Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations

Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick, *Member*, *IEEE*

Abstract—The paper presents SwiftSeg, a novel technique for online time series segmentation and piecewise polynomial representation. The segmentation approach is based on a least-squares approximation of time series in sliding and/or growing time windows utilizing a basis of orthogonal polynomials. This allows the definition of fast update steps for the approximating polynomial, where the computational effort depends only on the degree of the approximating polynomial and not on the length of the time window. The coefficients of the orthogonal expansion of the approximating polynomial—obtained by means of the update steps—can be interpreted as optimal (in the least-squares sense) estimators for average, slope, curvature, change of curvature, etc., of the signal in the time window considered. These coefficients, as well as the approximation error, may be used in a very intuitive way to define segmentation criteria. The properties of SwiftSeg are evaluated by means of some artificial and real benchmark time series. It is compared to three different offline and online techniques to assess its accuracy and runtime. It is shown that SwiftSeg—which is suitable for many data streaming applications—offers high accuracy at very low computational costs.

Index Terms—Time series, orthogonal polynomials, least-squares approximation, online segmentation, piecewise polynomial representation, SwiftSeg.

1 Introduction

Time series segmentation is an important issue not only in signal processing but also in data mining. Segmentation algorithms work either in a batch mode (offline) or online. The latter are needed for applications with harsh timing constraints (e.g., monitoring tasks), but they are also suited for applications where a huge amount of data must be processed.

A large class of segmentation algorithms not only determines *segment boundaries*—also referred to as *segmentation points*—but also a *piecewise linear representation* of the signal within a segment (typically, by linear interpolation or linear regression). As Keogh et al. pointed out in their very comprehensive review of such segmentation algorithms [1], [2], a piecewise linear representation—which is the most frequently used representation—can be used in many ways, e.g., to support

- fast similarity search,
- novel distance measures for time series,
- new clustering and classification algorithms, and
- change point detection.

According to [1], [2], most time series segmentation algorithms can be grouped into one of the following three categories:

- *sliding window* (a segment is grown until some error criterion is met),
- *top-down* (a time series is recursively partitioned until some error criterion is met), and
- *bottom-up* (starting with short segments that are merged until some error criterion is met).

Error criteria are often based on the least-squares error.

Sliding window algorithms—which are suitable for online applications—are very fast, but often lead to poor results. On the other hand, top-down and, in particular, bottom-up algorithms outperform sliding window techniques in terms of accuracy, but they typically cannot be used for online applications. As a consequence, a new algorithm, sliding window and bottom-up (SWAB), was introduced in [1], [2] in order to combine the advantages of the respective techniques.

In this paper, we describe a new approach to online segmentation of time series which we call *SwiftSeg*. The segmentation leads to a *piecewise polynomial representation* of segments with a polynomial of arbitrary degree (not just straight lines as in the case of a linear representation). SwiftSeg is a sliding window technique with very low runtime (lower than the combination of sliding window and bottom-up, of course), but its accuracy comes quite close to that of computationally much more expensive techniques. The key concept that allows this desirable property is a least-squares approximation of the time series in sliding and/or growing time windows by means of bases of orthogonal polynomials. The approximation—which is conducted by means of very fast update algorithms—yields optimal (in the least-squares sense) estimators of average, slope, curvature, change of curvature, etc., of the time series (signal) in the time windows. This information allows us to define segmentation criteria in a very intuitive way.

The remainder of the paper is structured as follows: Section 2 presents some related work in the field of online segmentation. In Section 3, we make some remarks on the solution of linear least-squares problems first. Then, we

- *E. Fuchs and J. Nitschke are with the Faculty of Computer Science and Mathematics, University of Passau, 94030 Passau, Germany. E-mail: {fuchse, nitschke}@fim.uni-passau.de.*
- *T. Gruber and B. Sick are with the Computationally Intelligent Systems Lab, University of Applied Sciences, Deggendorf, Germany. E-mail: {thiemo.gruber, bernhard.sick}@ft-deggendorf.de.*

Manuscript received 15 June 2008; revised 17 Apr. 2009; accepted 4 Oct. 2009; published online 25 Feb. 2010.
Recommended for acceptance by A. Srivastava.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-06-0358.
Digital Object Identifier no. 10.1109/TPAMI.2010.44.

present two techniques for a polynomial approximation of a time series in either sliding or growing windows. Finally, it is shown how the coefficients of the orthogonal expansion of the approximating polynomial can be used to define segmentation criteria. Section 4 presents some experimental results on artificial as well as real-world data. The properties—in particular, accuracy and runtime—of our approach are evaluated in detail. Finally, Section 5 summarizes the major findings.

2 Related Work

Algorithms for time series segmentation can be found in a wide range of applications, for example, medical and biological diagnostics, analysis of financial time series, speech processing, or sensor analysis. Here, we will focus on methods intended for use in an online mode or with any real-time conditions.

In [1], an overview and comparison of several existing methods is given and a new online algorithm, called SWAB, is introduced (see [2], too). A commonly used method for representing and segmenting time series is piecewise linear approximation, used, for example, in [3] in conjunction with time warping to segment electrocardiogram (ECG) signals. A fast algorithm for piecewise linear time series segmentation is proposed in [4], the so-called feasible span window and the stepwise feasible span window methods are described in [5], and a quasi-monotonic segmentation technique is suggested in [6]. Hidden Markov models (HMM) are also often used to represent different states and changes in signals and to detect segment boundaries (see, e.g., [7], [8]). In [9], a time series is segmented by measuring changes in Gaussian probability density functions that represent time-delay embedded data points within a sliding window. Also, fuzzy-clustering [10] and generalized Eigen-decomposition [11] are used to segment nonstationary or chaotic time series. Autoregressive moving average (ARMA) models in combination with a polynomial regression are used in [12].

To evaluate the runtime and the accuracy of SwiftSeg, we compare it to three other segmentation approaches:

1. *SW-SEG* is an online sliding window technique which is chosen to assess the runtime of SwiftSeg with respect to a fast segmentation technique.
2. *TD-SEG* is an offline top-down technique which is used to compare SwiftSeg to an algorithm yielding an optimal segmentation solution in terms of accuracy.
3. *SWBU-SEG* is an online segmentation technique based on a combination of sliding window and bottom-up which is selected to compare SwiftSeg to a segmentation algorithm that aims at offering a good compromise between accuracy and runtime.

SW-SEG is very closely related to SwiftSeg, as our new technique essentially is a sliding window approach.

TD-SEG is based on the segmentation approach described in [13], which uses dynamic programming to accelerate the search for an optimal segmentation. Given an error criterion and a number of segments, this technique considers all possible solutions to find the optimum. It is obvious that this approach cannot be used for online segmentation.

SWBU-SEG is a particular realization of SWAB [1] (see above) which is chosen for three reasons:

1. It is an online algorithm that leads to a piecewise, though linear, representation of segments.
2. It outperforms other techniques that also lead to a piecewise linear representation regarding accuracy.
3. It is already used in a number of applications, including biomedical signal monitoring [14], stock tracking and forecasting [15], and gesture spotting with body-worn inertial sensors [16].

Essentially, SWAB combines the ideas of sliding window and bottom-up in the following way [1]:

1. Samples of the time series are collected that correspond to a single segment using the (relatively fast) sliding window approach. The samples of the segment are added to a buffer.
2. As the samples move through the buffer, the (relatively accurate) bottom-up approach is given the chance to refine the segmentation as it has a "more global" view of the time series.

3 SwiftSeg: A Novel Technique for Online Time Series Segmentation

In this section, we briefly repeat some known, yet interesting facts concerning linear least-squares problems before we present two techniques for polynomial approximations in sliding and growing time windows, respectively. Then, we outline how this approximation can be exploited for time series segmentation.

3.1 Solution of Linear Least-Squares Problems

We are given a time series (signal) consisting of real-valued samples y_n with $n = 0, \ldots, N$ which are assumed to be measured at equidistant points in time x_n. This time series shall be modeled by a parameterized function $f(x) : \mathbb{R} \to \mathbb{R}$. Here, we assume that $f(x)$ is linearly dependent on a parameter vector $\mathbf{w}$ with elements $w_k \in \mathbb{R}$ $(k = 0, \ldots, K)$. Note that we do not claim that $f(x)$ is a linear function in x. More concretely, we assume that f is a linear combination of $K+1$ (linear or nonlinear) so-called basis functions f_k:

$$f(x) = \sum_{k=0}^{K} w_k \cdot f_k(x). \tag{1}$$

These basis functions may be polynomials, wavelets, sigmoid functions, or sinusoidal functions, for instance.

We may write the values of the $K+1$ basis functions for the $N+1$ points in time $x_0, \ldots, x_N$ into a matrix

$$\mathbf{F} \in \mathbb{R}^{(N+1)\times(K+1)} :$$

$$\mathbf{F} = \begin{pmatrix} f_0(x_0) & \ldots & f_K(x_0) \\ \vdots & \ddots & \vdots \\ f_0(x_N) & \ldots & f_K(x_N) \end{pmatrix}. \tag{2}$$

If we combine the $N+1$ samples of the overall time series into a vector $\mathbf{y}$ with elements y_n, the linear least-squares problem we want to solve can be denoted by

$$\min_{\mathbf{w}} \|\mathbf{F}\mathbf{w} - \mathbf{y}\|^2, \tag{3}$$

with $\|\cdot\|$ being the euclidean norm. Its solution $\mathbf{w}_{\mathrm{LS}}$ can be found by setting the derivative with respect to $\mathbf{w}$ to zero. First,

$$\begin{aligned}\|\mathbf{F}\mathbf{w}-\mathbf{y}\|^2 &= \langle \mathbf{F}\mathbf{w}-\mathbf{y}|\mathbf{F}\mathbf{w}-\mathbf{y}\rangle \\ &= \mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{w}-2\mathbf{y}^{\mathrm{T}}\mathbf{F}\mathbf{w}+\mathbf{y}^{\mathrm{T}}\mathbf{y},\end{aligned} \tag{4}$$

with $\langle\cdot|\cdot\rangle$ being the standard inner product in a real-valued vector space. Then,

$$\frac{\partial\|\mathbf{F}\mathbf{w}-\mathbf{y}\|^2}{\partial\mathbf{w}} = 2\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{w}-2\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5}$$

leads to the least-squares solution

$$\mathbf{w}_{\mathrm{LS}} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{6}$$

provided that the matrix $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ is regular, which we assume in the following [17], [18].

The matrix $\mathbf{F}^{+} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}$ is called the pseudo-inverse (or Moore-Penrose pseudo-inverse) of $\mathbf{F}$, that is

$$\mathbf{w}_{\mathrm{LS}} = \mathbf{F}^{+}\mathbf{y}. \tag{7}$$

In general, a real-valued pseudo-inverse $\mathbf{A}^{+}$ of a matrix $\mathbf{A}$ has the following two properties (two of the four so-called Penrose criteria). First, $(\mathbf{A}\mathbf{A}^{+})^{\mathrm{T}} = \mathbf{A}\mathbf{A}^{+}$. Second, $\mathbf{A}\mathbf{A}^{+}\mathbf{A} = \mathbf{A}$ and, consequently, $\mathbf{A}\mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} = \mathbf{A}\mathbf{A}^{+}$. Thus, the *residuum* resulting from this least-squares approximation is

$$\begin{aligned}r_{\mathrm{LS}} &= \|\mathbf{F}\mathbf{w}_{\mathrm{LS}}-\mathbf{y}\|^2 \\ &= \mathbf{y}^{\mathrm{T}}\underbrace{(\mathbf{F}\mathbf{F}^{+})^{\mathrm{T}}\mathbf{F}\mathbf{F}^{+}}_{\mathbf{F}\mathbf{F}^{+}}\mathbf{y}-2\mathbf{y}^{\mathrm{T}}\mathbf{F}\mathbf{F}^{+}\mathbf{y}+\mathbf{y}^{\mathrm{T}}\mathbf{y} \\ &= \mathbf{y}^{\mathrm{T}}\mathbf{y}-\mathbf{w}_{\mathrm{LS}}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{w}_{\mathrm{LS}},\end{aligned} \tag{8}$$

where we used $\mathbf{F}^{\mathrm{T}}\mathbf{y} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})\mathbf{F}^{+}\mathbf{y} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})\mathbf{w}_{\mathrm{LS}}$.

With the term (average) *squared error*, we refer to the residuum divided by the number of observed samples:

$$\sigma_{\mathrm{LS}}^2 = \frac{1}{N+1}\left(\mathbf{y}^{\mathrm{T}}\mathbf{y}-\mathbf{w}_{\mathrm{LS}}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{w}_{\mathrm{LS}}\right). \tag{9}$$

With (9), we could now determine the squared error once we have gotten the least-squares solution for $\mathbf{w}$.

In general, the solution of a linear least-squares problem is found by conducting a QR decomposition or a singular value decomposition (SVD) of $\mathbf{F}$ [17], [18].

Now, assume that the selected $K+1$ basis functions are orthogonal with respect to an inner product yielding the value $\sum_{n=0}^{N} f_{k_1}(x_n)\cdot f_{k_2}(x_n) = 0$ for any two basis functions f_{k_1} and f_{k_2} with $k_1 \neq k_2$. This is the case for special kinds of polynomials (see Section 3.2), for wavelet families, or the sinusoidal functions used for discrete Fourier transforms, for instance. Then,

$$\begin{aligned}&\mathbf{F}^{\mathrm{T}}\mathbf{F} \\ &= \begin{pmatrix} f_0(x_0) & \dots & f_0(x_N) \\ \vdots & \ddots & \vdots \\ f_K(x_0) & \dots & f_K(x_N)\end{pmatrix}\begin{pmatrix} f_0(x_0) & \dots & f_K(x_0) \\ \vdots & \ddots & \vdots \\ f_0(x_N) & \dots & f_K(x_N)\end{pmatrix} \\ &= \begin{pmatrix} \|f_0\|^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \|f_K\|^2\end{pmatrix}.\end{aligned} \tag{10}$$

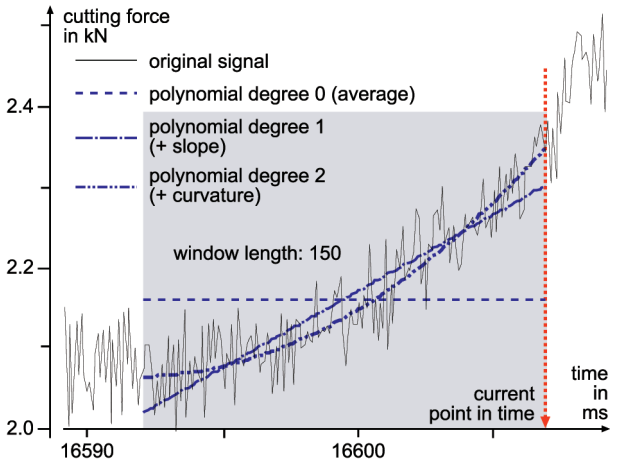

Fig. 1. Example for a polynomial approximation.

That is, $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ is a diagonal matrix which can be inverted if the elements in the diagonal, which are the squared norms of the basis functions, are nonzero. This can be easily guaranteed by an appropriate choice of basis functions. From (6), we then get

$$\begin{aligned}\mathbf{w}_{\mathrm{LS}} &= (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \\ &= \begin{pmatrix} f_0(x_0)\frac{1}{\|f_0\|^2} & \dots & f_0(x_N)\frac{1}{\|f_0\|^2} \\ \vdots & \ddots & \vdots \\ f_K(x_0)\frac{1}{\|f_K\|^2} & \dots & f_K(x_N)\frac{1}{\|f_K\|^2}\end{pmatrix}\begin{pmatrix} y_0 \\ \vdots \\ y_N\end{pmatrix} \\ &= \begin{pmatrix} \sum_{n=0}^{N}\frac{y_n}{\|f_0\|^2}f_0(x_n) \\ \vdots \\ \sum_{n=0}^{N}\frac{y_n}{\|f_K\|^2}f_K(x_n)\end{pmatrix}.\end{aligned} \tag{11}$$

That is, the least-squares solution can be written as a linear combination of the training samples (cf. the dual representations of classifiers which are common in the field of support vector machines, for instance).

With this result for $\mathbf{w}_{\mathrm{LS}}$, with (9), and with the definition $w_k = \sum_{n=0}^{N}\frac{y_n}{\|f_k\|^2}f_k(x_n)$ for $k = 0,\dots,K$ (elements of the solution vector $\mathbf{w}_{\mathrm{LS}}$), the squared error σ_{LS}^2 now becomes

$$\sigma_{\mathrm{LS}}^2 = \frac{1}{N+1}\left(\sum_{n=0}^{N} y_n^2 - \sum_{k=0}^{K} w_k^2\|f_k\|^2\right). \tag{12}$$

Note the similarity to Parseval's equality, in particular for the case of orthonormal basis functions, when the norms are 1.

3.2 Least-Squares Approximation with Orthogonal Polynomials

We will now consider a concrete realization of the time series model f, which will here be a polynomial p of a given degree K. As an example, Fig. 1 shows polynomials of degrees zero, one, and two approximating 150 values of a cutting force signal measured in an NC-lathe (taken from [19]).

We claim that the basis functions (polynomials) p_k with $k = 0,\dots,K$ used for the approximation must have the following properties:

- They must have different ascending degrees $0,\dots,K$.
- They must have a leading coefficient 1.

- They must be orthogonal with respect to the inner product

$$\langle p_{k_1} | p_{k_2} \rangle = \sum_{n=0}^{N} p_{k_1}(x_n) p_{k_2}(x_n) \tag{13}$$

for $k_1 \neq k_2$ in the time window considered.

Then, the approximating polynomial p is

$$p(x) = \sum_{k=0}^{K} \frac{\alpha_k}{\|p_k\|^2} p_k(x), \tag{14}$$

which is known as the *orthogonal expansion of the approximating polynomial*. If we compare this representation of the approximating function to (1), we state that the coefficients $\frac{\alpha_0}{\|p_0\|^2}, \frac{\alpha_1}{\|p_1\|^2}, \frac{\alpha_2}{\|p_2\|^2}, \ldots$ correspond to the weights w_k.

The required properties lead to the following advantages:

- The coefficients

$$\frac{\alpha_0}{\|p_0\|^2}, \frac{\alpha_1}{\|p_1\|^2}, \frac{\alpha_2}{\|p_2\|^2}, \ldots$$

can be directly interpreted as optimal (in the least-squares sense) estimators for average, slope (increase/decrease), curvature, and so on, of the time series in the time window which is considered. This allows an intuitive interpretation of the approximation result in particular, as these parameters are independent in the sense of [20], [21].
- The coefficients

$$\frac{\alpha_0}{\|p_0\|^2}, \frac{\alpha_1}{\|p_1\|^2}, \frac{\alpha_2}{\|p_2\|^2}, \ldots$$

can be computed with extremely fast update algorithms. That is, given an approximation for the points in time $x_0, \ldots, x_N$, we get an approximation—under certain conditions which will be detailed in the following sections—for either $x_1, \ldots, x_{N+1}$ (sliding window) or $x_0, \ldots, x_N, x_{N+1}$ (growing window) with a computational effort independent of N.

We present two solutions, one for the sliding window (SW) case and one for the growing window (GW) case. As the approximation is typically done for only a part of the original time series (x_n, y_n) with $n = 0, \ldots, N$, we partly take over the notation from the respective original publications.

3.2.1 Solution for the Sliding Window Approximation (SW Technique)

We provided a solution for the SW case in [22], [23], [24]. In contrast to the solution for the GW case (see below), it requires that the points in time for which values are available are equidistant.

Assume that, in a time window of length $M+1$, the values $y_0, y_1, \ldots, y_M$ measured at equidistant points in time $x_0, x_1, \ldots, x_M$ must be approximated by a polynomial p with degree $K \leq M$ ($M \in \mathbb{N}$, $K \in \mathbb{N}_0$) in the least-squares sense.

It is well known that orthogonal polynomials with leading coefficient 1 in the vector space $\mathcal{P}(\mathbb{R}, \mathbb{R})$ of real polynomials on $\mathbb{R}$ fulfill—with certain restrictions concerning an inner product on $\mathcal{P}(\mathbb{R}, \mathbb{R})$ (see [22], for details)—the following three-term recurrence relation:

$$p_{-1}(x) = 0, \tag{15}$$

$$p_0(x) = 1, \tag{16}$$

$$p_{k+1}(x) = (x - a_k) p_k(x) - b_k p_{k-1}(x). \tag{17}$$

The coefficients $a_k, b_k \in \mathbb{R}$ are given by

$$a_k = \frac{\langle \mathrm{id} p_k | p_k \rangle}{\langle p_k | p_k \rangle} \text{ for } k \in \mathbb{N}_0, \tag{18}$$

$$b_k = \frac{\langle p_k | p_k \rangle}{\langle p_{k-1} | p_{k-1} \rangle} \text{ for } k \in \mathbb{N}, \tag{19}$$

with the function id being the identity in the vector space $\mathcal{P}(\mathbb{R}, \mathbb{R})$. For polynomials orthogonal with respect to the discrete inner product, defined in (13), with degrees $k < M$, this three-term recurrence relation is applicable.

An important advantage of the three-term recurrence relation is that there exists an efficient algorithm (Clenshaw algorithm) for the evaluation of the approximating polynomial.

There are various systems of discrete orthogonal polynomials which can be used for the approximation [25]. One example are the *discrete Chebyshev polynomials*. For $0 \leq k_1$, $k_2 \leq M$, and $k_1 \neq k_2$ these polynomials are orthogonal with respect to the inner product

$$\langle p_{k_1} | p_{k_2} \rangle = \sum_{j=0}^{M} p_{k_1}(j) p_{k_2}(j). \tag{20}$$

Note that the approximation window $[x_0, x_1, \ldots, x_M]$ must be located at $[0, 1, \ldots, M]$ (cf. [22] and (13)). Thus, we have to change the viewpoint from a window which is sliding over the time series to a time series sliding under a (fixed) window.

As a special case of the so-called Hahn polynomials, the discrete Chebyshev polynomials fulfill the three-term recurrence relation with

$$a_k = \frac{M}{2}, \tag{21}$$

$$b_k = \frac{k^2((M+1)^2 - k^2)}{4(4k^2 - 1)}. \tag{22}$$

For a degree up to 4, these polynomials are set out in Table 1. Their squared norms are given by:

$$\|p_k\|^2 = \frac{(k\,!)^4}{(2\,k)\,!\,(2\,k+1)\,!} \cdot \prod_{i=-k}^{k} (M+1+i) \tag{23}$$

for $k = 0, \ldots, M+1$.

The approximation problem

$$\text{“find a polynomial } p \text{ of degree } K \text{ which}$$
$$\text{minimizes } \sum_{j=0}^{M} (p(j) - y_j)^2 \text{”}$$

could now principally be solved using the orthogonal expansion given in (14), where the orthogonal expansion coefficients α_k are given by

TABLE 1
Orthogonal Polynomials in the Sliding Window Case

$$
\begin{aligned}
p_0(x) &= 1\\
p_1(x) &= x - \frac{M}{2}\\
p_2(x) &= x^2 - Mx + \frac{M^2 - M}{6}\\
p_3(x) &= x^3 - \frac{3M}{2}x^2 + \frac{6M^2 - 3M + 2}{10}x - \frac{M(M-1)(M-2)}{20}\\
p_4(x) &= x^4 - 2Mx^3 + \frac{9M^2 - 3M + 5}{7}x^2 - \frac{2M^3 - 3M^2 + 5M}{7}x + \frac{M(M-1)(M-2)(M-3)}{70}
\end{aligned}
$$

$$\alpha_k = \sum_{j=0}^{M} y_j \cdot p_k(j) \tag{24}$$

for $k = 0, \ldots, K$ (cf. (11)). For the discrete Chebyshev polynomials, however, we described fast and efficient update formulas for the computation of the orthogonal expansion coefficients, where the computational costs for updates are independent of the window length (see [22]). The update formulas for a degree up to 4 are given in Table 2. Here, $\alpha_{i,t}$ is the coefficient corresponding to the basis polynomial of degree i in the approximation of the values $y_{t-M}, \ldots, y_t$. The low complexity (see below) and the numerical stability of this update algorithm provide coefficients in real time, even if very high sampling rates are required. Note that all terms containing M can only be computed in advance. The same holds for the norms of the basis polynomials. It is important to state that it is not necessary to determine the polynomials p_k explicitly if only the coefficients α_k are needed (cf. Section 3.3).

How can the updating mechanism be initialized, avoiding the high computational cost of a least-squares approximation "from scratch"? Assuming, we are given $M+1$ samples which are all zero, i.e., $y_{t-M} = 0, \ldots, y_t = 0$, then all of the coefficients can be set to zero, too. After $M+1$ update steps with the actually observed values, the approximation gets automatically correct.

From (12) and with the update steps for the α_k, we obtain the following update mechanism for the squared error:

$$
\begin{aligned}
\sigma_{t+1}^2 &= \frac{1}{M+1}\left(\sum_{j=0}^{M} y_{t+1-M+j}^2 - \sum_{k=0}^{K} \frac{\alpha_{k,t+1}^2}{\|p_k\|^2}\right)\\
&= \frac{1}{M+1}\left(\sum_{j=0}^{M} y_{t-M+j}^2 + y_{t+1}^2 - y_{t-M}^2 - \sum_{k=0}^{K} \frac{\alpha_{k,t+1}^2}{\|p_k\|^2}\right).
\end{aligned}
\tag{25}
$$

That is, the sum of the squared values of the time series can be updated quite easily and K steps are necessary to sum up the squared α_k divided by the squared norms.

In [19], [22], it is shown how additional weight functions could be considered which individually weight each value in the time window (e.g., with linear or exponential weight functions).

3.2.2 Solution for the Growing Window Approximation (GW Technique)

A solution for the GW case is provided by Elhay et al. in [26]. In contrast to the solution for the SW case (see above), it does not require that the points in time for which values are available be equidistant. Moreover, a new value could also be observed at a point in time within the already considered time window, but we do not use the two properties here.

It must be stated that in the GW case, the basis polynomials do not stay constant but must be computed anew in each time step, i.e., for every observed sample. The basis polynomials for the GW technique are orthonormal, i.e., the norms are 1. Thus, it cannot be guaranteed that the leading coefficient is 1, as required. However, as we will

TABLE 2
Formulas for the Fast Update of the Orthogonal Expansion Coefficients in the Sliding Window Case

$$
\begin{aligned}
\alpha_{0,t+1} &= \alpha_{0,t} + y_{t+M+1} - y_t\\
\alpha_{1,t+1} &= \alpha_{1,t} + \left(1 + \frac{M}{2}\right)\cdot y_{t+M+1} + \frac{M}{2}\cdot y_t - \alpha_{0,t+1}\\
\alpha_{2,t+1} &= \alpha_{2,t} + \frac{M^2 + 5\,M + 6}{6}\cdot y_{t+M+1} - \frac{M^2 - M}{6}\cdot y_t - 2\,\alpha_{1,t+1} - \alpha_{0,t+1}\\
\alpha_{3,t+1} &= \alpha_{3,t} + \frac{M^3 + 9\,M^2 + 26\,M + 24}{20}\cdot y_{t+M+1} + \frac{M^3 - 3\,M^2 + 2\,M}{20}\cdot y_t - 3\,\alpha_{2,t+1} - 3\,\alpha_{1,t+1} - \frac{M^2 + 2\,M + 12}{10}\cdot \alpha_{0,t+1}\\
\alpha_{4,t+1} &= \alpha_{4,t} + \frac{(M+5)!}{70(M+1)!}\cdot y_{t+M+1} - \frac{M!}{70(M-4)!}\cdot y_t - 4\,\alpha_{3,t+1} - 6\,\alpha_{2,t+1} - \frac{6M^2 + 12M + 162}{35}\cdot \alpha_{1,t+1}\\
&\quad - \frac{2M^2 + 4M + 12}{7}\cdot \alpha_{0,t+1}
\end{aligned}
$$

TABLE 3
Orthonormal Polynomials in the Growing Window Case

$$
\begin{aligned}
q_0(x) &= \frac{1}{\sqrt{m+1}} \\
q_1(x) &= \frac{1}{\sqrt{m+1}\delta_0}x - \frac{\gamma_0}{\sqrt{m+1}\delta_0} \\
q_2(x) &= \frac{1}{\sqrt{m+1}\delta_0\delta_1}x^2 - \frac{\gamma_0+\gamma_1}{\sqrt{m+1}\delta_0\delta_1}x + \frac{\gamma_0\gamma_1-\delta_0^2}{\sqrt{m+1}\delta_0\delta_1} \\
q_3(x) &= \frac{1}{\sqrt{m+1}\delta_0\delta_1\delta_2}x^3 - \frac{\gamma_0+\gamma_1+\gamma_2}{\sqrt{m+1}\delta_0\delta_1\delta_2}x^2 + \frac{\gamma_0\gamma_1+\gamma_0\gamma_2+\gamma_1\gamma_2-\delta_0^2-\delta_1^2}{\sqrt{m+1}\delta_0\delta_1\delta_2}x + \frac{\gamma_0\delta_1^2+\gamma_2\delta_0^2-\gamma_0\gamma_1\gamma_2}{\sqrt{m+1}\delta_0\delta_1\delta_2} \\
q_4(x) &= \frac{1}{\sqrt{m+1}\delta_0\delta_1\delta_2\delta_3}x^4 - \frac{\gamma_0+\gamma_1+\gamma_2+\gamma_3}{\sqrt{m+1}\delta_0\delta_1\delta_2\delta_3}x^3 + \frac{\gamma_0\gamma_1+\gamma_0\gamma_2+\gamma_0\gamma_3+\gamma_1\gamma_2+\gamma_1\gamma_3+\gamma_2\gamma_3-\delta_0^2-\delta_1^2-\delta_2^2}{\sqrt{m+1}\delta_0\delta_1\delta_2\delta_3}x^2 \\
&\quad + \frac{\gamma_0\delta_1^2+\gamma_0\delta_2^2+\gamma_1\delta_2^2+\gamma_2\delta_0^2+\gamma_3\delta_0^2+\gamma_3\delta_1^2-\gamma_0\gamma_1\gamma_2-\gamma_0\gamma_1\gamma_3-\gamma_0\gamma_2\gamma_3-\gamma_1\gamma_2\gamma_3}{\sqrt{m+1}\delta_0\delta_1\delta_2\delta_3}x \\
&\quad + \frac{\delta_0^2\delta_2^2-\gamma_0\gamma_1\delta_2^2-\gamma_0\gamma_3\delta_1^2-\gamma_2\gamma_3\delta_0^2+\gamma_0\gamma_1\gamma_2\gamma_3}{\sqrt{m+1}\delta_0\delta_1\delta_2\delta_3}
\end{aligned}
$$

see, the required solution (cf. (14)) can be easily obtained after an approximation step by rescaling the basis polynomials as well as the orthogonal expansion coefficients.

In the following, we will describe the construction of the GW solution, which mainly consists of two phases: an interpolation (initialization) phase and an approximation (updating) phase. For the first $K+1$ observed values, the size of the solution (which corresponds to the degree of the approximating polynomial) grows with each new data point, starting with the first point in time interpolated by a polynomial of degree zero. Once a required degree K is reached, the solution will be updated when a new data point (x_{t+1}, y_{t+1}) is added, but its size (i.e., the polynomial degree) is not further increased.

The basis polynomials q_k of exact degree $k = 0, \ldots, K$ (i.e., with nonzero leading coefficients) are defined by a constant lower Hessenberg matrix $\mathbf{J}_{K-1}$ and a scalar δ_{K-1}, where

$$x\mathbf{q}_{K-1}(x) = \mathbf{J}_{K-1}\mathbf{q}_{K-1}(x) + \delta_{K-1}q_K(x)\mathbf{e}_K \tag{26}$$

holds for any x with $\mathbf{q}_{K-1} = (q_0, q_1, \ldots, q_{K-1})^T$ (that is, $\mathbf{q}_{K-1}$ is the vector of all basis polynomials up to degree $K-1$) and $\mathbf{e}_K$ denoting the Kth unit vector with required dimensionality. For orthonormal polynomials, the so-called recurrence matrix $\mathbf{J}_{K-1}$ is symmetric and tridiagonal and given by its diagonal and offdiagonal elements $\gamma_0, \ldots, \gamma_{K-1}$ and $\delta_0, \ldots, \delta_{K-2}$, respectively:

$$\mathbf{J}_{K-1} = \begin{pmatrix} \gamma_0 & \delta_0 & 0 & \cdots & 0 \\ \delta_0 & \gamma_1 & \delta_1 & \cdots & 0 \\ 0 & \delta_1 & \gamma_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \gamma_{K-1} \end{pmatrix}. \tag{27}$$

$\mathbf{J}_{K-1}$ has real, distinct eigenvalues $\lambda_0, \ldots, \lambda_{K-1}$ that are roots of q_K with the corresponding eigenvectors $\mathbf{q}_{K-1}(\lambda_0), \ldots, \mathbf{q}_{K-1}(\lambda_{K-1})$. Therefore, q_K is the characteristic polynomial of $\mathbf{J}_{K-1}$. The update formulas for the diagonal and offdiagonal elements of the recurrence matrix will be given later.

Let $m+1$ be the number of samples observed so far, $\mathbf{P}_m = (\mathbf{q}_m(\lambda_0), \ldots, \mathbf{q}_m(\lambda_m))$, and $\mathbf{D}_m = \mathrm{diag}(\nu_0, \ldots, \nu_m)$ an $(m+1) \times (m+1)$ matrix with diagonal elements

$$\nu_j = \left\|\mathbf{q}_m(\lambda_j)\right\|, \quad j = 0, 1, \ldots, m. \tag{28}$$

Then, for any polynomial $q(x) = \sum_{k=0}^{m} \beta_k q_k(x)$ (cf. (14)) with polynomial coefficients $\boldsymbol{\beta}_m = (\beta_0, \ldots, \beta_m)$, the solution of the interpolation problem (phase 1) can be obtained by setting $\boldsymbol{\beta}_m = \mathbf{P}_m\mathbf{D}_m^2\mathbf{y}_{t,m+1}$, where $\mathbf{y}_{t,m+1} = (y_{t-m}, \ldots, y_t)^T$ are the values of the time series for the current problem. For $K < m$ (phase 2), this leads to the solution of the approximation problem $\boldsymbol{\beta}_K^* = (\beta_0, \ldots, \beta_K, 0, 0, \ldots, 0)^T$. The shortened vector $\boldsymbol{\beta}_K = (\beta_0, \ldots, \beta_K)^T$ is then given by

$$\boldsymbol{\beta}_K = \mathbf{P}_{K,m}\mathbf{D}_m^2\mathbf{y}_{t,m+1}, \tag{29}$$

where $\mathbf{P}_{K,m}$ is a matrix consisting of the first $K+1$ rows of $\mathbf{P}_m$. The orthonormal basis polynomials $\mathbf{q}_K$ can be obtained by evaluating the first $K+1$ rows of the following system of equations:

$$\begin{pmatrix} \mathbf{e}_1^T \\ (\mathbf{J}_K - x\mathbf{I}) \end{pmatrix}\mathbf{q}_K(x) = \begin{pmatrix} q_0(x) \\ -\delta_K q_{K+1}(x)\mathbf{e}_{K+1} \end{pmatrix}. \tag{30}$$

The resulting formulas for q_k with $k = 0, \ldots, 4$ are shown in Table 3. Given the vector $\boldsymbol{\beta}_K$ of polynomial coefficients, the approximating polynomial can be evaluated by a forward-substitution for $\mathbf{q}_K$ and an inner product with $\boldsymbol{\beta}_K$. Algorithm 1 shows how the approximating polynomial q can be evaluated at a point x.

Algorithm 1: Evaluation of the Approximating Polynomial q at a Point x.

Input: $\mathbf{J}_K$, $\boldsymbol{\beta}_K$, x
Output: $y = q(x)$

$q_0 = \frac{1}{\sqrt{m+1}}$
if $K > 0$ **then**
  $q_1 = q_0(x - \gamma_0)\frac{1}{\delta_0}$
  for $k = 2, \ldots, K$ **do**
    $q_k = (q_{k-1}(x - \gamma_{k-1}) - q_{k-2}\delta_{k-2})\frac{1}{\delta_{k-1}}$
$y = \mathbf{q}_K^T\boldsymbol{\beta}_K$

The construction of the least-squares approximation can now be done as follows: Starting with the initial case of the interpolation phase—a one-point set with (x_t, y_t)—given by

$$\mathbf{J}_{0,t} = (x_t) \quad \text{and} \quad \boldsymbol{\beta}_{0,t} = (y_t), \tag{31}$$

the entries of the recurrence matrix $\mathbf{J}_{k+1,t+1}$ and the vector of polynomial coefficients $\boldsymbol{\beta}_{k+1,t+1}$ are, for $k < K$ and a new data point (x_{t+1}, y_{t+1}), defined by

$$\mathbf{J}_{k+1,t+1} = \mathbf{Q}\begin{pmatrix} \mathbf{J}_{k,t} & \mathbf{0} \\ \mathbf{0}^T & x_{t+1} \end{pmatrix}\mathbf{Q}^T \tag{32}$$

and

$$\boldsymbol{\beta}_{k+1,t+1} = \mathbf{Q}\begin{pmatrix} \boldsymbol{\beta}_{k,t} \\ y_{t+1} \end{pmatrix}, \tag{33}$$

where $\mathbf{J}_{k,t}$ and $\boldsymbol{\beta}_{k,t}$ represent the current solution for the first $m+1$ samples. $\mathbf{Q}$ is the orthogonal similarity matrix determined by

$$\mathbf{Q}(\mathbf{e}_1\sqrt{m+1} + \mathbf{e}_{k+2}) = \mathbf{e}_1\sqrt{m+2} \tag{34}$$

and the fact that $\mathbf{J}_{k+1,t+1}$ again is required to be tridiagonal.

When the solution is built up to a given degree K and no further increase is required (i.e., $K \ll m$, for practical purposes), the updated solution in the approximation phase is defined by the $(K+1) \times (K+1)$ upper left submatrix of $\mathbf{J}_{K+1,t+1}$ and a vector comprising the first $K+1$ entries of $\boldsymbol{\beta}_{K+1,t+1}$. The values of $\mathbf{J}_{K+1,t+1}$ and $\boldsymbol{\beta}_{K+1,t+1}$ are for a new data point (x_{t+1}, y_{t+1}), again obtained by using (32) and (33), respectively. Algorithm 2 sketches the way for constructing and updating the solution.

Algorithm 2: Update of the Recurrence Matrix $\mathbf{J}_{k,t}$ and the Polynomial Coefficients $\boldsymbol{\beta}_{k,t}$ for a new Data Point (x_{t+1}, y_{t+1}).

Input: $\mathbf{J}_{k,t}$, $\boldsymbol{\beta}_{k,t}$, (x_{t+1}, y_{t+1})
Output: $\mathbf{J}_{k+1,t+1}$, $\boldsymbol{\beta}_{k+1,t+1}$

$c = \sqrt{\frac{m+1}{m+2}}$
$s = \frac{1}{\sqrt{m+2}}$
$\theta_1 = cs(x_{t+1} - \gamma_{0,t})$
$\theta_2 = -s\delta_{0,t}$
$\gamma_{k+1,t+1} = c^2 x_{t+1} + s^2\gamma_{0,t}$
$\gamma_{0,t+1} = c^2\gamma_{0,t} + s^2 x_{t+1}$
$\delta_{0,t+1} = c\delta_{0,t}$
$\beta_{k+1,t+1} = cy_t - s\beta_{0,t}$
$\beta_{0,t+1} = c\beta_{0,t} + sy_{t+1}$
for $j = 1, \ldots, k$ **do**
  $\zeta = \sqrt{\delta_{j-1,t+1}^2 + \theta_1^2}$
  $c = \frac{\delta_{j-1,t+1}}{\zeta}$
  $s = \frac{\theta_1}{\zeta}$
  $\theta_1 = cs(\gamma_{k+1,t+1} - \gamma_{j,t}) + \theta_2(c^2 - s^2)$
  $\gamma_{j,t+1} = c^2\gamma_{j,t} + 2cs\theta_2 + s^2\gamma_{k+1,t+1}$
  $\gamma_{k+1,t+1} = c^2\gamma_{k+1,t+1} - 2cs\theta_2 + s^2\gamma_{j,t}$
  $\delta_{j-1,t+1} = \zeta$
  $\delta_{j,t+1} = c\delta_{j,t}$
  $\beta_{j,t+1} = c\beta_{j,t} + s\beta_{k+1,t+1}$
  $\beta_{k+1,t+1} = c\beta_{k+1,t+1} - s\beta_{j,t}$
  $\theta_2 = -s\delta_{j,t}$
$\delta_{k,t+1} = |\theta_1|$
$\beta_{k+1,t+1} = \text{sign}(\theta_1)\beta_{k+1,t+1}$

Since the basis polynomials q_k are orthonormal (i.e., $\|q_k\| = 1$) and have leading coefficient $\frac{1}{\xi_k}$ with $\xi_k = \sqrt{m+1}\prod_{i=0}^{k-1}\delta_i$ (cf. Table 3), it follows that $\frac{\alpha_k}{\|p_k\|^2} = \frac{\beta_k}{\xi_k}$ for $\alpha_k = \beta_k\xi_k$ and $p_k = q_k\xi_k$. The optimal estimators $\frac{\alpha_k}{\|p_k\|^2}$ mentioned above are, thus, directly given by $\frac{\beta_k}{\xi_k}$.

Using (12) and the update steps for the values of $\boldsymbol{\beta}_{k,t}$, the least-squares error can easily be obtained with an update step which is similar to the SW case:

$$\sigma_{t+1}^2 = \frac{1}{m+2}\left(\sum_{j=0}^{m} y_{t-m+j}^2 + y_{t+1}^2 - \sum_{k=0}^{K}\beta_{k,t+1}^2\right). \tag{35}$$

In addition to the updating of the approximating polynomials as described above, a downdating could also be done with the techniques described in [26]. Thus, it would be possible to replace the SW technique by an appropriate extension of the GW technique (updating in combination with downdating). However, this technique would require higher runtimes for low polynomial degrees—which we need for online segmentation—as the basis polynomials must be updated in each step. The reason is that in the case of the SW technique, the window is fixed and the time series moves under this window. In the case of this extended GW technique, the window moves over a fixed time series where the x_t become larger. Moreover, the downdating may become numerically unstable even for quite short time series.

An additional weight of each value in the time window can be used to solve a weighted least-squares approximation problem (see [26], for details).

3.2.3 Some Remarks on Computational Complexity

If we have a time series of a certain length, say N, the computational costs of a least-squares approximation of the time series with a polynomial of degree K are $O(K^2 \cdot N)$ in a standard approach—based on SVD or QR decomposition—and $O(K \cdot N)$ if orthogonal basis polynomials are used (see also [17], [18], [22]).

If we have an online application and want to approximate the time series in either a sliding or a growing window of a certain length, say M, we have three alternatives: a standard approximation, an approximation based on orthogonal polynomials, or an update from an existing solution based on orthogonal polynomials as described in this paper. The first solution has computational costs of $O(K^2 \cdot M)$, the second costs $O(K \cdot M)$, and the third costs only $O(K^2)$ if the SW technique is used and $O(K)$ if the GW technique is taken. We want to emphasize that these are the costs for including a new (measured) sample in an approximating polynomial in an online application. To our knowledge, there is no online time series segmentation approach that is based on the third, most effective, solution, which is independent of M.

The complexity class is certainly interesting, but the actual runtimes are influenced by constant factors and additional terms (with lower degrees of influence) as well. The experiments in Section 4 will show that for low polynomial degrees—which we need for online segmentation here—the SW technique is even faster than the GW technique (see [22], [24], for details). The main reason has already been mentioned: In the GW case, the basis polynomials must be determined anew with each update step.

3.3 SwiftSeg: An Online Segmentation Framework

The two approximation techniques, SW and GW, now provide a lot of alternatives to define segmentation criteria. It is possible to use GW and SW either alone or in combination. If they are used in combination they could either be run such that the last data points of the two time windows (sliding and growing) are identical or they could be run with a certain temporal offset. Criteria could be based, for example,

- on the orthogonal expansion coefficients (e.g., absolute values or sign switches of slope or curvature),
- on the values of the residuum or the average least-squares error, or
- on the deviation of a predicted value (by means of a local model of the signal) from an actually measured value of the signal (e.g., weighted by means of the least-squares error).

These criteria could either be used alone or in combination. They could also be combined with "standard" criteria such as criteria considering the segment length, the number of zero crossings, or the signal values themselves. The form of the combination could be either a linear combination using some weights or a combination of several predicates (e.g., indicating whether a threshold is reached or not).

The various orthogonal expansion coefficients and the least-squares error are not only very descriptive, but also—due to the orthogonality of the basis polynomials—independent in the sense of [20], [21]. Thus, they can be used in a very intuitive way. If the GW technique is involved in the segmentation, it is not only possible to get segmentation points, but also to obtain a very compact, abstract representation of the segment (i.e., a *piecewise polynomial representation* of the signal). These local models of the time series—given by the orthogonal expansion coefficients and the squared error—could be used for further processing steps of the time series which can be used in various ways, as sketched in Section 1. It could also be exploited that—again due to the orthogonality—any solution with a certain degree K of the approximating polynomial includes any approximating polynomial of degree $0, 1, \ldots, K-1$ as well. This holds not only for the approximating polynomial itself, but also for the least-squares error, which could be decomposed accordingly.

Altogether, with our new approach, a powerful new framework for online segmentation—which we call SwiftSeg—becomes available which cannot be investigated in full detail in Section 4. Thus, we now formally define only a few criteria that are actually applied in the following section.

Assume that at a certain point in time x_t, we are given an approximating polynomial p obtained by the SW or the GW approximation in a sliding or growing window with length $L+1$ for the samples $y_{t-(L+1)}$ up to y_{t-1}. In addition, we know the number of the sign switches of the slope s_{t-1} within this window (we count them by observing $\alpha_{1,t-2} \cdot \alpha_{1,t-1}$). Then, a new sample y_t is received and the following criteria can be evaluated:

- sign switches of the slope with threshold $th_{\mathrm{SSS}} \in \mathbb{N}^+$

$$c_{\mathrm{SSS}} = \begin{cases} 1, & \text{if } s_{t-1} > th_{\mathrm{SSS}}, \\ 0, & \text{otherwise}, \end{cases} \tag{36}$$

- deviation of the predicted value to the actually measured value (unweighted) with threshold $th_{\mathrm{DPU}} \in \mathbb{R}^+$

$$c_{\mathrm{DPU}} = \begin{cases} 1, & \text{if } |p(x_t) - y_t| > th_{\mathrm{DPU}}, \\ 0, & \text{otherwise}, \end{cases} \tag{37}$$

- squared error with threshold $th_{\mathrm{SQE}} \in \mathbb{R}^+$

$$c_{\mathrm{SQE}} = \begin{cases} 1, & \text{if } \sigma_{t-1} > th_{\mathrm{SQE}}, \\ 0, & \text{otherwise}, \end{cases} \tag{38}$$

- residuum with threshold $th_{\mathrm{RES}} \in \mathbb{R}^+$

$$c_{\mathrm{RES}} = \begin{cases} 1, & \text{if } \sigma_{t-1} \cdot (L+1) > th_{\mathrm{RES}}, \\ 0, & \text{otherwise}, \end{cases} \tag{39}$$

- maximum with threshold $th_{\mathrm{MAX}} \in \mathbb{R}^-$

$$c_{\mathrm{MAX}} = \begin{cases} 1, & \text{if } \alpha_{1,t-2} \cdot \alpha_{1,t-1} < 0 \ \wedge \ \frac{\alpha_{2,t-1}}{\|p_2\|^2} < th_{\mathrm{MAX}}, \\ 0, & \text{otherwise}. \end{cases} \tag{40}$$

A segmentation point is set at time x_{t-1} if a threshold was reached (predicate yields value 1).

In most applications, a single criterion is sufficient to realize an appropriate segmentation technique, e.g., c_{MAX} that can be used to segment at each local maximum of the time series, or c_{RES} that is also applied by many other segmentation techniques.

In some of our experiments, we run the SW and the GW techniques in parallel such that the current point in time x_t corresponds to the end of the two time windows. Having set a segmentation point at x_{t-1}, we let the SW window slide further and we begin a new GW window starting at x_t. Segmentation criteria are typically evaluated for the SW technique. If a segmentation point is set at x_{t-1}, the GW technique provides us with a polynomial representation of the new segment of the time series, i.e., for the samples $y_{t-(L+1)}, \ldots, y_{t-1}$. This representation may be used in various ways, as set out in Section 1. For example, the similarity of two segments—even if they differ in length—may be numerically assessed by comparing their abstract descriptions (coefficients of the orthogonal expansion).

A local model of a time series may also be exploited to improve a segmentation result if characteristic segmentation points (e.g., minima or maxima) have to be found. If the need to set a segmentation point is stated, it is possible to set this point to an optimum of the local model (e.g., a polynomial of degree 2), which is determined analytically. This option is referred to as "jump to local optimum" in the following.

4 Experimental Evaluation

In this section, we will first investigate some properties of SwiftSeg and then compare it to some other segmentation techniques (cf. Section 2).

4.1 Some Basic Properties of SwiftSeg

First, we will set out the segmentation results for two artificial time series using either the GW or the SW

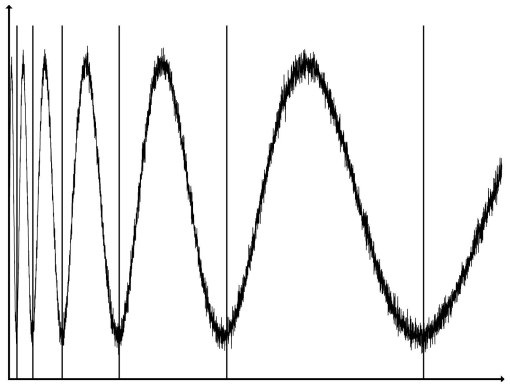

Fig. 2. Result of a segmentation of Sin1 with the GW technique: time series and segmentation points.

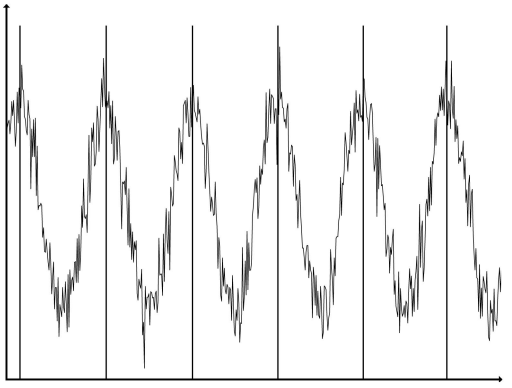

Fig. 3. Result of a segmentation of Sin2 with the SW technique: time series and segmentation points (detail).

approximation together with a single segmentation criterion. Each time series used in this section consists of a sinusoidal component and an added random component, i.e., white noise with zero mean $\mathcal{N}(0;\sigma^2)$.

The first time series Sin1 is generated using the sinusoidal function $\sin(10 \cdot \ln(x))$ and $\sigma = 0.05$. This time series will be segmented in the interval $[3.33, 155]$, where the range is $[-1, 1]$. It consists of 4,572 samples. We use the GW technique with a polynomial of degree 2. The segmentation criterion is c_{DPU} with a threshold of 0.85. The result is shown in Fig. 2. Here and in the following figures, the segmentation points (i.e., segment boundaries) are depicted with vertical lines. It can be stated that in this example, the time series is segmented close to the minima of the underlying sinusoidal function. This behavior is investigated in Table 4 in some more detail. The relative error refers to the length of the interval between the preceding and the current minimum.

Vice versa, we will now apply the SW technique with a single criterion to another time series. This second time series Sin2 is generated using the sinusoidal function $\sin^2(x)$ and $\sigma = 0.1$. Here, we segment within the interval $[0, 141]$, where the range is $[0, 1]$. This time series consists of 4,222 samples. The approximating polynomial has degree 2 and the length of the sliding window is 30. We use the segmentation criterion c_{SSS} with a threshold of 2. In this experiment, we use the option "jump to local optimum" described in Section 3.3. The result is shown in Fig. 3 (only in the interval $[0, 19.5]$). The maxima of the time series are nicely hit as set out in Table 5.

If we compare the techniques GW and SW, we see that they have different strengths which could be exploited in different kinds of applications. The SW technique is qualified to detect short-term trends, i.e., trends within the sliding window. The GW technique has a broader view on the time series and it is therefore able to consider long-term trends. We will now show how a combination of the techniques can be utilized to segment and represent another, third time series.

The third time series Sin3 is generated using the sinusoidal function $a \cdot \sin(x \cdot \frac{b}{4})$, where both $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are generated by a discrete-time stochastic process (Markov process). The standard deviation of the white noise is $\sigma = 1$. Here, the segmentation interval is $[0, 215]$ and the range is $[-54, 48]$. The number of samples of the time series is 4,306. The degrees of the approximating polynomials are 2 for both the GW and the SW technique. The length of the sliding window in case of the SW technique is 30. To build a segmentation technique, we combine the criterion c_{DPU} (threshold 2.5) for the GW technique with the criterion c_{MAX} (threshold -0.00075) for the SW technique. The two predicates—that may yield values of 0 or 1—are combined with identical weights of 1 and an overall threshold of 2 is applied. That is, both predicates must be fulfilled to set a segmentation point.

The result of the segmentation is shown in Fig. 4a. In addition to the segmentation points, the local models of the segments, obtained by means of the GW technique, are depicted. It can be stated that the maxima are found with a small delay. This could be improved as shown above for the time series Sin2 (the option "jump to local optimum" is not used here). In Fig. 4a, we see that the approximating polynomials of degree 2 do not represent the local behavior in a satisfying way in many segments. Fig. 4b shows the approximating polynomials with degree 4 for the same

TABLE 4
Result of a Segmentation of Sin1 with the GW Technique: Evaluation of the Precision

	Minimum (Position)	Segmentation Point (Position)	Absolute Error	Relative Error
1	5.63	5.83	0.20	8.7%
2	10.55	10.73	0.18	3.6%
3	19.78	19.80	0.02	0.2%
4	37.10	37.43	0.33	1.9%
5	69.49	70.67	1.18	3.6%
6	130.25	128.17	2.07	3.4%

TABLE 5
Result of a Segmentation of Sin2 with the SW Technique: Evaluation of the Precision

	Maximum (Position)	Segmentation Point (Position)	Absolute Error	Relative Error
1	1.57	1.57	0.00	0.0%
2	4.71	4.73	0.02	0.6%
3	7.85	7.90	0.05	1.6%
4	10.99	11.03	0.04	1.3%
5	14.14	14.17	0.03	0.9%
6	17.28	17.23	0.05	1.6%

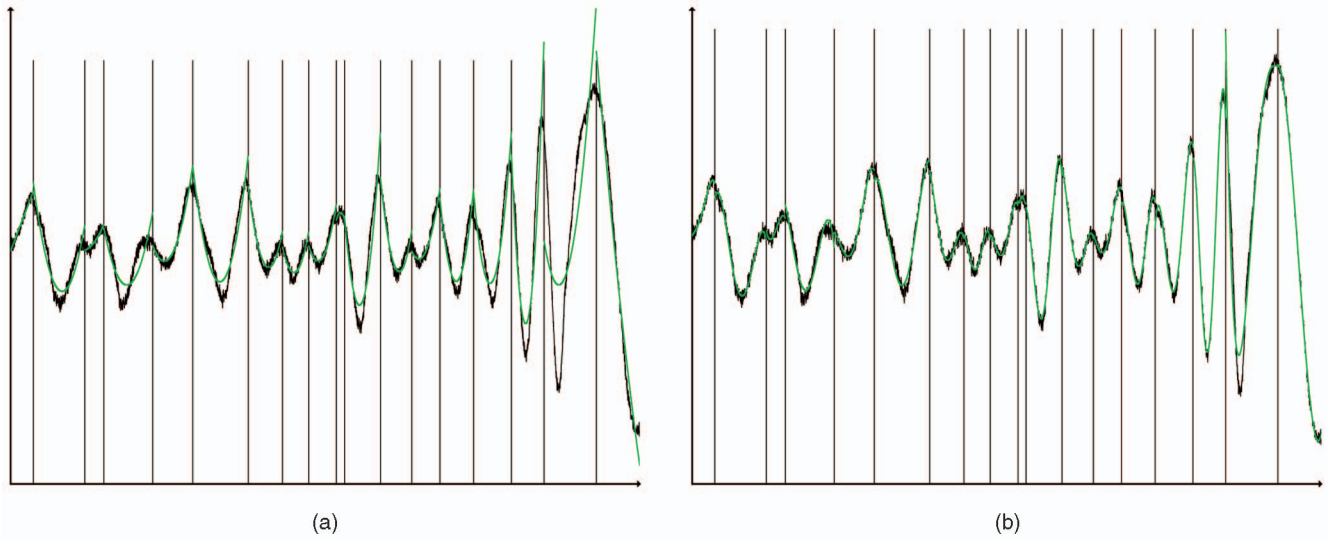

Fig. 4. Results of a segmentation of Sin3 with a combination of SW and GW techniques: time series, local model with (a) degree 2 or (b) degree 4, and segmentation points.

segments. In this case, the time series is represented much better. Only in the second-to-last segment do we see that an approximation with polynomials might not always be appropriate for sinusoidal signals. Here, the reason is that the parameter b (see above) changes significantly within the segment.

In the following, we will investigate the robustness of our approach with respect to varying starting points of the segmentation and varying standard deviation of the white noise, respectively.

In practical applications—for example, to compare similar time series or parts of a time series—it is important that the segmentation points do not vary significantly with slightly varying starting points. To investigate the robustness of our approach, we started the segmentation of Sin3 with five different offsets which were chosen with respect to the length of the first segment as determined above. Table 6 shows the results of this experiment. It can be stated that with a small variation of the starting point up to 20 percent of the segment length, even the first segment boundary is almost not affected. This changes with an offset of 40 and 60 percent, but the second or any subsequent segment boundary are not affected at all. This indicates that SwiftSeg actually is quite robust.

A similar robustness could also be observed for small variations of the standard deviation (white noise). In this case, an adaptation of the thresholds of the criteria is necessary.

TABLE 6
Results of a Segmentation of Sin3 with Various Starting Points: Shift of Segmentation Points

Relative Offset (1st Segment)	Absolute Offset (Data Points)	Shift 1st Segment Boundary	Shift 2nd Segment Boundary	Shift 3rd Segment Boundary
5%	8	0	0	0
10%	15	+1	0	0
20%	30	0	0	0
40%	61	+10	0	0
60%	91	+10	0	0

The runtimes of SwiftSeg for the three time series are given in Table 7 (results of 1,000 repetitions). The results were obtained with a PC with an AMD Athlon 64 3200+ processor with 2.0 GHz and 1 GB main memory. It can be stated that the runtimes per data point are about 0.5-2 μs. In contrast to many other online segmentation techniques (cf. also Section 4.2), the execution times are approximately equal for each data point. Thus, an application under very harsh timing constraints would be possible. From Table 7, it can be seen that the runtime of SW is lower than the runtime of GW. The runtime is highest for the combination of both techniques. The approximating polynomial had degree 2 for each of the time series.

In an application of a segmentation technique, the errors with respect to the approximating polynomial may be quite important. Therefore, we determined the average and the maximum errors within each segment (here, absolute errors). In a second step, we again computed the average and the maximum of those values over all segments. Table 8 shows these values for the experiments in which local models for the segments were obtained, i.e., not for Sin2, where we only used the SW technique. In the case of $K = 4$ for Sin3, all errors are much lower than with $K = 2$.

4.2 Comparison to Other Segmentation Techniques

Having investigated some basic properties of SwiftSeg, we will now assess its runtime and accuracy by comparing it to some other segmentation techniques (cf. Section 2).

TABLE 7
Runtimes for the Segmentation of the Sinusoidal Time Series

Time Series	Number of Samples	Technique	Overall Time in ms	Standard Deviation	Time per Sample in μs
Sin1	4572	GW	3.59	0.103	0.785
Sin2	4222	SW	1.34	0.078	0.317
Sin3	4306	GW & SW	6.81	0.052	1.582

TABLE 8
Absolute Errors with Respect to the Approximating Polynomial (GW Technique)

Number of Segmentation Points	Average Error Within a Segment		Maximum Error Within a Segment	
	Average of all Segments	Maximum of all Segments	Average of all Segments	Maximum of all Segments
Time Series Sin1 (Range: $[-1, 1]$)				
6	0.2376	0.2761	0.7576	0.8116
Time Series Sin3 with $K = 2$ (Range: $[-54, 48]$)				
17	3.027	13.81	10.25	31.32
Time Series Sin3 with $K = 4$ (Range: $[-54, 48]$)				
17	1.210	3.979	4.660	20.97

4.2.1 Comparison to Online Segmentation Techniques

In a first experiment, we compare SwiftSeg to SW-SEG and SWBU-SEG (cf. Section 2), which can both be used for online segmentation:

- SW-SEG is a common sliding window technique based on a piecewise linear approximation of a time series. This approximation is realized with a standard technique (QR decomposition using [27]).
- SWBU-SEG is a typical realization of the SWAB segmentation technique, also with a piecewise linear approximation based on QR decomposition.

In this experiment, the three segmentation techniques use the same criterion, namely c_{RES}.

The time series for this experiment was taken from the PhysioNet signal archive (see [28], for details). All 19 available V4 (cf. [29]) channels from the records of the *European ST-T Database* were used, each with a length of 1,800,000 values (sampled at 250 Hz for two hours). The samples range from -6.895 to 4.850 (the physical unit is mV).

We start the experiment with SW-SEG with a threshold of $th_{\mathrm{RES}} = 0.1$. The segmentation results (overall number of segments, errors, and overall runtime for segmenting the $34.2 \cdot 10^6$ samples) are shown in Table 9.

SWBU-SEG is a technique that aims at improving the representation accuracy at the cost of additional runtime. Here, we parameterize it in a way such that a very similar number of segmentation points is obtained ($BSize = 200$, $MaxErr = 0.241$). Table 9 shows that the accuracy is actually improved substantially, but the runtime increases significantly as well.

Then, SwiftSeg is taken for a segmentation with various degrees of the approximating polynomials (for any degree, we use c_{RES} with a threshold of $th_{\mathrm{RES}} = 0.1$). Here, the time series is approximated with the GW technique. The results for a polynomial degree of 1 reveal the big advantage of the efficient approximation based on orthogonal polynomials. Compared to SW-SEG, the runtime is reduced by a factor of about 155. The other numbers in this row are identical to the ones obtained for SW-SEG as both only differ in the approximation technique. With an increasing degree of the approximating polynomial, the number of segments is reduced (the criterion is the same) and the accuracy is improved. The runtime costs are marginal: As expected, we see a linear increase of runtime with the degree of the approximating polynomial. Even with a degree of 30, the overall runtime is reasonably small. When segmenting with degree 4, one period (heartbeat) roughly corresponds to three to four segments.

TABLE 9
Comparison of Online Segmentation Techniques Using ECG Time Series

Degree	Number of Segmentation Points	Average Error Within a Segment		Maximum Error Within a Segment		Runtime in s
		Average of all Segments	Maximum of all Segments	Average of all Segments	Maximum of all Segments	
SW-SEG						
1	900 352	0.0819	0.6950	0.1981	4.2409	1 636
SWBU-SEG						
1	900 442	0.0392	0.2437	0.0942	0.3923	10 718
SwiftSeg						
0	1 284 414	0.1130	1.7811	0.2325	4.3734	8.71
1	900 352	0.0819	0.6950	0.1981	4.2409	10.53
2	702 027	0.0608	0.2970	0.1792	4.0332	12.33
3	595 690	0.0530	0.3289	0.1590	3.5359	14.07
4	519 226	0.0480	0.3503	0.1516	3.3178	16.12
5	475 771	0.0436	0.3334	0.1448	3.0183	18.04
6	446 617	0.0399	0.2180	0.1411	2.5810	19.80
7	417 073	0.0370	0.1684	0.1361	2.1210	21.68
8	389 267	0.0346	0.2667	0.1329	2.4074	23.42
9	366 027	0.0322	0.2181	0.1313	1.9969	25.28
10	345 265	0.0302	0.1765	0.1312	2.2451	27.03
...	...	...	...	...	...	...
20	207 062	0.0200	0.1200	0.1285	0.5668	44.96
...	...	...	...	...	...	...
30	172 704	0.0166	0.0724	0.1236	0.8267	62.58

In this first experiment, SwiftSeg needs a polynomial degree of about 7 to outperform SWBU-SEG with a polynomial degree of 1 in terms of accuracy (third column in Table 9). Apart from the bottom-up improvement step, which is missing in SwiftSeg, one reason is certainly the smaller number of segments. Another reason is that we did not yet exploit the additional segmentation possibilities of SwiftSeg. This will now be done in a second experiment, where we employ a time series that is also quite long.

The time series Lorenz is based on the so-called Lorenz attractor, a three-dimensional structure corresponding to the long-term behavior of a chaotic flow (cf. [30]). The variables x, y, and z of this system are defined by three coupled ordinary differential equations and evolve over time in a complex, nonrepeating way. The parameters of the equations, called the Prantl number σ, the Rayleigh number ρ, and a physical proportion β, were set to 10, 28, and $\frac{8}{3}$, respectively. For this set of parameters, the system exhibits chaotic behavior and therefore defines a so-called strange attractor (cf. [31]). To generate the data set, we calculate 10^7 samples with a step width of 10^{-2} (i.e., increasing x, y, and z by $10^{-2} \cdot \frac{\mathrm{d}}{\mathrm{d}t}$ in each iteration) starting at $(x, y, z) = (0, 1, 1)$ and use the time series of the x-coordinate of each position (the range is $[-21.09, 21.29]$).

First, the time series is segmented with SwiftSeg at each local maximum using an SW technique with a polynomial of degree 2, window length 10, and the criterion c_{MAX} with a threshold of $th_{\mathrm{MAX}} = -5 \cdot 10^{-4}$. With a runtime of about 2.5 s only (see Table 10), this leads to 115,786 segmentation points.

TABLE 10
Comparison of Online Segmentation Techniques Using the Lorenz Time Series

Degree	Average Error Within a Segment		Maximum Error Within a Segment		Run-Time in s
	Average of all Segments	Maximum of all Segments	Average of all Segments	Maximum of all Segments	
SwiftSeg (Segmentation Only)					
none	-	-	-	-	2.48
SwiftSeg (Segmentation and Representation)					
0	3.7219	7.7578	9.6437	21.4608	3.59
1	2.9691	6.5246	8.5582	20.1578	4.23
2	1.4067	5.3834	4.7996	19.0484	4.77
3	1.0920	5.3099	3.3934	19.2360	5.36
4	0.8661	3.7792	2.5641	13.4999	5.89
5	0.6638	3.0961	2.0283	12.3045	6.42
SWBU-SEG					
1	1.9236	5.9364	4.9077	14.4594	6 460

Then, we add an approximation based on the GW technique (various degrees) to get a piecewise polynomial representation of the time series. In Table 10, we state an almost linear increase of the runtime with the degree of the approximating polynomial. From degree 0 to 1, the step size is a bit larger, which is due to the fact that, for degree 0, the loop in Algorithm 2 is not entered. In contrast to the first experiment, the improvement of accuracy with an increasing degree is remarkable here.

To underline the significance of this result, we again run SWBU-SEG with a polynomial degree of 1 and parameters $BSize = 500$ and $MaxErr = 1{,}577$ to obtain a very similar number of segmentation points (here, 115,791). Again, it can be seen that SWBU-SEG yields more accurate results than SwiftSeg with the same polynomial degree. In this experiment, however, SwiftSeg outperforms SWBU-SEG if polynomials with degree 2 or higher are used. The runtimes of SwiftSeg are much lower not only due to the fast approximation techniques, but also due to the missing bottom-up steps. We also want to emphasize that the runtimes of SwiftSeg are not only low. The runtimes *per sample* are nearly constant, which is an important feature for data streaming applications (cf. [32]). In an online application with harsh timing constraints, where it is necessary to decide upon a segmentation boundary before the next sample arrives, the worst-case time complexity for a single sample is an important issue.

Altogether, these two experiments outlined the runtime benefits from applying the fast polynomial approximation techniques and the improvement of the representation accuracy that can be gained by using higher polynomial degrees. The computational costs of using higher degrees are quite low, the memory costs from storing the representation of a time series instead of the raw time series linearly depend on the number of segments and the polynomial degree. SwiftSeg outperforms SWBU-SEG in a standard realization (QR decomposition and piecewise linear representation), but it also became clear that SWBU-SEG could be improved significantly by adopting the GW technique for the least-squares approximation and by taking polynomials with higher degrees. Anyway, the original description of the SWAB technique is independent from concrete realizations of the approximation steps (cf. [1], [2]).

TABLE 11
Comparison to an Optimal Segmentation Technique: Time Series and Number of Segmentation Points

	ER	WL	SS	ECG
Range	$[0.497, 0.724]$	$[5.4, 12.5]$	$[0, 1]$	$[-410, 800]$
Number of Samples	2 567	2 191	1 000	2 500
Number of Segmentation Points	36	42	22	48

4.2.2 Comparison to an Optimal Segmentation Technique

In a third experiment, we compare SwiftSeg to TD-SEG (cf. Section 2), an offline segmentation technique:

- TD-SEG leads to optimal results with respect to a given criterion and a given number of segments. It is realized with a dynamic programming approach.

We run SwiftSeg with the GW technique, various polynomial degrees, and the segmentation criterion c_{SQE} first. The thresholds th_{SQE} are chosen in a way such that we get the same number of segments for all degrees to make a comparison possible. Then, TD-SEG is executed with the same criterion, the same degrees, and the number of segments determined by SwiftSeg.

To make this experiment feasible, we need relatively short time series as the runtimes of TD-SEG are quite high. From the benchmark data sets used in [1] for a comparison of various segmentation algorithms, four are available for our experiments, namely, Exchange Rates (ER), Water Level (WL), Space Shuttle (SS), and ECG (cf. Fig. 6 and Table 11).

As above, the absolute errors within the segments are evaluated by computing the average error over all samples. Results obtained on the four time series are shown in Fig. 5.

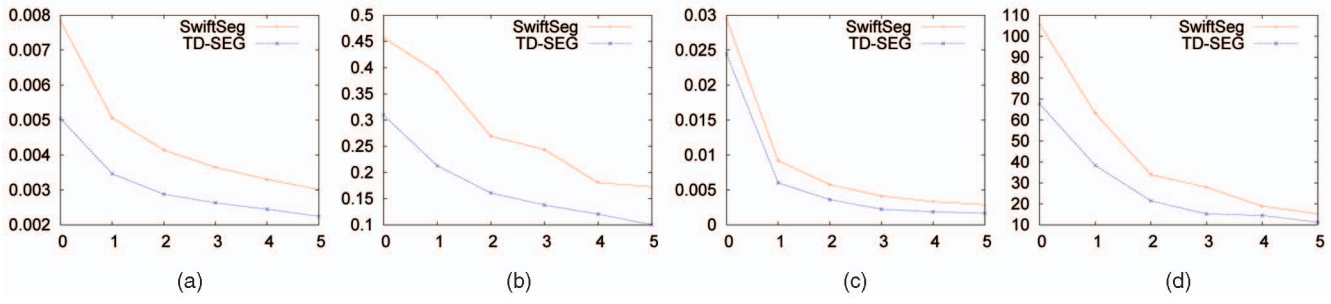

Fig. 5. Comparison to an optimal segmentation technique: modeling accuracy (average absolute error) of TD-SEG and SwiftSeg with respect to the degree of the approximating polynomial. (a) Exchange Rates (ER). (b) Water Level (WL). (c) Space Shuttle (SS). (d) ECG.

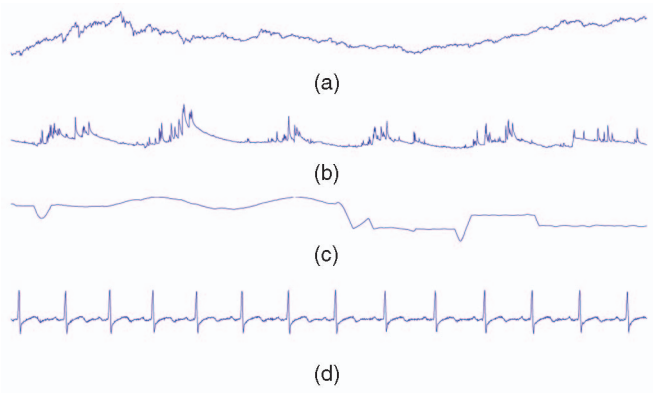

Fig. 6. Time series data used for the comparison to an optimal segmentation technique. (a) Exchange Rates (ER) with 2,567 Samples. (b) Water Level (WL) with 2,191 Samples. (c) Space Shuttle (SS) with 1,000 Samples. (d) ECG with 2,500 Samples.

It can be seen that the representation accuracy of SwiftSeg is quite close to that of an optimal segmentation approach. Depending on the time series or the polynomial degree, SwiftSeg is able to reach nearly the accuracy of TD-SEG and, thus, the best possible segmentation with the criterion c_{SQE}. This is remarkable since the runtimes of TD-SEG are considerably higher: Although the dynamic programming approach improves the runtime of TD-SEG substantially, there is still a factor of about 10^8 (depending on the data set and the number of segments) compared to SwiftSeg if a standard approximation based on QR decomposition is used. However, TD-SEG could also be accelerated with the GW technique.

Altogether, the experiments in this section have shown that SwiftSeg is a technique for online time series segmentation that is not only extremely fast but also leads to representation results that come quite close to that of an optimal technique. If even higher accuracy is needed, SwiftSeg could be modified in a SWAB-line manner. This, however, is at the expense of additional runtime. Moreover, the worst-case runtime per sample is much higher due to the bottom-up step which is sometimes interposed. In any case, the concrete realization depends on the needs of a particular application.

5 Summary and Outlook

In this paper, we presented a new framework for online time series segmentation. The goal was to provide a segmentation technique that is both, fast and accurate. This is achieved with a polynomial approximation of the time series using a basis of orthogonal polynomials. Moreover, the approach provides a piecewise polynomial representation based on the orthogonal expansion of the approximating polynomial. The coefficients of the orthogonal expansion, that can be interpreted as optimal (in the least-squares sense) estimators of average, slope, curvature, etc., of the time series within a segment, may not only be used to control the segmentation process. They could also be taken to measure the similarity of segments (see, e.g., [33], [34]). In our future work, we will also investigate the use of other orthogonal functions for the approximation (see, e.g., [35]), the clustering and classification of time series motifs obtained with this segmentation technique, and the application of the framework in fields such as gesture spotting and activity recognition [36], analysis of financial time series [37], and handwriting analysis [38], [39]. We will also focus on a segmentation of multivariate time series and a comparison to some other techniques, e.g., techniques based on Kalman filters.

In the interest of competitive research, our data sets and the code of the update techniques used for this work will be made available by e-mailing the last author.

Acknowledgments

This work was supported by the German Research Foundation (DFG) under the project number SI 674/6-1. The authors would like to thank G. Pisinger and A. Zimmermann, who gave an important hint to the update formulas for the squared error (residuum) of the polynomial approximation, and T. Reitmaier, who implemented the update for the SW technique. The authors also highly appreciate the support of E. Keogh, who provided some of the data sets, and they thank G. Leha and the anonymous reviewers of the paper for many very valuable comments that helped them to improve the quality of the paper.

References

[1] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An Online Algorithm for Segmenting Time Series," *Proc. IEEE Int'l Conf. Data Mining*, pp. 289-296, 2001.

[2] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," *Data Mining in Time Series Databases*, M. Last, A. Kandel, and H. Bunke, eds., vol. 57, ch. 1, pp. 1-22, World Scientific Publishing, 2004.

[3] H.J.L.M. Vullings, M.H.G. Verhaegen, and H.B. Verbruggen, "ECG Segmentation Using Time-Warping," *Proc. Second Int'l Symp. Advances in Intelligent Data Analysis, Reasoning about Data*, pp. 275-285, 1997.

[4] D. Lemire, "A Better Alternative to Piecewise Linear Time Series Segmentation," *Proc. SIAM Data Mining '07*, pp. 545-550, 2007.

[5] X. Liu, Z. Lin, and H. Wang, "Novel Online Methods for Time Series Segmentation," *IEEE Trans. Knowledge and Data Eng.*, vol. 20, no. 12, pp. 1616-1626, Dec. 2008.

[6] W. Fitzgerald, D. Lemire, and M. Brooks, "Quasi-Monotonic Segmentation of State Variable Behaviour for Reactive Control," *Proc. 20th Nat'l Conf. Artificial Intelligence*, pp. 1145-1150, 2005.

[7] T. Mori, Y. Nejigane, M. Shimosaka, Y. Segawa, T. Harada, and T. Sato, "Online Recognition and Segmentation for Time-Series Motion with HMM and Conceptual Relation of Actions," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, pp. 2856-2863, 2005.

[8] J. Kohlmorgen, S. Lemm, K.-R. Muller, S. Liehr, and K. Pawelzik, "Fast Change Point Detection in Switching Dynamics Using a Hidden Markov Model of Prediction Experts," *Proc. Ninth Int'l Conf. Artificial Neural Networks*, vol. 1, pp. 204-209, 1999.

[9] J. Kohlmorgen and S. Lemm, "An On-Line Method for Segmentation and Identification of Non-Stationary Time Series," *Proc. IEEE Signal Processing Soc. Workshop Neural Networks for Signal Processing XI*, pp. 113-122, 2001.

[10] Y. Gorshkov, I. Kokshenev, Y. Bodyanskiy, V. Kolodyazhniy, and O. Shylo, "Robust Recursive Fuzzy Clustering-Based Segmentation of Biological Time Series," *Proc. Int'l Symp. Evolving Fuzzy Systems*, pp. 101-105, 2006.

[11] Y. Rao and J. Principe, "A Fast On-Line Generalized Eigendecomposition Algorithm for Time Series Segmentation," *Proc. Adaptive Systems for Signal Processing, Comm., and Control Symp.*, pp. 266-271, 2000.

[12] A. Li, S. He, and Z. Qin, "Real-Time Segmenting Time Series Data," *Proc. Web Technologies and Applications—Fifth Asian-Pacific Web Conf.*, X. Zhou, Y. Zhang, and M.E. Orlowska, eds., pp. 178-186, 2003.

[13] R. Bellman, "On the Approximation of Curves by Line Segments Using Dynamic Programming," *Comm. ACM,* vol. 4, no. 6, p. 284, 1961.
[14] L. Tang, B. Cui, H. Li, G. Miao, D. Yang, and X. Zhou, "Effective Variation Management for Pseudo Periodical Streams," *Proc. ACM SIGMOD '07,* pp. 257-268, 2007.
[15] H. Shena, C. Xu, X. Han, and Y. Pan, "Stock Tracking: A New Multi-Dimensional Stock Forecasting Approach," *Proc. Eighth Int'l Conf. Information Fusion,* vol. 2, pp. 1375-1382, 2005.
[16] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture Spotting with Body-Worn Inertial Sensors to Detect User Activities," *Pattern Recognition,* vol. 41, pp. 2010-2024, 2008.
[17] G.H. Golub and C.F. van Loan, *Matrix Computations,* third ed. Johns Hopkins Univ. Press, 1996.
[18] Å. Björck, *Numerical Methods for Least Squares Problems.* SIAM, 1996.
[19] E. Fuchs, C. Gruber, T. Reitmaier, and B. Sick, "Processing Short-Term and Long-Term Information with a Combination of Polynomial Approximation Techniques and Time-Delay Neural Networks," *IEEE Trans. Neural Networks,* vol. 20, no. 9, pp. 1450-1462, Sept. 2009.
[20] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, 1998.
[21] *Feature Extraction, Construction, and Selection: A Data Mining Perspective,* H. Liu and H. Motoda, eds. Kluwer Academic Publishers, 1998.
[22] E. Fuchs, "Schnelle Quadratmittelapproximation in gleitenden Zeitfenstern mit diskreten orthogonalen Polynomen," PhD dissertation, Univ. of Passau, 1999.
[23] E. Fuchs and K. Donner, "Fast Least-Squares Polynomial Approximation in Moving Time Windows," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing,* vol. 3, pp. 1965-1968, 1997.
[24] E. Fuchs, "On Discrete Polynomial Least-Squares Approximation in Moving Time Windows," *Applications and Computation of Orthogonal Polynomials,* W. Gautschi, G. Golub, and G. Opfer, eds., vol. 131, pp. 93-107, Birkhäuser, 1999.
[25] A. Nikiforov, S. Suslov, and V. Uvarov, *Classical Orthogonal Polynomials of a Discrete Variable.* Springer-Verlag, 1991.
[26] S. Elhay, G.H. Golub, and J. Kautsky, "Updating and Downdating of Orthogonal Polynomials with Data Fitting Applications," *SIAM J. Matrix Analysis and Applications,* vol. 12, no. 2, pp. 327-353, 1991.
[27] R.F. Boisvert, J. Hicklin, B. Miller, C. Moler, R. Pozo, K. Remington, and P. Webb, "JAMA: A Java Matrix Package," http://math.nist.gov/javanumerics/jama/, 2005.
[28] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, and H.E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," *Circulation,* vol. 101, no. 23, pp. 215-220, 2000.
[29] R. Klinge, *Das Elektrokardiogramm.* Thieme, 2002.
[30] E.N. Lorenz, "Deterministic Nonperiodic Flow," *J. Atmospheric Sciences,* vol. 20, no. 2, pp. 130-141, 1963.
[31] W. Tucker, "A Rigorous ODE Solver and Smale's 14th Problem," *Foundations of Computational Math.,* vol. 2, no. 1, pp. 53-117, 2002.
[32] M.R. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on Data Streams," *External Memory Algorithms,* J.M. Abello and J.S. Vitter, eds., pp. 107-118, Am. Math. Soc., 1999.
[33] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," *Proc. 34th Int'l Conf. Very Large Data Bases,* pp. 1542-1552, 2008.
[34] P.-F. Marteau, "Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 2, pp. 306-318, Feb. 2009.
[35] E. Fuchs, T. Hanning, and O. Schwarz, "An Update Algorithm for Fourier Coefficients," *Proc. 12th European Signal Processing Conf.,* pp. 1509-1512, 2004.
[36] J.A. Ward, P. Lukowicz, G. Tröster, and T.E. Starner, "Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 10, pp. 1553-1567, Oct. 2006.
[37] J. Mager, U. Paasche, and B. Sick, "Forecasting Financial Time Series with Support Vector Machines Based on Dynamic Kernels," *Proc. IEEE Conf. Soft Computing in Industrial Applications,* pp. 252-257, 2008.
[38] C. Gruber, T. Gruber, and B. Sick, "Online Signature Verification with New Time Series Kernels for Support Vector Machines," *Advances in Biometrics,* D. Zhang and A.K. Jain, eds., pp. 500-508, Springer-Verlog, 2006.
[39] T. Artieres, S. Marukatat, and P. Gallinari, "Online Handwritten Shape Recognition Using Segmental Hidden Markov Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 2, pp. 205-217, Feb. 2007.

Erich Fuchs received the diploma in computer science from the University of the Armed Forces in Munich, Germany, in 1984, and the PhD degree in computer science from the University of Passau, Germany, in 1999. Currently, he is a managing director of the Institute for Software Systems in Technical Applications at the University of Passau. His interests include approximation methods, orthogonal polynomials, and their applications in image and signal processing.

Thiemo Gruber received the diploma in computer science from the University of Passau, Germany, in 2007. Currently, he is working toward the PhD degree in computer science in the Faculty of Computer Science and Mathematics at the University of Passau, where he is conducting research and development in the areas of machine learning (in particular, time series analysis) with applications in gesture spotting and activity recognition, online signature verification and identification as well as medical applications of biometric writing systems.

Jiri Nitschke received the diploma in computer science from the University of Passau, Germany, in 2009. Currently, he is with the newspaper *Sueddeutsche Zeitung*. His interests include algebra, set theory, and machine learning.

Bernhard Sick received the diploma in 1992 and the PhD degree in 1999, both in computer science, from the University of Passau, Germany. Currently, he is an assistant professor (Privatdozent) at this university. He is conducting research and development in the areas of theory and application of soft-computing techniques, organic computing, data mining, intrusion detection, and biometrics. He has authored more than 70 peer-reviewed publications in these areas. He is a member of the IEEE (Systems, Man, and Cybernetics Society, Computer Society, and Computational Intelligence Society) and GI (Gesellschaft fuer Informatik). He is an associate editor of the *IEEE Transactions on Systems, Man, and Cybernetics, Part B*; he holds one patent and has received several thesis, best paper, teaching, and inventor awards.