{as } t \in [t[0], t[2]]; \\ s[f](t), & \text{as } t \in [t[2], t[N-2]]; \\ P(t)[N-2] + A_N[f]\,(t[N-1] - t)_+^3, & \text{as } t \in [t[N-2], t[N]]; \end{cases} \tag{17}$$

is a cubic spline that quasi-interpolates the function $f(t)$ *on the finite grid* $\mathbf{t}_N = \{t[\nu]\}\,,\ \nu = 0, 1, ..., N$.

Proof : If $t = t[2] + h[2]\,\tau,\ \ \tau \in [0.1]$, then $s[f](t) = P(t)[2] + F[1]\,(1 - \tau)^3 + F[2]\,\tau^3$, where $F[k]$ is given in Eq. 10. However, for the spline extension to $[t[0], t[2]]$, we utilize the polynomial $P(t)[1]$ and represent the spline by

$$s[f](t) = P(t)[1] + s[\tilde{R}](t), \quad \tilde{R}(t) := f(t) - P(t)[1]. \tag{18}$$

□

An additional grid point $t[-1] < t[0]$ is provisionally introduced. Then, the spline $s[\tilde{R}](t) = \sum_{\nu=-1}^{2} \tilde{q}[\nu + 2]\,b(t)[\nu]$. Due to Proposition 3.1, the location of the point $t[-1]$ does not affect the spline's values at the interval $[t[2], t[3]]$.

We need to know the behavior of the spline $s[\tilde{R}](t)$ as $t \to t[2] + 0$. The reminder term $\tilde{R}(t)$ vanishes at the points $\{t[0], ..., t[3]\}$. Therefore, the coefficients are

$$\tilde{q}[1] = \tilde{q}[2] = 0, \quad \tilde{q}[3] = \beta_1[3]\,\tilde{R}(t[4]), \quad \tilde{q}[4] = \beta_0[4]\,\tilde{R}(t[4]) + \beta_1[4]\,\tilde{R}(t[5]),$$

where $\beta_i[\nu]$ are defined in Eq. 7. When $t \in [t[2], t[3]]$, the B-splines $b(t)[1] = \alpha\,(t - t[1])^3 + \gamma\,(t - t[2])^3$ and $b(t)[2] = \delta\,(t - t[2])^3$, where $\alpha,\ \gamma$ and δ are constants. Consequently, at this interval, the spline is structured as $s[\tilde{R}](t) = A_0[f]\,(t - t[1])^3 + B\,(t - t[2])^3$. When $t = t[2]$, we have

$$s[\tilde{R}](t[2]) = s[f](t[2]) - P(t[2])[1] = s[f](t[2]) - f[2] = A_0[f]\,(h[1])^3. \tag{19}$$

Now, using Eqs. 9 and 10, we get

$$A_0[f] = \frac{s[f](t[2]) - f[2]}{(h[1])^3} = \frac{F[1]}{(h[1])^3} = -f[0, 1, 2, 3, 4]\,\frac{(h[2])^2\,(t[4] - t[0])}{3\,h[1]\,(t[3] - t[1])}. \tag{20}$$

In addition to Eq. 19, the following relations for the derivatives hold:

$$s'[\tilde{R}](t[2]) = 3A_0[f]\,(h[1])^2, \quad s''[\tilde{R}](t[2]) = 6A_0[f]\,h[1]. \tag{21}$$

We define the function $\varphi_0(t) := P(t)[1] + A_0[f]\,(t - t[1])_+^3$. This function consists of two pieces of cubic polynomials, which are glued at the point $t[1]$ such that $\varphi_0(t) \in C^2[t[0], t[2]]$. On the other hand,

$$\begin{aligned} \varphi_0(t[2]) &= P(t[2])[1] + A_0[f]\,(h[1])^3 = f[2] + A_0[f]\,(h[1])^3 = s[f](t[2]), \\ \varphi_0'(t[2]) &= P(t[2])'[1] + 3A_0[f]\,(h[1])^2 = s[f]'(t[2]), \\ \varphi_0^{(2)}(t[2]) &= P(t[2])^{(2)}[1] + 6A_0[f]\,(h[1])^2 = s[f]^{(2)}(t[2]). \end{aligned}$$

Therefore, the function

$$\hat{s}[f](t) := \begin{cases} \varphi_0(t), & \text{as } t \in [t[0], t[2]]; \\ s[f](t), & \text{as } t \in [t[2], t[N-2]] \end{cases}$$

is a cubic spline.

The design for the extension of the spline to $[t[N-2], t[N]]$ is similar to the design at the left hand side of the interval.

Remark 3.13 Five grid samples are needed for the computation of the spline $\bar{s}[f](t)$ at the interval $[t[1], t[2]]$ and only four samples for the interval $[t[0], t[1]]$. A similar situation takes place at the intervals $[t[N-2], t[N-1]]$ and $[t[N-1], t[N]]$.

Remark 3.14 The spline $\bar{s}[f](t)$ defined by Eq. 17 interpolates the function $f(t)$ at the points $t[0]$, $t[1]$ and $t[N-1]$, $t[N]$. At the intervals $[t[0], t[1]]$ and $[t[N-1], t[N]]$, the spline $\bar{s}[f](t)$ coincides with the interpolating polynomials $P(t)[1]$ and $P(t)[N-2]$, respectively.

Theorem 3.15 *The following error estimate is true for* $t \in [t[1], t[2]]$*:*

$$\max_{t\in[t[1],t[2]]} |f(t) - \bar{s}[f](t)| \le \frac{\bar{h}^4(16-3\sqrt{2})}{12\sqrt{2}} \max_{t\in I[0]} |f[0,1,2,3,t]|, \tag{22}$$

where $\bar{h} := \max_{\nu=0,1,2,3} h[\nu]$*. If a function* $f(t) \in C^4[I[0]]$*, then for* $t \in [t[1], t[2]]$ *we have*

$$\max_{t\in[t[1],t[2]]} |f(t) - \bar{s}[f](t)| \le \frac{\bar{h}^4(16-3\sqrt{2})}{288\sqrt{2}} \max_{t\in I[0]} \left| f^{(4)}(t) \right|. \tag{23}$$

If the grid $\boldsymbol{t}$ *is uniform, at least locally (it means that* $h[\nu] = h$ *for* $\nu = 0, 1, 2, 3$*), then*

$$\max_{t\in[t[1],t[2]]} |f(t) - \bar{s}[f](t)| \le \frac{h^4(16-3\sqrt{2})}{288\sqrt{2}} \max_{t\in I[0]} |f^{(4)}(t)| \approx 0.0497\, h^4 \max_{t\in I[0]} |f^{(4)}(t)|. \tag{24}$$

The constant $(16-3\sqrt{2})/288\sqrt{2} \approx 0.0289$ *is the least possible in the inequality at* Eq. 24.

For $t \in [t[0], t[1]]$*, the following error estimate is true:*

$$\max_{t\in[t[0],t[1]]} |f(t) - \bar{s}[f](t)| \le \frac{\bar{h}^4}{24} \max_{t\in I[0]} |f^{(4)}(t)|. \tag{25}$$

The estimate is sharp even for the uniform grid. It becomes an identity for the function $f(t) = t^4$*. Similar estimates hold at the interval* $[t[N-2], t[N]]$*.*

Proof In Appendix. □

3.4 Extrapolation of signals using cubic splines

Assume that a continuous function $f(t)$ is supported on the interval $[t[0], t[N+1]]$ but only the samples $\{f[\nu] = f(t[\nu]\}, \ \nu = 0, ..., N$, are available. In order to approximate the function $f(t)$ on the interval $I_r := [t[N], t[N+1]]$ and, in particular, to "predict" the sample $f[N+1]$, we extend the quasi-interpolating spline $\bar{s}[f](t)$ constructed in Theorem 3.12 to the interval I_r.

Define the extended spline $s_r[f](t)$ by

$$s_r[f](t) := \begin{cases} \bar{s}[f](t), & \text{as } t \in [t[0], t[N]] \ ; \\ P(t)[N-2] + A_r[f]\,(t - t[N])^3, & \text{as } t \in I_r \end{cases} \tag{26}$$

and choose the constant $A_r[f]$ such that, under some conditions, the difference $D_r[f] := f(t[N+1]) - s_r[f](t[N+1])$ is minimal. The function $s_r[f](t)$ is a cubic spline because it is piece-wise cubic polynomial and continuous together with its first and second derivatives at the point $t[N]$.

The difference $D_r[f]$ is

$$\begin{aligned} D_r[f] \ &= \ f(t[N+1]) - P(t[N+1])[N-2] - A_r[f]\,(h[N])^3 \\ &= \ C_r[f]\,f[N-3, N-2, N-1, N, N+1] - A_r[f]\,(h[N])^3, \\ C_r[f] \ &:= \ (t[N+1] - t[N-3])(t[N+1] - t[N-2])(t[N+1] - t[N-1])h[N]. \end{aligned}$$

We choose $A_r[f] := C_r[f]\,f[N-4, N-3, N-2, N-1, N]/(h[N])^3$. Then, the difference becomes

$$\begin{aligned} D_r[f] \ &= \ C_r[f] \\ &\quad (f[N-3, N-2, N-1, N, N+1] - f[N-4, N-3, N-2, N-1, N]) \\ &= \ C_r[f]\,f[N-4, N-3, N-2, N-1, N, N+1]\,(t[N+1] - t[N-4]). \end{aligned}$$

Denote $h_r := \max_{\nu=N-3,...,N} h[\nu]$.

Proposition 3.16 *If the fifth-order DD satisfies $|f[N-4, N-3, N-2, N-1, N, N+1]| \le F$, then the extrapolation error at the point $t[N+1]$ is estimated by*

$$|f(t[N+1]) - s_r[f](t[N+1])| \le 120 F\, h_r^5.$$

If the function $f(t)$ has a continuous fifth-order derivative at the interval $[t[N-4], t[N+1]]$, then the estimation

$$|f(t[N+1]) - s_r[f](t[N+1])| \le \max_{t \in [t[N-4], t[N+1]]} |f^{(5)}(t)|\, h_r^5 \tag{27}$$

is true. In particular, if $f(t)$ coincides with a fourth-degree polynomial on the interval $[t[N-4], t[N+1]]$, then $s_r[f](t[N+1]) = f(t[N+1])$.

A similar design is carried out at the left hand side of the sampling interval. In order to extrapolate the spline $\bar{s}[f](t)$ defined on the grid $t[0], ..., t[N]$ to a point $t[-1] < t[0]$, we define the new spline as follows:

$$s_l[f](t) := \begin{cases} \bar{s}[f](t), & \text{as } t \in [t[0], t[N]] \ ; \\ P(t)[1] + A_l[f]\,(t - t[0])^3, & \text{as } t \in I_l, \end{cases} \tag{28}$$

where the interval $I_l := [t[-1], t[0]]$ and

$$A_l[f] = -f[0, 1, 2, 3, 4]\,(t[1]-t[-1])\,(t[2]-t[-1])\,(t[3]-t[-1])/h[-1]^2. \quad (29)$$

3.5 Remarks on the real-time spline computation

Due to the fact that no more than 6 adjacent grid samples are needed for the computation of the spline $s[f]$ at a point t, the spline can be computed in real time. It means that the spline follows the samples that arrive one after another at random times.

Assume that the samples of a function $f(t)$, $\{f[k]\}$, $k = 0, ..., N$, are available, where $f[k] = f(t[k])$. Then, the spline $s[f]$ can be designed on the interval $[t[0], t[N]]$ in line with the scheme described in Sections 3.2 and 3.3 in the following way:

- At the inner subinterval $[t[2], t[N-2]]$, the spline is constructed by a regular 6-samples algorithm from Section 3.2, while at the intervals $[t[0], t[2]]$ and $[t[N-2], t[N]]$ the 5- and 4-samples extension formulas from Section 3.3 are utilized.
- When the sample $f[N+1] = f(t[N+1])$ arrives, the spline at the interval $[t[N-2], t[N-1]]$ is recomputed by utilizing 6 samples $\{f[k]\}$, $k = N-4, ..., N+1$. The spline at the interval $[t[N-1], t[N]]$, which was constructed by using 4 samples $\{f[k]\}$, $k = N-3, ..., N$, is recomputed with the new set of samples $\{f[k]\}$, $k = N-3, ..., N+1$. The spline is extended to the interval $[t[N], t[N+1]]$ by using the samples $\{f[k]\}$, $k = N-2, ..., N+1$. The spline remains unchanged at the interval $[t[0], t[N-2]]$.
- When the sample $f[N+2] = f(t[N+2])$ arrives, the spline at the interval $[t[N-1], t[N]]$ is recomputed by utilizing 6 samples $\{f[k]\}$, $k = N-3, ..., N+2$. The spline at the interval $[t[N], t[N+1]]$, which was constructed by using 4 samples $\{f[k]\}$, $k = N-2, ..., N+1$, is recomputed with the new set of samples $\{f[k]\}$, $k = N-2, ..., N+2$. The spline is extended to the interval $[t[N+1], t[N+2]]$ by using samples $\{f[k]\}$, $k = N-1, ..., N+2$. The spline remains unchanged at the interval $[t[0], t[N-1]]$.

Remark 3.17 Note that the arrival of two additional samples $f[N+1]$ and $f[N+2]$ leads to the re-computation at the interval $[t[N-2], t[N]]$ of the initial spline, which was defined at $[t[0], t[N]]$, while the spline remains unchanged at the interval $[t[0], t[N-2]]$. This fact substantiates our claim that the spline follows the arriving samples.

The above scheme is illustrated in Example 2 in Section 3.6.

3.6 Examples

Example 1: Restoration of the *sine* function from randomly located samples: Figure 1 illustrates results from the restoration and extrapolation experiments of the function $\sin(t) + \sin 2t$ from 30 randomly spaced samples (top plot) and from uniformly spaced samples (bottom plot). In both cases, the first and the last samples

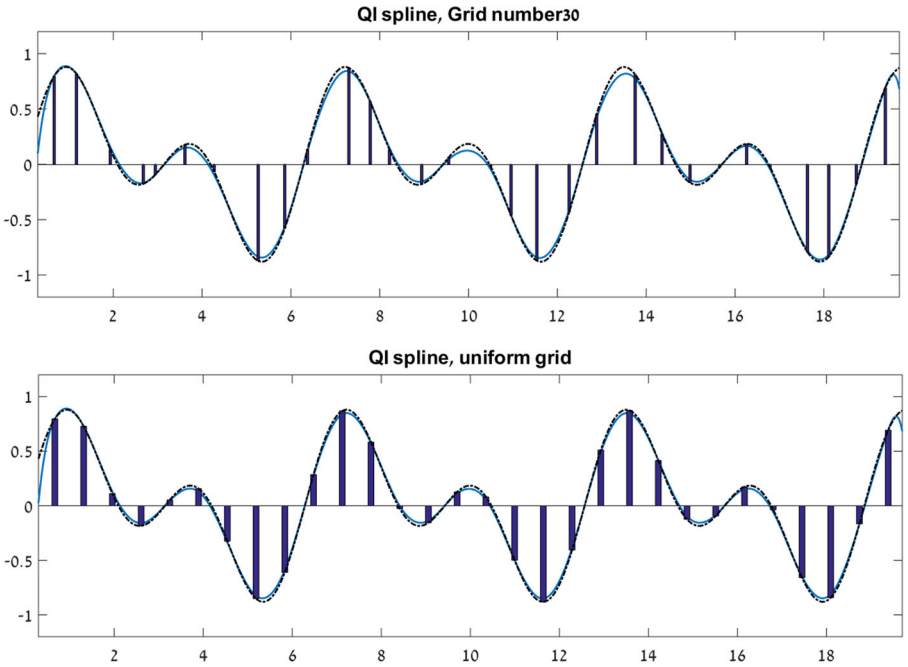

Fig. 1 Top: restoration of the function $f(t) = \sin t + \sin 2t$ from 30 randomly spaced samples by the quasi-interpolating $s[f](t)$. Bottom: restoration by $s[f](t)$ from 30 equally spaced samples. Dash-dot lines denote the original function and bars denote the available samples. The splines $s[f](t)$ are displayed by solid lines

were taken at $t[0] = 0.67$ and $t[N] = 19.23$ and the splines were extrapolated to $t[-1] = 0.3$ and $t[N+1] = 19.7$.

We observe that the splines at either grid restore the function near perfectly. However, the extrapolated samples differ from the original ones. It happens due to the fact that in both cases the discrepancy of the differences $f[N-3, N-2, N-1, N, N+1] - f[N-4, N-3, N-2, N-1, N]$ is essential. A similar situation exists at the left hand side of the grid. However, when the function $f(t)$ is a fourth-degree polynomial, the extrapolation is exact. It is displayed in Fig. 2. The function $f(t)$ is a fourth-degree polynomial. The spline is construcred using 6 randomly spaced grid points. The first and last samples were taken at $t[0] = 0.35$ and $t[N] = 9.9$ and the spline was extrapolated to $t[-1] = -3$ and $t[N+1] = 13$.

Example 2: Illustration of the real-time spline computation: Figure 3 illustrates the scheme for the real-time spline computation when samples of a function to be approximated by the spline arrive sequentially at random times. The scheme was described in Section 3.5. Initially, the spline was designed on the interval $[t[0], t[N]]$ where in this example $N = 9$. In the figure, it is shown by the dashed curve. On the arrival of the sample $f[N+1]$, the spline is extended to the interval $[t[N], t[N+1]]$. In the process, the spline is recomputed at the interval $[t[N-2], t[N]]$. The extended

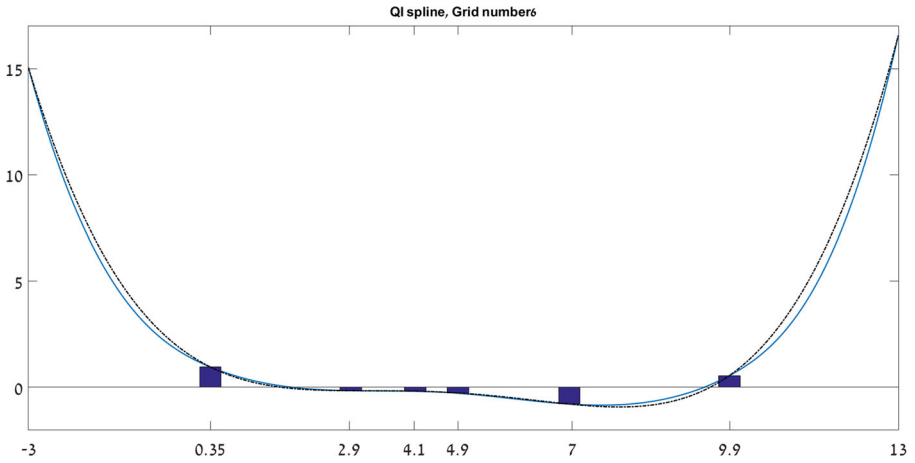

Fig. 2 Extrapolation of the spline $s[f](t)$ from six grid points denoted by bars. Dashed line denotes the original function and solid line denotes the spline $s[f](t)$

spline is denoted by a dash-dot curve. On the arrival of the sample $f[N+2]$, the spline is extended to the interval $[t[N+1], t[N+2]]$. In the process, the spline is updated at the interval $[t[N-1], t[N+1]]$. The extended spline is denoted by a solid line.

4 Spline-based wavelet transform

The *Lifting Scheme*, introduced in [18, 19], is a method that constructs bi-orthogonal wavelet transforms and provides their efficient implementation. The main feature of the lifting scheme is that all the constructions are derived directly in the spatial

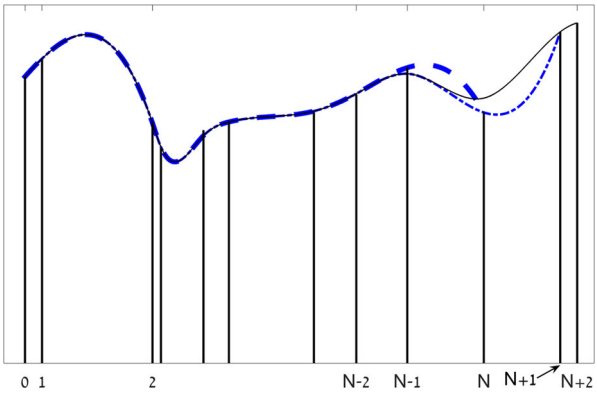

Fig. 3 *Dashed line*: spline $s[f](t)$ computed on the interval $[t[0], t[N]]$. *Dash-dot line*: the spline is extended to the interval $[t[N], t[N+1]]$. *Solid line*: the spline is extended to the interval $[t[N+1], t[N+2]]$

domain and therefore can be custom designed to more general, irregular settings such as non-uniformly spaced data samples and bounded intervals. In addition, the lifting scheme fits well to the real-time execution of the wavelet transforms. In this section, we outline the lifting scheme and describe how to use the local quasi-interpolating cubic splines, designed in Section 3, for the construction of wavelet transforms of non-equally sampled discrete-time signals and in the situation when signal's samples arrive subsequently at random time moments.

4.1 Discrete lifting wavelet transform

The lifting wavelet transform of a discrete-time signal can be implemented either in a *primal* or in a *dual* mode. We outline only the primal mode.

4.1.1 Primal decomposition

The lifting wavelet decomposition of a signal from the space l_1 consists of four steps:

1. **Split**: The signal $\mathbf{f} = \{f[k] = f[t[k]]\}$ is split into the even and the odd sub-arrays such that $\mathbf{f} = \mathbf{e} \bigcup \mathbf{o}$ where $\mathbf{e} := \{e[k] = f[2k]\}$ and $\mathbf{o} := \{o[k] = f[2k+1]\}$.
2. **Predict**: The even array $\mathbf{e}$ is used to approximate (predict) the odd array $\mathbf{o}$. Then, the array $\mathbf{o}$ is replaced by the array $\mathbf{d} = \mathbf{o} - \mathcal{P}\,\mathbf{e}$ where $\mathcal{P}$ is a linear $l_1 \to l_1$ *prediction* operator. If the predictor is correctly chosen, then this step decorrelates the signal and reveals its high-frequency components.
3. **Update(Lifting)**: The even array $\mathbf{e}$ is updated by using the new odd array $\mathbf{d}$. For this, we use a linear $l_1 \to l_1$ *updating* operator $\mathcal{U}$, which is assumed to commute with $\mathcal{P}$, to get $\mathbf{a} = \mathbf{e} + \mathcal{U}\,\mathbf{d}$. Provided that the updating operator is properly chosen, the even array $\mathbf{e}$ is transformed into a downsampled and smoothed replica of $\mathbf{f}$.
4. **Normalization**: Finally, the smoothed $\mathbf{y}^0$ and the details $\mathbf{y}^1$ transform coefficient arrays, are obtained by the normalization $\mathbf{y}^0 = \sqrt{2}\mathbf{a}, \quad \mathbf{y}^1 = \mathbf{d}/\sqrt{2}$.

4.1.2 Primal reconstruction

Reconstruction of the signal $\mathbf{f}$ from the arrays $\mathbf{y}^0$ and $\mathbf{y}^1$ is implemented in a reverse order:

1. **Undo Normalization**: $\mathbf{a} = \mathbf{y}^0/\sqrt{2}, \quad \mathbf{d} = \sqrt{2}\mathbf{y}^1$.
2. **Undo Lifting**: The even array is restored by $\mathbf{e} = \mathbf{a} - \mathcal{U}\,\mathbf{d}$.
3. **Undo Predict**: The odd array component is restored by $\mathbf{o} = \mathbf{d} + \mathcal{P}\,\mathbf{e}$.
4. **Undo Split**: Restoration of the signal from its even and odd arrays by $\mathbf{f} = \mathit{Merge}\{\mathbf{e}, \mathbf{o}\}$.

The lifting transform is perfectly invertible with any choice of the operators $\mathcal{P}$ and $\mathcal{U}$. The direct and inverse transforms can be symbolically represented in a matrix form:

$$\begin{pmatrix} \mathbf{y}^0 \\ \mathbf{y}^1 \end{pmatrix} = \begin{pmatrix} \sqrt{2}\mathrm{I} & 0 \\ 0 & \mathrm{I}/\sqrt{2} \end{pmatrix} \begin{pmatrix} \mathrm{I} & \mathcal{U} \\ 0 & \mathrm{I} \end{pmatrix} \begin{pmatrix} \mathrm{I} & 0 \\ -\mathcal{P} & \mathrm{I} \end{pmatrix} \begin{pmatrix} \mathbf{e} \\ \mathbf{o} \end{pmatrix} = \tilde{\mathbf{M}} \begin{pmatrix} \mathbf{e} \\ \mathbf{o} \end{pmatrix},$$

where I is the identical operator and the analysis "polyphase matrix" $\tilde{\mathbf{M}}$ of the first decomposition level is

$$\tilde{\mathbf{M}} := \begin{pmatrix} \sqrt{2}\,(\mathrm{I} - \mathcal{U}\,\mathcal{P}) & \sqrt{2}\,\mathcal{U} \\ -\mathcal{P}/\sqrt{2} & \mathrm{I}/\sqrt{2} \end{pmatrix}.$$

The inverse transform is represented by

$$\begin{pmatrix} \mathbf{e} \\ \mathbf{o} \end{pmatrix} = \begin{pmatrix} \mathrm{I} & 0 \\ \mathcal{P} & \mathrm{I} \end{pmatrix} \begin{pmatrix} \mathrm{I} & -\mathcal{U} \\ 0 & \mathrm{I} \end{pmatrix} \begin{pmatrix} \mathrm{I}/\sqrt{2} & 0 \\ 0 & \mathrm{I}\sqrt{2} \end{pmatrix} \begin{pmatrix} \mathbf{y}^0 \\ \mathbf{y}^1 \end{pmatrix} = \mathbf{M} \begin{pmatrix} \mathbf{y}^0 \\ \mathbf{y}^1 \end{pmatrix},$$

where the synthesis "polyphase matrix" $\mathbf{M}$ of the first decomposition level is

$$\mathbf{M} := \begin{pmatrix} \mathrm{I}/\sqrt{2} & -\sqrt{2}\,\mathcal{U} \\ \mathcal{P}/\sqrt{2} & \sqrt{2}\,(\mathrm{I} - \mathcal{P}\,\mathcal{U}) \end{pmatrix}.$$

It is readily seen that, once the operators $\mathcal{P}$ and $\mathcal{U}$ commute with each other, we have $\mathbf{M}\,\tilde{\mathbf{M}} = \mathrm{I}$. Therefore, subsequent applications of the operators $\tilde{\mathbf{M}}$ and $\mathbf{M}$ to the vector $(\mathbf{e},\ \mathbf{o})^T$ restores this vector.

The multilevel wavelet transform is achieved by the iterated application of the lifting operations to the smoothed coefficient arrays. The prediction and updating operators can be different for different decomposition levels. Reconstruction of the signal $\mathbf{f}$ from the coefficient array $\{\mathbf{y}^0_{[m]} \bigcup \mathbf{y}^1_{[m]} \bigcup \cdots \mathbf{y}^1\}$ is implemented in a reverse order.

4.2 Spline-based prediction and updating operators

Wavelet transforms generated by the lifting scheme are determined by the choice of the prediction $\mathcal{P}$ and the updating $\mathcal{U}$ operators. Splines provide flexible tools for the design of such operators. The idea is to construct a spline on the even sub-grid $\{t[2k]\}$, which either interpolates or quasi-interpolates the samples of the signal $\mathbf{e}$. The values of this spline at the odd grid points $\{t[2k+1]\}$ are used for the prediction of the odd samples $\mathbf{o}$. The next step is to construct a spline on the odd grid points, which (quasi-)interpolates the samples of the prediction-error signal $\mathbf{d}$. Then, the values of this splines at the even grid points are computed. These values are used for updating the even samples $\mathbf{e}$.

In the case of equally spaced samples, calculations are reduced to low-pass filtering of the corresponding arrays [4]. Application of wavelet transforms to signals sampled on finite grids and, in particular, to images, requires to extend the signals beyond their boundaries, otherwise distortion appears near the boundaries [5]. Various extension schemes have been developed to deal with the boundary effects of finite-length signals: zero padding, periodic extension and symmetric extension are basic extension methods. However, quasi-interpolating splines on finite intervals designed in Section 3.3, make it possible to implement wavelet transforms of signals sampled on bounded intervals without extending the signals beyond their boundaries.

The prediction and updating operations are reduced to the design of the splines $s[f]$ on different grids and computation of their values at intermediate points.

4.2.1 Prediction and updating operators on unlimited grids

To highlight the structure of the operators, we use a standard representation of the quasi-interpolating spline via the B-splines given in Eq. 8:

$$\begin{aligned} s[f](t) &= \sum_{\nu\in\mathbb{Z}} q_1[\nu+2]\, b(t)[\nu] \\ &= \sum_{\nu\in\mathbb{Z}} (\beta_{-1}[\nu+2]\, f[\nu+1]+\beta_0[\nu+2]\, f[\nu+2]+\beta_1[\nu+2]\, f[\nu+3])\; b(t)[\nu] \\ &= \sum_{\nu\in\mathbb{Z}} f[\nu+2]\, B(t)[\nu], \end{aligned} \tag{30}$$

where the coefficients $\beta_i[k]$ are defined by Eq. 7 and the cubic spline $B(t)[k]$ is

$$B(t)[k] := \beta_{-1}[k+3]\, b(t)[k+1] + \beta_0[k+2]\, b(t)[k] + \beta_1[k+1]\, b(t)[k-1]. \tag{31}$$

The spline $B(t)[k]$ is supported on the interval $(t[k-1], t[k+5])$. Therefore, if $t \in [t[k], t[k+1]]$, then the sum in Eq. 30 comprises only 6 terms:

$$s[f](t) = \sum_{\nu=k-4}^{k+1} f[\nu+2]\, B(t)[\nu]. \tag{32}$$

As before, $\mathbf{e} = \{e[k] = f[2k]\}$ and $\mathbf{o} = \{o[k] = f[2k+1]\}$. Denote by $b_e(t)$ and $B_e(t)$ the B-spline $b(t)$ and the spline $B(t)$, respectively, defined on the even sub-grid $\mathbf{t}_e := \{t[2k]\}$ and by $b_o(t)$ and $B_o(t)$ the splines defined on the odd sub-grid $\mathbf{t}_o := \{t[2k+1]\}$. Equation 32 implies that the values at the odd grid points of the spline $s[e](t)$, which is constructed on the even sub-grid, are

$$s[e](t[2k+1]) = \sum_{\nu=k-4}^{k+1} f[2(\nu+2)]\, B_e(t[2k+1])[2\nu] = \sum_{\nu=k-4}^{k+1} P_{[k]}[\nu]\, e[\nu], \tag{33}$$

$$P_{[k]}[\nu] := \begin{cases} B_e(t[2k+1])[2\nu], & \text{for } \nu = k-4, ..., k+1\ ; \\ 0, & \text{otherwise.} \end{cases}$$

The coefficients $\left\{P_{[k]}[\nu]\right\}$ form a six-diagonal matrix $\mathbf{P}$, which is a Toeplitz matrix in the case of uniform grid. Consequently, the prediction operation can be represented as the matrix multiplication by $\mathcal{P}\,\mathbf{e} = \mathbf{P}\,\mathbf{e}$.

Similarly, the updating operation can be represented as the matrix multiplication: $\mathcal{U}\{\mathbf{d}\} = \mathbf{U}\,\mathbf{d}$, where the six-diagonal matrix $\mathbf{U} = \left\{U_{[k]}[\nu]\right\}$

$$\begin{aligned} s[d](t[2k]) &= \sum_{\nu=k-5}^{k} (f[2(\nu+2)+1]-s[e](t[2(\nu+2)+1]))\, B_o(t[2k])[2\nu+1] \\ &= \sum_{\nu=k-5}^{k} U_{[k]}[\nu]\, d[\nu], \quad U_{[k]}[\nu] := \begin{cases} B_o(t[2k])[2\nu+1], & for\, \nu = \\ & k-5, ..., k; \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{34}$$

Remark 4.1 When processing a single signal and, especially, in the real-time mode, the computations by fast algorithms described in Section 3 are preferable. However, in a case when either multiple signals defined on the same grid or an array of row signals are processed, the necessary values of the splines $\mathbf{B}(t)$ can be pre-computed. Then, according to Eqs. 33 and 34, the computations of the splines' values $s[e](t[2k+1])$ and $s[d](t[2k])$ can be implemented by the six-tap weighted moving averages of the signal's samples with weights consisting of values of the splines $\mathbf{B}(t)$ given in Eq. 31. Thus, computations are significantly accelerated.

4.2.2 Prediction and updating operators on limited grids

Assume that the function f is sampled on a bounded interval, thus $f[k] = f(t[k]),\ k = 0, ..., N$ are available.

Assume that N is an odd integer (Fig. 4):

Prediction: The odd samples $f[2k+1],\ k = 2, ..., (N-1)/2-3$, are predicted using the formulas in Eq. 33. The samples $f[1]$ and $f[3]$ as well as $f[N-4]$ and $f[N-2]$ are predicted by the values of the spline $s[e](t)$ at the respective grid points $t[1]$ and $t[3]$ as well as $t[N-4]$ and $t[N-2]$. These values are computed using the formulas in Eq. 17. The remaining sample $f[N]$ is predicted by the extrapolation of the spline $s[e](t)$ to the grid point $t[N]$ using the algorithm described in Section 3.4. Thus, we derive the differences $d[k] = f[2k+1] - s[e](t[2(k+2)+1]),\ k = 0, ..., (N-1)/2$.

Update: The even samples $f[2k],\ k = 3, ..., (N-1)/2-2$, are updated by the values of the spline $s[d](t[2k])$ using the formulas in Eq. 34. To update the samples $f[2]$, $f[4]$, $f[N-3]$, $f[N-1]$, the values of $s[d](t)$ are computed using the formulas in Eq. 17. The remaining sample $f[0]$ is predicted by the extrapolation of the spline $s[d](t)$ to the grid point $t[0]$ using the algorithm described in Section 3.4.

A similar design is performed when N is an even integer.

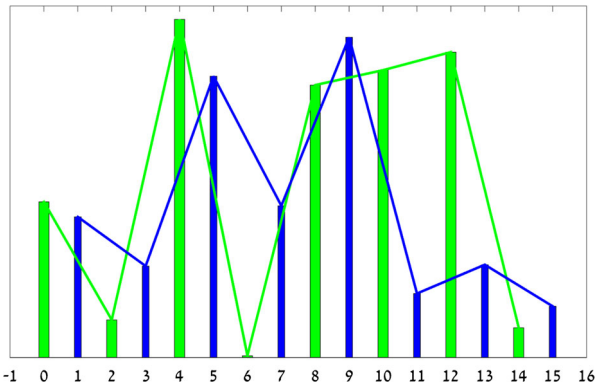

Fig. 4 Random samples, N odd: green – even samples, blue – odd samples

To implement the next step of the wavelet transform, the above operations are applied to the smooth array $\mathbf{y}^0$. As a result, the arrays $\mathbf{y}^0_{[2]}$ and $\mathbf{y}^1_{[2]}$ are produced. This process repeats itself.

The spline- based wavelet transform possesses the local "discrete vanishing moments" property, which is formulated next.

Proposition 4.2 *Assume that* $\{f[\nu]\}$, $\nu = 2k-4, ..., 2k+6$, *are samples of the cubic polynomial* $f[\nu] = P^3(t[\nu])$. *Then, the wavelet transform coefficient* $y^1[k] = 0$.

Proof The first-level transform coefficients are $y^1[k] = (f[2k+1] - s[e](t[2k+1]))/\sqrt{2}$. Theorem 3.8 implies that this difference is zero. □

Remark 4.3 Theorem 3.15 and Proposition 3.16 imply that the "discrete vanishing moments" property remains valid near the boundaries of the interval.

Figure 5 displays a randomly sampled signal and coefficients from its four-level wavelet transform. Note that the inverse wavelet transform from these coefficients provides a restoration of the signal where its maximal deviation from the original, which is displayed in Fig. 6, is $1.5\,10^{-12}$.

4.3 Real-time execution of the wavelet transform

The transform scheme described in Section 4.2.2 makes it possible to execute the wavelet transform of a signal in real time. It means that the samples $f[\nu]$ of a signal

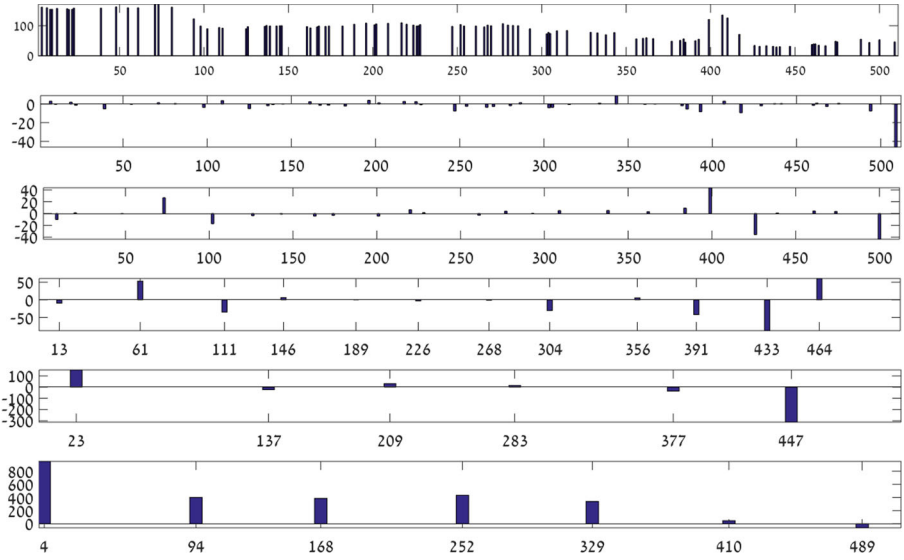

Fig. 5 Wavelet transform of a randomly sampled signal. Top: the source signal. Bottom: smooth coefficients from the fourth decomposition level. The other 4 plots: the detail coefficients from first to fourth decomposition levels

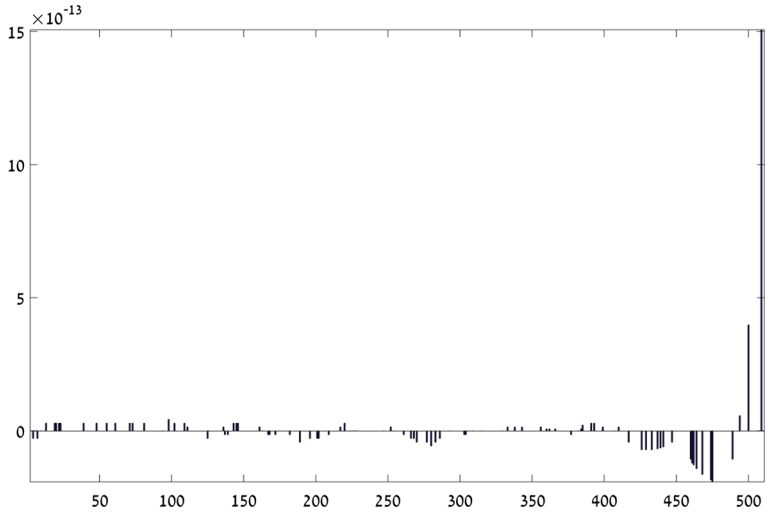

Fig. 6 Discrepancy between the original signal and its restoration from the wavelet coefficients shown in Fig. 5. Maximal discrepancy is $1.5\,10^{-12}$

arrive sequentially at random time moments $t[\nu],\ \nu = 0, 1, ...$, while the arrival of a new sample requires recomputing of only a few adjacent transform coefficients.

- Assume that at the time $t[N]$ the samples $f[\nu],\ \nu = 0, ..., N$, arrived already and N is an odd number. Then, the one-level decomposition is implemented as was described in Section 4.2.2. Recall that the initial coefficients $y^0[0]$, $y^0[1]$, $y^0[2]$ and $y^1[0]$, $y^1[1]$ are produced by using the formulas in Eq. 17 and the extrapolation algorithm. The coefficients $y^0[\nu],\ \nu = 3, ..., (N-1)/2-2$, and $y^1[\nu],\ \nu = 2, ..., (N-1)/2-3$, are computed by using the regular formulas in Eq.33. The coefficients $y^0[(N-1)/2-1]$, $y^0[(N-1)/2]$, and $y^1[(N-1)/2-1]$, $y^1[(N-1)/2-2]$ are produced by using the formulas in Eq. 17. The coefficient $y^1[(N-1)/2]$ is derived by the extrapolation algorithm. The number $(N+1)/2$ of smooth coefficients $y^0[\nu]$ is the same as the number of the detail coefficients $y^1[\nu]$.
- When the sample $f[N+1]$ arrives, the new smooth coefficient $y^0[(N+1)/2]$ is derived by updating the even sample $f[N+1]$ using the extrapolation of the spline $s[d](t)$ to the grid point $t[N+1]$. In addition, the detail coefficient $y^1[(N-1)/2-2]$ is recomputed in a regular way by prediction from 6 even samples $f[2\nu],\ \nu = (N-1)/2-4, ...(N+1)/2$. The rest of the detail and the smooth coefficients remain unchanged. The number of the smooth coefficients $y^0[\nu]$ is $(N+3)/2$ while the number of the detail coefficients $y^1[\nu]$ is $(N+1)/2$.
- When the sample $f[N+2]$ arrives, the new detail coefficient $y^1[(N+1)/2]$ is derived by predicting the odd sample $f[N+2]$ using the extrapolation of the spline $s[e](t)$ to the grid point $t[N+2]$. In addition, the smooth coefficient $y^0[(N-1)/2-1]$ is recomputed in a regular way by updating from 6 numbers $d[\nu],\ \nu = (N-1)/2-4, ...(N+1)/2$. The rest of the detail and smooth

coefficients remain unchanged. The number $(N+3)/2$ of smooth coefficients $y^0[\nu]$ is the same as the number of the detail coefficients $y^1[\nu]$.

At the same time, the above procedures are applied to the smooth coefficient array $\left\{y^0[\nu]\right\}$, $\nu = 0, ..., (N-1)/2$, to obtain the arrays $\left\{y^0_{[2]}[\nu]\right\}$ and $\left\{y^1_{[2]}[\nu]\right\}$, and so on.

CPU example An irregular grid was defined as $\mathbf{t}\,\{t[k] = k + \epsilon[k]\}$, $k = 1, ..., N$, where $N = 20000$ and $\epsilon[k]$ are Gaussian random variables, whose STD=0.2. The initial 10 grid points are

k	1	2	3	4	5	6	7	8	9	10
t[k]	0.8608	1.8889	3.1780	3.9217	5.0921	5.7455	6.5421	8.1339	9.0181	10.2559

The transforms were applied to two signals $\mathbf{x}_1$ and $\mathbf{x}_2$ such that :

$$x_1[k] := \sin\frac{t[k]\,50}{N}, \quad [x_2[k] := \sin\frac{t[k]\,50}{N} + \epsilon[k], \ , k = 1, ..., N.$$

Figure 7 displays the signals $\mathbf{x}_1$ and $\mathbf{x}_2$.

The 10-level lifting wavelet transform was applied to the signals $\mathbf{x}_1$ and $\mathbf{x}_2$, then signals were restored from the transform coefficients by the inverse wavelet transform. The transforms were executed by non-optimized Matlab codes on the PC with the Intel(R) Core(TM) i7-3770 CPU@3.40 GHz processor. Herewith, the CPU time was *3.357637 seconds* for the direct transform and *3.374883 seconds* for the inverse transform.

The differences between the original $\mathbf{x}_1$ and $\mathbf{x}_2$ signals and the restored $\mathbf{X}_1$ and $\mathbf{X}_2$ ones are displayed in Fig. 8. In both cases the differences do not exceed $1.5 \cdot 10^{-15}$. We emphasize that no boundary effects arrived in process of the signals' restoration

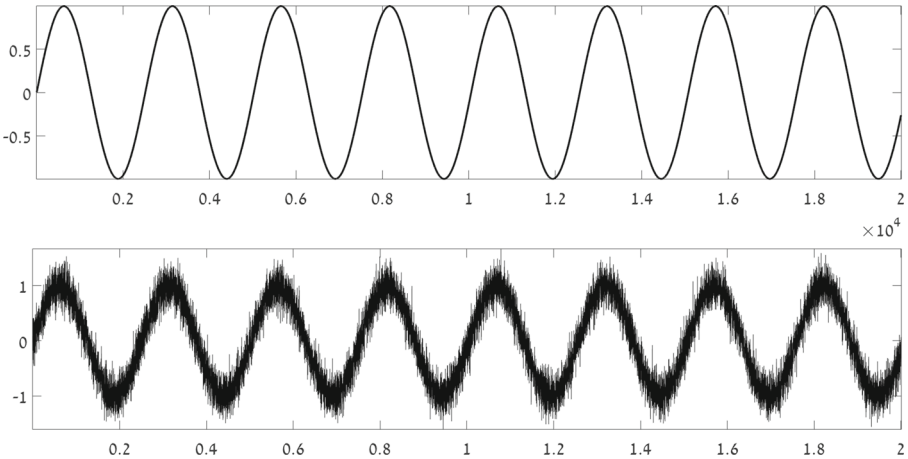

Fig. 7 Top: signal $\mathbf{x}_1$. Bottom: signal $\mathbf{x}_2$

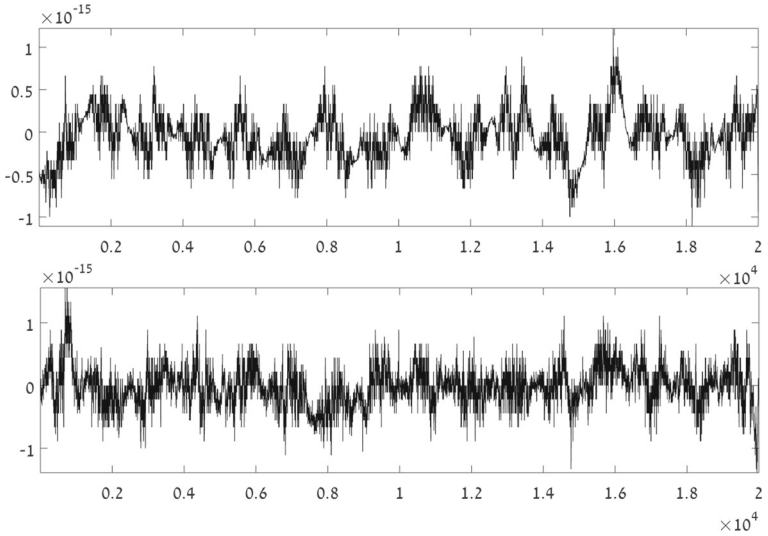

Fig. 8 Differences between original and restored signals. Top: signal $\mathbf{x}_1 - \mathbf{X}_1$. Bottom: $\mathbf{x}_2 - \mathbf{X}_2$

from wavelet transforms even for highly irregular signal $\mathbf{x}_2$ displayed in the bottom frame in Fig. 7.

5 Conclusion

The local cubic splines, which were designed in Section 3 and supplied with a simple fast computational algorithms, can serve as an efficient tool for the real-time signal processing. As an input, the splines use either clean or noisy arbitrarily-spaced samples. Sharp estimates of the approximation error were established. An important application of the designed splines is the real-time wavelet analysis of non-uniformly sampled signals.

Summary of the paper's contributions

- Simple algorithm for the real-time computation of the local cubic quasi-interpolating splines including a smooth extension of the splines to the boundaries of the sampling interval.
- Formulas for the extrapolation of the splines beyond the sampling interval, while retaining the approximation accuracy. The formulas can be used for the *prediction-correction* signal processing.
- Sharp estimates of the approximation errors for the local cubic quasi-interpolating splines.
- Design of wavelet transforms of the arbitrarily sampled signals based on the local cubic quasi-interpolating splines. The wavelet transforms have four local "discrete vanishing moments" in the sense of Proposition 4.2.

- Real-time scheme of the execution of the wavelet transforms of signals whose samples arrive dynamically and sequentially at random times.

In future work, the approach will be extended to the higher-order splines. The designed splines and spline- based wavelet transforms will be applied to online denoising and compression of practical signals such as, for example, ECG signals. Another potential field of application of the developed techniques is the real-time process control including detection of specific transient events.

Appendix

We recall the elementary Intermediate Value Theorem (IVT): *If a function $f(t)$ is continuous on the interval $[a, b]$ and α and β are real numbers with the same sign, then there exists a point $c \in [a, b]$ such that $\alpha f(a) + \beta f(b) = (\alpha + \beta) f(c)$.*

Proof of Theorem 3.7 Due to Eq. 4, the reminder term of the interpolation is

$$R(t) := f(t) - P(t)[k] = f[t, k-1, k, k+1, k+2]$$
$$(t - t[k-1])\,(t - t[k])\,(t - t[k+1])\,(t - t[k+2]).$$

Grid samples are $R(t[\nu]) = 0$ for $\nu = k-1, k, k+1, k+2$. For $\nu = k-2, k+3$, we have

$$\begin{aligned} R(t[k-2]) &= h[k-2]\,T[k]\,(t[k] - t[k-2])\,(t[k+1] - t[k-2]), \\ R(t[k+3]) &= h[k+2]\,T[k+1]\,(t[k+3] - t[k])\,(t[k+3] - t[k+1]), \\ T[k] &:= f[k-2, k-1, k, k+1, k+2]\,(t[k+2] - t[k-2]). \end{aligned} \tag{35}$$

The spline $s[f](t)$ restores the polynomial $P(t)[k]$, therefore,

$$\begin{aligned} s[f](t) &= s[P](t) + s[R](t) = P(t)[k] + s[R](t) \\ &= P(t)[k] + \sum_{\nu=k-3}^{k} Q[\nu+2]\,b(t)[\nu], \\ Q[\nu+2] &= \beta_{-1}[\nu+1]\,R(t[\nu+1]) \\ &\quad + \beta_0[\nu+2]\,R(t[\nu+2]) + \beta_1[\nu+3]\,R(t[\nu+3]). \end{aligned}$$

Keeping in mind that $R(t[\nu]) = 0$ for $\nu = k-1, k, k+1, k+2$, we get

$$\begin{aligned} s[f](t) &= P(t)[k] + \beta_{-1}[k-1]\,R(t[k-2])\,b(t)[k-3] \\ &\quad + \beta_1[k+2]\,R(t[k+3])\,b(t)[k]. \end{aligned} \tag{36}$$

Equation 5 implies that for $t = t[k] + h[k]\,\tau,\ \tau \in [0, 1]$,

$$b(t)[k] = (t[k+4] - t[k])\,\frac{(t - t[k])^3}{w_4'[k](t[k])} = \frac{(h[k])^2\,\tau^3}{(t[k+2] - t[k])\,(t[k+3] - t[k])}, \tag{37}$$

$$b(t)[k-3] = \frac{(h[k])^2\,(1-\tau)^3}{(t[k+1] - t[k-2])(t[k+1] - t[k-1])}. \tag{38}$$

The explicit expression in Eq. 9 for the spline $s[f](t)$ is obtained by substitution of the B-spline samples from Eqs. 37 and 38, the coefficients β_j from Eq. 7 and the samples of $R(t)$ from Eq. 35 into Eq. 36. □

Proof of Theorem 3.8 As before, $\tau = (t - t[k])/h[k] \in [0, 1]$. By using Eqs. 9 and 4, we can represent the remainder term of the approximation as follows:

$$\begin{aligned}\rho(t) &:= f(t) - s[f](t) = f(t) - P^3(t)[k] + F[k-1]\,(1-\tau)^3 + F[k]\,\tau^3 \\ &= \alpha\, f[t, k-1, k, k+1, ..., k+2] + \beta\, f[k-1, k, k+1, k+2, k+3] \\ &\quad + \gamma\, f[k-2, k-1, k, k+1, k+2],\end{aligned}$$

where the coefficients are

$$\begin{aligned}\alpha &= \omega_3[k-1](t) = (t - t[k-1])\,(t - t[k])\,(t - t[k+1])\,(t - t[k+2]) \geq 0, \\ \beta &= \frac{(h[k])^2\,(h[k+1])^2\,(t[k+3] - t[k-1])}{3\,(t[k+2] - t[k])}\,\tau^3 \geq 0, \\ \gamma &= \frac{(h[k])^2\,(h[k-1])^2\,(t[k+2] - t[k-2])}{3\,(t[k+1] - t[k-1])}\,(1-\tau)^3 \geq 0.\end{aligned}$$

The IVT implies that

$$\rho(t) = f[\vartheta, k-1, k, k+1, ..., k+2]\,(\alpha + \beta + \gamma),$$

where $\vartheta \in (t[k-2], t[k+3])$. Denote $H(\tau) := \alpha + \beta + \gamma$. Then, we have

$$\begin{aligned}H(\tau) &= (h[k])^2\Big((h[k+1] + h[k](1-\tau))(h[k-1] + h[k]\tau)\,\tau\,(1-\tau) \\ &\quad + \frac{(h[k+1])^2\,(t[k+3] - t[k-1])}{3\,(h[k+1] + h[k])}\,\tau^3 \\ &\quad + \frac{(h[k-1])^2\,(t[k+2] - t[k-2])}{3\,(h[k] + h[k-1])}\,(1-\tau)^3\Big).\end{aligned}$$

It is readily verified that

$$H(\tau) \leq \bar{h}^4\left((2-\tau)(1+\tau)\tau(1-\tau) + \frac{2}{3}(\tau^3 + (1-\tau)^3)\right) \leq \frac{35}{48}\,\bar{h}^4,$$

where $\bar{h} := \max_{\nu=k-2,...,k+2} h[\nu]$. Consequently, for $t \in [t[k], t[k+1]$ we have the estimation

$$|\rho(t)| \leq \frac{35\bar{h}^4}{48} \max_{\vartheta \in (t[k-2], t[k+3])} |f[\vartheta, k-1, k, k+1, ..., k+2]|\,.$$

Equation 13 follows from Eq. 3. If $h[\nu] = h$ for $\nu = k-2, ..., k+2$ then

$$H(\tau) = h^4\left((2-\tau)(1+\tau)\tau(1-\tau) + \frac{2}{3}\left(\tau^3 + (1-\tau)^3\right)\right) \leq \frac{35}{48}\,h^4,$$

which implies (14). In this case, when $\tau = 1/2$, the function $H(1/2) = 35/48\,h^4$. Therefore, 35/1152 is the least possible constant in Eq. 14. □

Proof of Theorem 3.15 When $t \in [t[1],\ t[2]]\,,\ \ t = t[1] + h[1]\tau,\ \ \tau \in [0,\ 1]$, the remainder term is

$$\begin{aligned}\rho(t) &:= f(t) - \bar{s}[f](t) = f(t) - P^3(t)[1] - A_0[f]\,(t - t[1])^3 \\ &= f[0,\ 1,\ 2,\ 3,\ t](t - t[0])(t - t[1])(t - t[2])(t - t[3]) \\ &\quad + f[0,\ 1,\ 2,\ 3,\ 4]\,\frac{(h[2])^2\,(t[4] - t[0])}{3\,h[1]\,(t[3] - t[1])}\,(t - t[1])^3.\end{aligned}$$

On the interval $[t[1],\ t[2]]$, both coefficients of the differences are positive and from the IVT we have

$$\begin{aligned}\rho(t) &= f[0,\ 1,\ 2,\ 3,\ \xi]\,H(\tau), \quad \xi \in I[0], \\ H(\tau) &:= (h[1])^2\tau \\ &\quad \left((h[0] + h[1]\tau)(1-\tau)\,(h[2] + h[1](1 - \tau)) + \frac{(h[2])^2\,(t[4] - t[0])}{3\,(h[2] + h[1])}\,\tau^2\right) \\ &\le \bar{h}^4\,\tau\,\left(3\tau^3 - 4\tau^2 - 3\tau + 6\right) \le \bar{h}^4\,\frac{16 - 3\sqrt{2}}{12\sqrt{2}},\end{aligned}$$

where $\bar{h} := \max_{k=0,1,2,3} h[k]$. Hence, the estimation in Eq. 22 follows. Equation 23 follows from Eq. 3.

If $h[0] = h[1] = h[2] = h[3] = h$ then we have the sharp estimation in Eq. 24, which becomes an identity for the function $f(t) = t^4$.

When $t \in [t[0],\ t[1]],\ \ t = t[0] + h[0]\tau,\ \ \tau \in [0,\ 1]$, the remainder term is

$$\begin{aligned}\rho_0(t) &:= f(t) - \bar{s}[f](t) = f(t) - P^3(t)[1] \\ &= -f[0,\ 1,\ 2,\ 3,\ t](h[0])^2\tau(1 - \tau)(h[0](1 - \tau) + h[1]) \\ &\quad \times(h[0](1 - \tau) + h[1] + h[2]).\end{aligned}$$

As above, $\bar{h} = \max_{k=0,1,2} h[k]$. Then, the estimation holds

$$|\rho(t)| \le \frac{\bar{h}^4}{24}\max_{t\in I[0]}|f^{(4)}(t)|\,\tau\,\left|\tau^3 - 6\tau^2 + 11\tau - 6\right| \le \frac{\bar{h}^4}{24}\max_{t\in I[0]}|f^{(4)}(t)|.$$

The estimation is sharp even for the uniform grid. It becomes an identity for the function $f(t) = t^4$. □

References

1. Abramowitz, M., Stegun, I.A.: Handbook of mathematical functions with formulas, graphs, and mathematical tables. Dover, New York (1972)
2. Aldroubi, A., Cabrellib, C., Molterb, U.M.: Wavelets on irregular grids with arbitrary dilation matrices and frame atoms for $l^2(\mathbb{R}^d)$. Applied and Comp Harm. Analysis **17**(2), 119–140 (2004)
3. Averbuch, A.Z., Neittaanmäki, P., Zheludev, V.A.: Spline and spline wavelet methods with applications to signal and image processing, Volume I: Periodic splines. Springer (2014)
4. Averbuch, A.Z., Neittaanmäki, P., Zheludev, V.A.: Spline and spline wavelet methods with applications to signal and image processing, Volume II: Non-periodic splines. Springer (2015)
5. Brislawn, C.M.: Classification of nonexpansive symmetric extension transforms for multirate filter banks. Appl. Comput. Harmon. Anal. **3**(4), 337–357 (1996)

6. Cai, T., Brown, L.: Wavelet shrinkage for nonequispaced samples. Annals of Statistics **26**(5), 1783–1799 (1998)
7. Daubechies, I., Guskov, I., Schroder, P., Sweldens, W.: Wavelets on irregular point sets. Phil. Trans. R. Soc. Lond. A **357**, 2397–2413 (1999)
8. de Boor, C.: A practical guide to splines. Springer, New York (1978)
9. de Boor, C.: Divided differences. Surveys in Approximation Theory 1 (2005)
10. Hall, C.A.: On error bounds for spline interpolation. J. Approx. Theory **1**, 209–218 (1968)
11. Dahmen, W., Carnicer, J.M., Pen, a., J.M.: Local decomposition of renable spaces and wavelets. Appl. Comput Harmon. Anal **3** (1996)
12. Liu, Z., Mi, Y., Mao, Y.: An improved real-time denoising method based on lifting wavelet transform. Measurement Science Review **14**(3), 152–159 (2014)
13. Lyche, T., Schumaker, L.L.: Local Spline Approximation Methods. MRC Technical Summary Mathematics Research Center University of Wisconsin (1974)
14. Schoenberg, I.J.: Contributions to the problem of approximation of equidistant data by analytic functions, vol. 4:45–99 (1946). Parts A and B
15. Schumaker, L.L.: Spline functions: Basic theory. John Wiley & Sons, New York (1981)
16. Stoer, J., Bulirsch, R.: Introduction to numerical analysis, Second edition. Springer–Verlag, New York (1993)
17. Suturin, M.G., Zheludev, V.: On the approximation on finite intervals and local spline extrapolation . Russian J. Numer. Anal. Math. Modelling **9**(1), 75–89 (1994)
18. Sweldens, W.: The lifting scheme: A custom-design construction of biorthogonal wavelets. Appl. Comput. Harmon. Anal. **3**(2), 186–200 (1996)
19. Sweldens, W.: The lifting scheme: A construction of second generation wavelets. SIAM. J. Math. Anal. **29**(2), 511–546 (1997)
20. Vanraesa, E., Jansenb, M., Bultheela, A.: Stabilised wavelet transforms for non-equispaced data smoothing. Signal Processing **82**(12), 1979–1990 (2002)
21. Xia, R., Meng, K., Qian, F., Wang, Z.L.: Online wavelet denoising via a moving window. Acta Automatica Sinica **33**(9), 897–901 (2007)
22. Zav'yalov, Y.S., Kvasov, B.I., Miroshnichenko, V.L.: Methods of spline-functions. Nauka, Moscow, 1980. (In Russian)
23. Zheludev, V.: Local spline approximation on arbitrary meshes. Sov. Math. **8**, 16–21 (1987)