Time Series Segmentation and Segment Representation

Herbert Teun Kruitbosch, Ioannis Giotis and Michael Biehl

Abstract—This paper compares three time series segmentation techniques: sliding window, bottom up and top down. Each technique is tested w.r.t. each of four segment representations. All representations are least square fits of a set of basis functions. These sets of basis functions are: monomials, Chebyshev polynomials, Fourier series and Haar wavelets. The idea is that the least square estimated parameters of a segment allow to classify segments depending on the quality of the segmentation and representation. This paper conducts two tests considering each of the four representations without segmentation. At first, the variance of the least square estimated parameters when subjected to noise and zero padding at both sides is measured. Subsequently, the representation's capability of detecting similar and different segments is measured, by comparing the empirical distributions of distances between similar segments and between different segments. The representation's distance measure is defined by de euclidean distance between the estimated parameters of two segments. Finally the segmentation techniques are tested. Synthetic signals with additive white Gaussian noise are segmented and afterwards the segments are classified. The performance of the segmentation is illustrated via de performance of the classification, by comparing it with the classification of a *perfect segmentation*. The experiments show that the Chebyshev basis is the best representation and that the bottom up technique results in the best classification.

Index Terms—time series, representations, linear least square fit, linear least square estimation, orthogonal basis, Fourier, Chebyshev, shape based representation, wavelets, segmentation, sliding window, bottom up, top down, IJkdijk.

1 Introduction

A time series is a sequence of measurements taken at successive moments in time. For example, the IJkdijk project measures temperature, water pressure and other physical quantities occurring on the Dutch dikes, typically at equidistant moments in time. During distincitive periods in time, weather and dike conditions lead to characteristic behaviour in the measurements. Locating meaningful segments, which identify such periods, may help detect anomalies or provide useful input for other machine learning algorithms.

However, time series often contain a large number of measurements, possibly too much to store in memory or to do calculations within a reasonable amount of time. Therefore mining algorithms can benefit from a reduced, piece-wise representation which approximates the time series. Moreover, important time series mining tasks like clustering, classification and indexing benefit from a piecewise representation in order to focus on *behaviour* that only exist in a certain part of a time series. However, a piecewise representation requires a time series segmentation, which is an ill-posed problem, since different criteria regarding the amount of error or the number of segments can be used. Moreover, it is hard to define an optimum between both; more segments lead to a lower error, but may also allow the segments to *overfit* and hence capture noise. This paper will discuss three popular algorithms for segmentation: sliding window, bottom up and top down [4].

All segmentation algorithms discussed need a representation to fit on the segments. The representation is important, since certain *representational models* may not be able to represent important shapes occuring in the time series accurately. For example consider the possibility of segmenting a $\frown$-like shape as one segment: a linear approximation will result in a higher error than a second order polynomial approximation. Hence a segmentation based on a second order polynomial will, with higher probability, classify such a shape as one segment than segmentation based on a linear approximation. Apart from the segmentation *quality*, there are some other important considerations regarding representation, for example sensitivity to noise. This document will discuss four representations which are all based on a linear least square representation for a given set of basis functions. These sets are: monomials ($1, x, x^2, ...$), Chebyshev polynomials (see section 2.1.3), Fourier series ($1, \exp(-i2\pi x/N), ...$) and Haar wavelets (see section 2.1.5).

This document will test the aforementioned segmentation algorithms and the various representation models with respect to *classification* of synthetically created time series. An important note is that segments within such a synthetic time series are based on a model $y[n] = f(n) + w[n]$ where f is the model and w is white Gaussian noise: $w \sim \mathcal{N}(0, \sigma^2)$. Hence a least square fit on a set of basis functions matching those of f will have a bias to perform better and therefore these tests should be repeated on *real world data*.

This paper has been organised in the following way. Section 2 gives a brief overview of all segmentation techniques and representations. It provides some considerations regarding a linear least square fit, and discusses the orthogonal property which the three latter bases have. The section also discusses the 3 segmentation algorithms. Section 3 shows results regarding the performance of the different representations and the segmentation techniques. Finally, section 4 presents the conclusions and directions for future work.

2 Background and related work

The previous section introduced two important concepts regarding time series segmentation, the segmentation algorithm and the segment representation. Each segmentation algorithm aforementioned takes a representation model as input. This section first discusses these representation models and then the segmentation algorithms.

2.1 Segment Representations

Each representation represents a segment, hence we first define the term *segment*. For notational convenience, we allow the indices of a segment S to go from 1 to $j-i$ instead of from i to $j-1$.

Definition 2.1. *Given a time series $T = t_1, \ldots, t_n$, a* ***segment*** *$S = s_1, \ldots, s_{j-i}$ is a part of T such that $S = t_i, t_{i+1}, \ldots, t_{j-1}$ for some $1 \leq i < j \leq n+1$.*

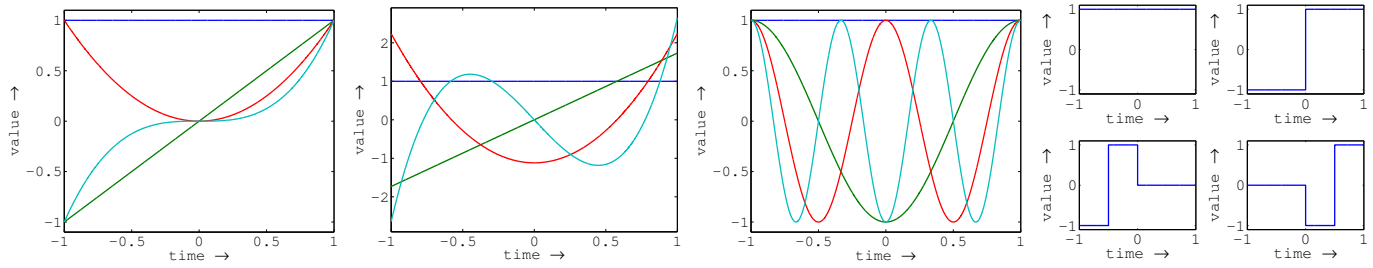
Less formally, a segment is a part of a time series which does not contain *holes*. This paper will now discuss *linear* least square parameter estimation and the representational models.

2.1.1 Linear Least Square Estimation

This document will discuss four representations which are all linear least square estimations on a segment, as introduced by Gauss [3] [6].

- *Herbert Teun Kruitbosch is a Msc. student at the university of Groningen, E-mail: herbertkruitbosch@gmail.com,*
- *Ioannis Giotis, second supervisor, Target Holding B.V. Groningen, I.E.Giotis@rug.nl,*
- *prof. dr. M. (Michael) Biehl, first supervisor, Professor University of Groningen, m.biehl@rug.nl.*

Manuscript received 20 December 2013 as a part of the 15EC internship for the MSc Computing Science of Herbert Teun Kruitbosch.



(a) Monomial basis (b) Chebyshev basis (c) Fourier basis (d) Haar basis

Fig. 1. The first 4 functions of a monomial basis, Chebyshev basis, Fourier basis and Wavelet basis.

A linear least square estimation can be justified if we assume that a segment $S = s_1, \ldots, s_N$, can be described as $s[n] = \sum_{m=1}^{M} \theta_m f_m(n) + w[n]$, where $f_1, \ldots, f_M$ is a set of basis functions and $w \sim \mathcal{N}(0, \sigma^2)$. Here $w[n] \sim \mathcal{N}(0, \sigma^2)$ denotes white Gaussian noise with variance σ^2 which is identically distributed and independent along the samples $n = 1, \ldots, N$. A (linear) least squeare estimation determines the unknown parameters $\theta_1, \ldots, \theta_M$, such that the *error* between $s[n]$ and $\sum_{m=1}^{M} \theta_m f_m(n)$ is minimal, e.g.

$$\hat{\theta}_{LSE} = \text{argmin}_\theta \sum_{n=1}^{N} \left(s[n] - \sum_{m=1}^{M} \theta_m f_m(n) \right)^2. \quad (1)$$

According to the Central Limit Theorem, the assumption that the noise is normally distributed is valid for a segment containing many samples. However, the assumption that our set of basis functions form an adequate basis, such that the noise w is uncorrelated w.r.t. samples, may be invalid.

Correlation exists if the physical properties of the time series do not allow it to be segmented in such a way that one segment can be represented by the addition of white Gaussian noise and a linear combination of basis functions. However, even if the basis functions are adequate and the physical properties allow for such segments to be found, a segmentation algorithm may not find these *correct segments*. Instead, the algorithm may find a segment which does not coincide with a correct segment: a segment may *be slightly wrong*, overlap multiple correct segments or be a small part of a correct segment. Such a *wrong segment* will have correlated noise with respect to the model with basis functions $f_1, \ldots, f_M$.

If the noise, however, is uncorrelated, the least square estimator is, by the Gauss-Markov Theorem, a minimal variance unbiased estimator. A segmentation algorithm that finds segments which have uncorrelated noise will result in estimated parameters which are least sensitive to noise with respect to the Cramér-Rao bound.

2.1.2 Monomial representation

Approximating a segment using a larger degree of monomials allows for more representational freedom than a linear approximation. Therefore, the first considered representation is a linear least square fit with the first M monomials as basis.

The least square fit needs to be scale and resolution invariant in order to meaningfully compare estimated parameters. Consider the basis $f_i = (2x/N - 1)^i$ for $i = 0, 1, \ldots, M-1$ for a segment with $S = s_0, \ldots, s_N$ at times $t_0 = 0, \ldots, t_N = N$. Then $(2x/N - 1)$ is defined on $[-1, 1]$ and hence the shape of the basis function is independent of the resolution or position in the time dimension. Some of these monomials are shown in figure 1a.

In order to solve the parameters θ using a least square estimation, we construct a matrix A containing the basis functions values: $a_{ij} = f_i(t_j)$. Then the least square fit is defined and solved by:

$$\hat{\theta}_{LSE} = \text{argmin}_\theta \|\mathbf{A}\theta - \mathbf{s}\|_2^2 = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{s}. [3] \quad (2)$$

The parameters $\hat{\theta}$ are the result. Such a fit allows us to reduce the representation of one segment and allows for a distance measure to be defined on the parameters. For example, the euclidean distance for two segments, with parameters θ_1 and θ_2 respectively, is: $\|\hat{\theta}_1 - \hat{\theta}_2\|_2$.

2.1.3 Chebyshev representation

A Chebyshev basis of order $M-1$ is a set of M orthogonal polynomials, they are recursively defined by:

$$f_{-1}(x) = 0, f_0(x) = 1$$
$$f_{i+1} = (x - \frac{N-1}{2}) f_i(x) - \frac{i^2(N^2 - k^2)}{16i^2 - 4} f_{i-1}(x)$$

for a time series measured at times $1, 2, \ldots, N$ [2]. The function f_{-1} is not part of the basis. Some of the polynomials of this basis are shown in figure 1b. Considering the least square estimation (2) for this basis, the $D = (\mathbf{A}^T\mathbf{A})^{-1}$ term is diagonal, since A is orthogonal [2]. The i-th diagonal element of D is defined by:

$$d_{ii} = \frac{1}{\sum_{j=1}^{N} f_i(j)^2}.$$

This simplifies the least square estimation for an orthogonal basis to:

$$\hat{\theta}_i = \frac{1}{d_{ii}} \sum_{j=1}^{N} f_i(j) \cdot s_j.$$

Meaning that the parameter θ_i only depends on the basis function f_i. Therefore the value of θ_i does not change when the order of the polynomial $M-1$ changes.

2.1.4 Fourier representation

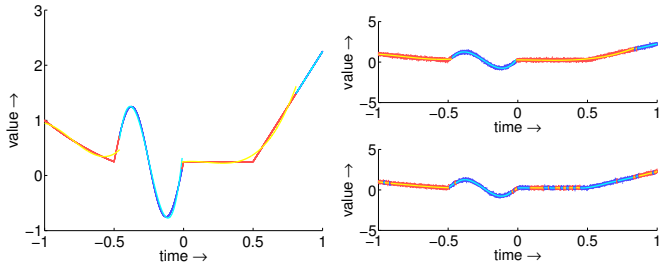
The Fourier representation is also based on an orthogonal set of functions [1], however these are periodic and not polynomials. The basis functions are defined by:

$$f_i(x) = \exp(-\sqrt{-1} \cdot 2\pi \cdot i \cdot x/N),$$

of which some are shown in figure 1c. Similar to section 2.1.3 we can define $d_{ii} = \frac{1}{\sum_{j=1}^{N} f_i(j)^2}$ and reduce the least square fit to:

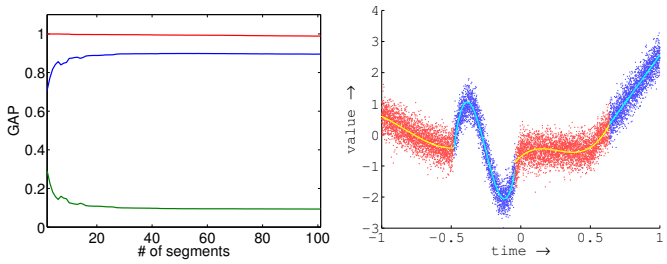
$$\hat{\theta}_i = \frac{1}{d_{ii}} \sum_{j=1}^{N} \exp(-\sqrt{-1} \cdot 2\pi \cdot i \cdot j/N).) \cdot s_j.$$

This is known as the Discrete Fourier Transform. It is intuitive to calculate these $\hat{\theta}_i$ using a Fast Fourier Transform in order to gain speed. However, this would require zero padding the segment to a



(a) No noise (b) Additive noise $\mathcal{N}(0,0.1)$

Fig. 2. Segmentation of a time series using sliding window. The red and dark blue points are data, the yellow and light blue curves are the fitting, using the Chebyshev basis of section 2.1.3. The signals in (b) contains additive white Gaussian noise $\mathcal{N}(0,0.1)$. The two segmentations in (b) show the sensitivity to parameters. Whilst the top segmentation is quite well, using $\varepsilon_{max} = 0.012$, the bottom one performs significantly worse using $\varepsilon_{max} = 0.010$.



(a) GAP (b) Segmentation

Fig. 3. Segmentation of a time series using the bottom up algorithm. The red and dark blue points are data, the yellow and light blue curves are the fitting, using the Chebyshev basis of section 2.1.3. The colors indicate segments found by the algorithm. The signal is a combination of different polynomials and a sine wave, with additive white Gaussian noise $\mathcal{N}(0,0.1)$. The GAP statistic used in (a) is discussed in section 2.2.2. The red curve is the error of the random order time series (see section 2.2.2), the green curve of the error of the time series and the blue curve the GAP statistic (again, see section 2.2.2). The segmentation finds 6 segments, which allows for a maximal GAP value (blue curve in (a)).

size 2^n, with $n \in \mathbb{N}$. Therefore the Fast Fourier Transform is not scale and resolution invariant with respect to the time dimension. Consider for example two similar segments, however one has 128 samples and needs no padding. The other has 129 samples and is padded to 256 samples. Therefore the basis functions differ significantly, due to the division by the number of samples in the exponent of f_i. Thus, even though the segments are similar, the they will have different estimated parameters.

2.1.5 Haar wavelet representation

The Haar wavelets series define a *binary tree* of basis functions. Before defining the tree, we define

$$f_0(x) = 1$$
$$f(x|x_r,x_0,x_l) = \begin{cases} -1 & x_r \le x \le x_0 \\ \frac{x_0-x_r+1}{x_l-x_0} \approx 1 & x_0 < x \le x_l \\ 0 & otherwise. \end{cases}$$

For N samples the root of the tree is $f_1(x) = f(x;0,\lfloor N/2 \rfloor, N)$. The parameters x_r, x_0 and x_l define respectively the left bound, the center and the right bound of the wavelet. A Haar wavelet basis function will have $x_0 \approx \frac{1}{2}(x_r + x_l)$. Then the two children of a function $f(x|x_r,x_0,x_l)$ are defined recursively:

$$left(f)(x|x_r,x_0,x_l) = f(x|x_r, \lfloor \tfrac{1}{2}(x_r+x_0) \rfloor, x_0)$$
$$right(f)(x|x_r,x_0,x_l) = f(x|x_0+1, \lfloor \tfrac{1}{2}(x_0+1+x_l) \rfloor, x_l).$$

Given this definition, $\frac{x_0-x_r+1}{x_l-x_0} \approx 1$, since $x_0 - x_r + 1 \approx x_l - x_0$ for the functions in the tree. The fraction is used to ensure that the basis functions are orthogonal for $x = 1,\ldots,N$. Regular definitions of the Haar wavelets do not use this fraction. The functions in the Haar wavelet basis have the same shape, but each level further from the root of the tree, halves the size of the period (see figure 1d). Moreover, each function defines exactly one period at a certain location on the time axis, and is 0 otherwise.

Haar wavelets can be compared with the Fourier basis in the sense that the period of one *block wave* gets smaller when we add more basis functions. However, the wavelets which are deeper in the tree, are mostly 0. Meaning that the coëfficient, or parameter, of such a wavelet can represent a local shape where the wavelet is non-zero. For example, the third and fourth wavelets, shown in figure 1d, have the same shape but sifted w.r.t. time. Hence both sides can be *fitted* independently. On the other hand, the Fourier basis considers more *wavelengths*. For example, the first eight functions of the Fourier basis contains sine-functions where for the wavelengths $\frac{1}{1}$, $\frac{1}{2}$, $\frac{1}{3}$, ... and $\frac{1}{7}$ of the time series length. For the Haar wavelets the first eight basis functions include the wavelengths $\frac{1}{1}$, $\frac{1}{2}$ and $\frac{1}{4}$ of the time series length.

2.2 Segmentation

Now that we have discussed 4 bases which can represent a segment, we can discuss the segmentation process. First, we define a segmentation as follows:

Definition 2.2. *Given a time series T, a segmentation S is a partition of T, such that for each part $s \in S$ it holds that: if $i \le j \le k$ and $t_i, t_k \in s$ then $t_j \in s$. These parts are called segments.*

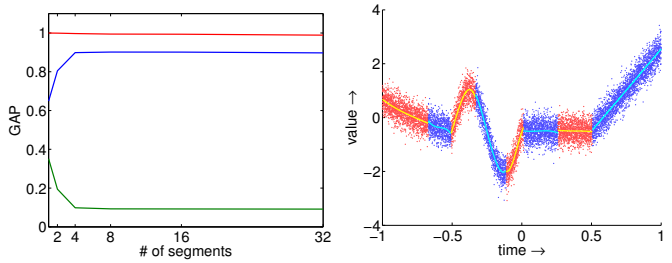
Less formally, a segment is not allowed to have *holes*. For simplicity, the three algorithms in this section return the *breakpoints* of the segments, instead of the segments themselves. They are defined as follows:

Definition 2.3. *Given a segmentation S of a time series $T = t_1 \ldots t_n$, a set of **breakpoints** $B = b_1,\ldots,b_m \subset \mathbb{R}$ of S is defined such that $S = \{(t_{b_i},\ldots,t_{b_{i+1}-1}) | i \in 1 \ldots m-1\}$. Hence by definition $b_1 = 1$ and $b_m = n+1$. Less formally, the other breakpoints define where the time series is* split *into segments.*

Determining the *optimal* number of segments is an ill-posed problem; more segments result in a lower error. However, it is not realistic that all segments only contain a few samples. This trade-off between number of segments and error can be solved by a maximum allowed error, a maximum allowed number of samples or both. These need to be specified by the user. However, both bottom up and top down segmentation allow for the use of a GAP statistic: a heuristic to determine the ideal number of segments, introduced by Tibshirani et al. [7]. The statistic is based on an *elbow* between the number of segments and the amount of error.

Also, none of the algorithms consider all possible segmentations; hence they risk finding a local optimum, w.r.t. the amount of error, which is significantly worse than the *global optimum*. Especially the bottom up and top down algorithms try to tackle this, but do not guarantee that the segmentation found is optimal.

This section will first discuss the sliding window, then the bottom up and finally the top down algorithm.



(a) GAP (b) Segmentation

Fig. 4. Segmentation of a time series using the top down algorithm. Figure (a) shows the optimal segmentation according to the GAP statistic discussed in section 2.2.2, it results in 8 segments. Colors are as in figure 1. In figure (b) the green curve is the $\varepsilon_{segmentation}$ of the time series with respect to the number of segments, the red curve is the $\varepsilon_{segmentation}$ for the random order time series and finally the blue curve is the GAP.

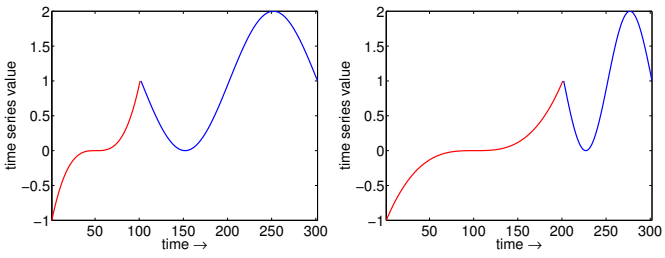


Fig. 5. Two time series, both consist of the same two shapes, yet the resolutions of the shapes is different for both.

Algorithm 1 Sliding window segmentation as defined in [5]

Require: Time series $T = (t_1, \ldots, t_n)$
Maximum error ε
Fitter $fit : \mathbb{R}^n \mapsto \mathbb{R}^n$
Minimum segment size *minsize*
Error function $error : \mathbb{R}^i \mapsto \mathbb{R}^i \mapsto \mathbb{R}$
Ensure: A set of breakpoints $B \subset [1 \ldots n]$ defining the segmentation of T

$B = [1]$ ▷ List with one element
$anchor = 1$
$upper = minsize + 1$
while $upper < n - minsize$ **do**
  $Sub = t_{anchor}, \ldots, t_{upper}$ ▷ A subsequence of T
  if $error(Sub, fit(Sub)) > \varepsilon$ **then**
    append $upper$ to B ▷ A new breakpoint is added
    $anchor = upper$
    $upper = anchor + minsize + 1$
  end if
end while
append $n+1$ to B

2.2.1 Sliding Window Segmentation

Sliding window segmentation is a simple online segmentation algorithm, which considers some small segment at the start of the time series and expands it with samples until the error between the representation and the segment exceeds some parameter ε. Then a new segment is started. See algorithm 1.

Figure 2 shows such a segmentation. Although the algorithm is simple, it is hard to determine which ε to use. A large ε may cause a segment to become infinitely long, whereas a small ε may create many small segments, which fit to noise. Moreover, the algorithm tends to get stuck in local minima, since the error may decrease as the segment grows.

The error of a segment $\mathbf{s}$ is defined as the euclidean distance between its fit and itself:

$$\varepsilon_{segment}(\mathbf{s}) = \|\mathbf{s} - fit(\mathbf{s})\|_2^2$$

Sliding window segmentation can be effective if the level of noise is known or can be estimated. The benefit of sliding window then is that there is no need to have the complete time series, the segmentation can be updated online as new samples become available.

Algorithm 2 Bottom up segmentation as defined in [5]

Require: Time series $T = (t_1, \ldots, t_n)$
Maximum error ε
Fitter $fit : \mathbb{R}^n \mapsto \mathbb{R}^n$
Minimum segment size *minsize*
Error function $error : \mathbb{R}^i \mapsto \mathbb{R}^i \mapsto \mathbb{R}$
Minimum number of segments *minsegments*
Ensure: A set of breakpoints $B \subset [1 \ldots n]$ defining the segmentation of T

$B = [1, 1 + minsize, 1 + 2 \cdot minsize, \ldots, N+1]$ ▷ List with a fine segmentation
$C = []$ ▷ Empty list for storing the merge costs
for $i = 1$ to $\#B - 2$ **do**
  $lower, upper = C[i], C[i+2] - 1$
  $Sub = t_{lower}, \ldots, t_{upper}$ ▷ A subsequence of T
  $C[i] = error(Sub, fit(Sub))$
end for
while $min(C) < \varepsilon$ and $\#B > minsegments$ **do**
  $idx = argmin(i, C[i])$
  delete $B[i+1], C[i+1]$
  $l_1, u_1, l_2, u_2 = C[i], C[i+1] - 1, C[i], C[i+1] - 1$
  $Sub_1 = t_{l_1}, \ldots, t_{u_1}$
  $Sub_2 = t_{l_2}, \ldots, t_{u_2}$
  $C[i] = error(Sub_1, fit(Sub_2))$
  $C[i+1] = error(Sub_2, fit(Sub_2))$
end while

2.2.2 Bottom up Segmentation

The bottom up approach starts with a fine grained segmentation, where all segments have small size and are of the same size. Then the algorithm calculates the error for each concatenation of two adjacent segments. The pair of segments in such a concatenation with minimal error is merged to one segment. This is done iteratively until either the error exceeds some defined threshold or the number of segments left has reached some defined bound. See algorithm 2.

However, bottom up segmentation does allow for applying a GAP-like statistic as proposed in [7]. If we define the error of a segmentation to be:

$$\varepsilon_{segmentation}(S) = \frac{1}{N} \sum_{\mathbf{s} \in S} \varepsilon_{segment}(\mathbf{s}),$$

where N is the total number of samples in the time series. The statistic used to find an optimal number of segments for $\mathbf{y}$ is:

$$\mathbf{y}_{ro} = \mathbf{y} \text{ in random order}$$
$$\mathbf{S}_{ro} = bottomup(\mathbf{y}_{ro}, ..., \#segments)$$
$$\mathbf{S} = bottomup(\mathbf{y}, ..., \#segments)$$
$$GAP(\#segments|\mathbf{y}) = \varepsilon_{segmentation}(\mathbf{S}) - \varepsilon_{segmentation}(\mathbf{S}_{ro}).$$

We use the segmentation of $\mathbf{y}$ in random order, to get the error of a random time series with the same distribution and histogram as the input time series. Finally, the optimal number of segments is determined by $\text{argmax}_{\#segments} GAP(\#segments|\mathbf{y})$.

Algorithm 3 Top Down segmentation as defined in [5]

Require: Time series $T = (t_1, \ldots, t_N)$
Maximum error ε
Fitter $fit : \mathbb{R}^n \mapsto \mathbb{R}^n$
Minimum segment size *minsize*
Error function $error : \mathbb{R}^i \mapsto \mathbb{R}^i \mapsto \mathbb{R}$
Maximum recursion depth k
Ensure: A set of breakpoints $B \subset [1 \ldots n]$ defining the segmentation of T

if $k = 0$ **then**
  $B = [0, N]$; **return**
end if
$\varepsilon_{min} = \infty, i_{min} = -1$
for $i = minsize + 1$ to $N - minsize$ **do**
  $Sub_1 = t_1, \ldots, t_{i-1}$
  $Sub_2 = t_i, \ldots, t_N$
  $\varepsilon_0 = error(T, fit(Sub_1); fit(Sub_2))$
  if $\varepsilon_0 < \varepsilon_{min}$ **then**
    $\varepsilon_{min} = \varepsilon_0; i_{min} = i$
  end if
end for
if $\varepsilon_{min} > \varepsilon$ **then**
  $B = [0, N]$; **return**
end if
$Sub_1 = t_1, \ldots, t_{i_{min}-1}$
$Sub_2 = t_{i_{min}}, \ldots, t_N$
$B_1 = topdown(Sub_1, \varepsilon, fit, minsize, error, k-1)$
$B_2 = topdown(Sub_2, \varepsilon, fit, minsize, error, k-1)$
delete first and last elements of B_1 and B_2;
add i_{min} to all elements of B_2
$B = [1; B_1; i_{min}; B_2, N]$

2.2.3 Top down Segmentation

Instead of merging a finer grained segmentation, top down segmentation starts off by assuming one segment. The algorithm tries each time point as a possible breakpoint and splits where $\varepsilon_{segmentation}$ is minimal. Then the algorithm repeats itself recursively for both segments, left and right of the breakpoint. See algorithm 3.

This algorithm can gain a huge speed up by only considering a subset of the possible break points, for example by considering $k = 10$ breakpoints at equidistant locations.

As a stop condition the algorithm can use error and recursion depth. The former has the disadvantage that the error needs to be chosen using some valid, possibly nonexistent, heuristic. The latter forces the segmentation to have a power of 2 segments. This also makes it harder to use a GAP statistic. A GAP statistic would need to select one of $1, 2, 4, 8, \ldots$, whilst the optimal number of segments may not be close to a power of 2. However, using the GAP statistic is possible. This paper uses this statistic to determine the number of segments for top down segmentation. Figure 3 shows such a segmentation.

3 Results

This section tests the three algorithms and the four bases using a synthetic time series. The next section will describe how these time series are created. Then the first test shows the stability of the parameter estimation by empirical calculation of the parameter variance. The stability of the parameters is measured w.r.t. noise and *padded segments*. The padded segments simulate segments which mostly coincide with the actual shape, but are slightly wrong.

Subsequently, the euclidean distance between segments using the parameters of one of the representations are measured for a synthetic set of segments. The distribution of these distances is shown for distances between similar shapes only differing in noise and resolution and different shapes, to test whether the representation has enough discriminating power.

Finally, this document will create random time series build from concatenated predefined shapes. This provides a time series for which an *ideal segmentation* is defined. Then the segmentation algorithms and the representations are tested w.r.t. accuracy. That is: how many of the time series samples can be classified as the correct shape?

3.1 Synthetic time series

So far, the term *shape* was mentioned several times. It is, however, not clearly defined yet. An important first note is that a shape is scale and resolution invariant: they are defined by function values calculated at equidistant samples for the domain $x \in [-1, 1]$, independent of the number of samples.

The following Matlab code illustrates the idea. The shapes $f1$ and $f2$ are considered the same, even though $f2$ has approximately twice the number of samples. Figure 5 shows two time series, built from these shapes.

```
% two shapes defined by two functions:
f = @(x) x.^3; g = @(x) sin(pi*x)+1;
% different shapes with different resolutions:
x1=linspace(-1,1,101); x2=linspace(-1,1,201);
f1=f(x1); g1=g(x1); f2=f(x2); g2=g(x2);
```

For the tests, there exist three sets of shapes, one is based on polynomial functions, the second is based on sine-functions, the third is the union of both. The sets are defined by f_i and g_i respectively:

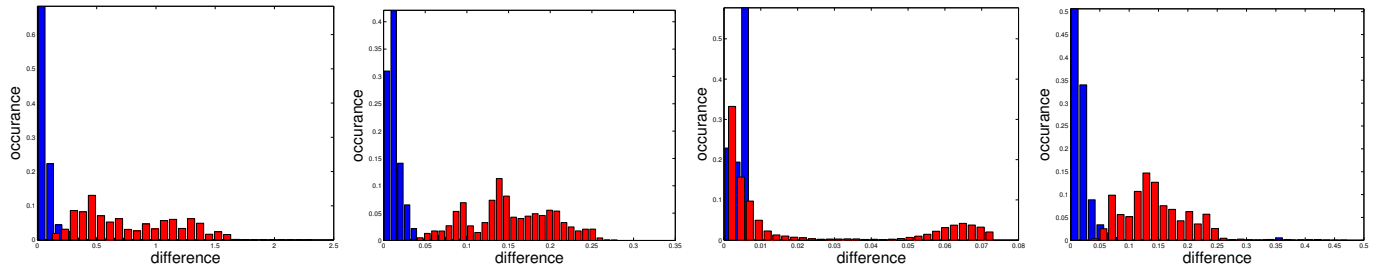
$$f_1(x) = 0, f_2(x) = x, f_3(x) = x^2, f_4 = x^3 \qquad \text{(polynomial)}$$
$$g_1(x) = 0, g_2(x) = \sin(\pi x) \qquad \text{(periodic)}$$

Each of these functions defines a shape of a certain type. The synthetic time series used in this paper consists of $N = 10001$ samples and 10 segments. However, the latter number is not used as input for the segmentations algorithm. Furthermore, each segment contains at least 50 samples, but the $10001 - 10 \cdot 50$ samples left are distributed randomly among the the segments. Each segment is created using a random basis function shown above, for the domain $[-1, 1]$. Furthermore, a basis function $f'(x) = f(x) + c$ is used, with c such that the leftmost sample of the previous segment equals the rightmost of the new one. Finally, white Gaussian noise, either $\mathcal{N}(0, 1)$ or $\mathcal{N}(0, 0.1)$, is added to the time series. A synthetic time series, based on the polynomial shapes, is shown in figure 8a. This figure also shows labels of the samples, which is discussed later in section 3.4.

The synthetic data that is used in this section is either a synthetic shape, which is based on one shape model, or a synthetic time series, which is a concatenation of shapes, as described in the previous paragraph.

3.2 Parameter estimation variance

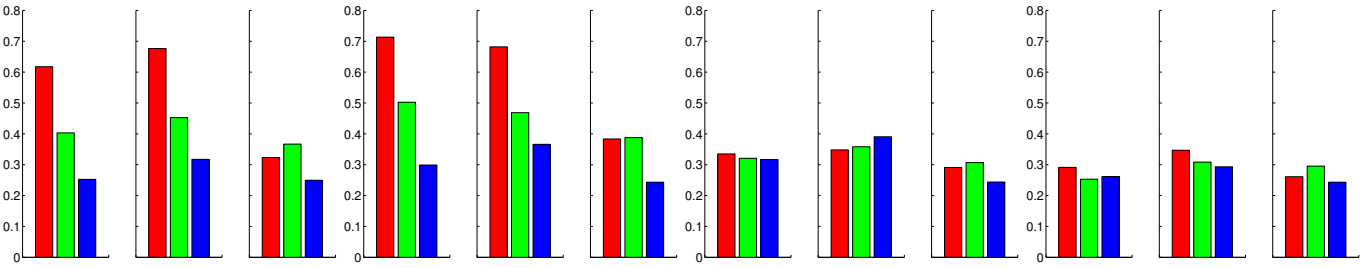
In order to verify the stability of the parameter, the shapes $0.3x^3 + x^2 - 1.5x + 3$ and $\sin(\pi x)$ were fit on the domain $[-1, 1]$ using the four representations. The results are in table 1 to table 4.The first two tables show the effect of white Gaussian noise on the variance of the parameters. The second two tables show the effect of prepending and appending the shapes on the variance of the parameters. This simulates the effect of a reasonable, yet inaccurate segmentation on the variance of the estimated parameters.



(a) Monomial basis (b) Chebyshev basis (c) Fourier basis (d) Haar basis

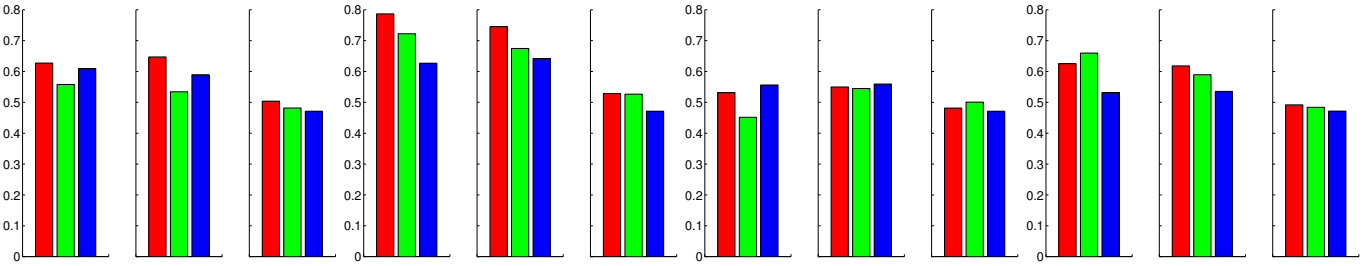
Fig. 6. Distribution of the euclidean distance between estimated parameters of two shapes, for all four bases. The shapes are taken from the third set (see section 3.1), with both polynomial and periodic shapes. The distances are measured between each pair out of a set which contains 30 shapes of each type. The shapes in the set have a random number of samples between 10^2 and 10^5, and contain additive white Gaussian noise $\mathcal{N}(0,1)$. The blue histogram shows the occurrence of distances with respect to shapes of the same type. The red histogram shows the occurrence of distances between shapes of different types. All representations use the first four basis functions.

Polonomial synthetic time series



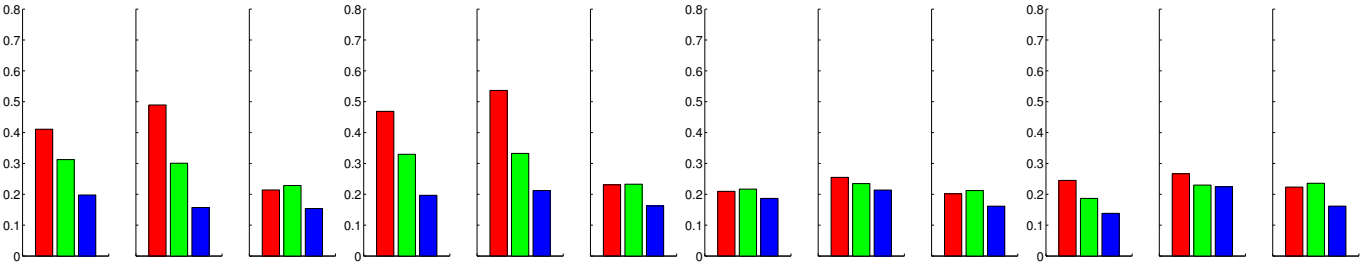
(a) Monomial basis (b) Chebyshev basis (c) Fourier basis (d) Haar basis

Periodic synthetic time series



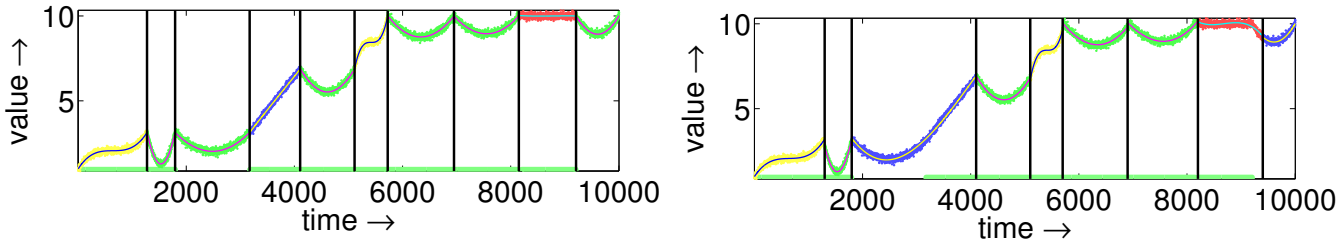
(e) Monomial basis (f) Chebyshev basis (g) Fourier basis (h) Haar basis

Both polynomial and periodic synthetic time series



(i) Monomial basis (j) Chebyshev basis (k) Fourier basis (l) Haar basis

Fig. 7. Each bar represents the mean *value* of 30 segmentations of synthetic time series (see section 3.1). This *value* is determined as the fraction of samples, 10001 for each signal, that were classified correctly (see section 3.4).The y-axis represents this fraction. Respectively, the red, green and blue bar represent bottom up, top down and sliding window segmentation. Furthermore, each subfigure (a)-(l) shows the results for 3 levels of noise, respectively: no noise, $\mathcal{N}(0,0.1)$ and $\mathcal{N}(0,1)$.



(a) Labelled time series

(b) Classification result

Fig. 8. In (a) a time series is shown, which is a concatenation of the polynomials 0 (red), x (blue), x^2 (green) and x^3 (yellow) on the domain $[-1,1]$ with additive white Gaussian noise $\mathcal{N}(0,0.1)$. In (b) a segmentation of that signal is shown, using bottom up and the gap statistic (see section 2.2.2). The vertical black lines mark the breakpoints. Each segment in (b) is classified against a one of the four polynomial shapes, and labelled with respect to the parameters which have minimal euclidean distance. The x-axis is marked green where the samples in (a) and (b) are labelled equally.

The three orthogonal bases outperform the monomial basis, in almost all cases the variance is at least 10 times smaller. This is shown in tables 3 and table 4.

parameter	$Var(\hat{\theta}_1)$	$Var(\hat{\theta}_2)$	$Var(\hat{\theta}_3)$	$Var(\hat{\theta}_4)$
monomial	0.072	0.435	1.422	3.290
Chebyshev	0.053	0.131	0.127	0.081
Fourier	0.048	0.061	0.076	0.069
Haar wavelets	0.048	0.056	0.243	0.060

Table 1. Table with variances of 4 parameters for different bases used in least square estimation. Based on a 100 signals with shape $0.3x^3 + x^2 - 1.5x + 3$ sampled at 101 points and additive white Gaussian noise $\mathcal{N}(0,1)$.

parameter	$Var(\hat{\theta}_1)$	$Var(\hat{\theta}_2)$	$Var(\hat{\theta}_3)$	$Var(\hat{\theta}_4)$
monomial	0.023	0.199	0.123	0.516
Chebyshev	0.008	0.009	0.012	0.013
Fourier	0.009	0.009	0.035	0.033
Haar wavelets	0.009	0.008	0.027	0.030

Table 2. Equivalent to table 1 for the shape $\sin(\pi x)$.

Also when prepending and appending, the orthonormal bases outperform the monomial basis. This is shown in the following two tables:

parameter	$Var(\hat{\theta}_1)$	$Var(\hat{\theta}_2)$	$Var(\hat{\theta}_3)$	$Var(\hat{\theta}_4)$
monomial	0.070	0.489	1.450	3.627
Chebyshev	0.049	0.131	0.124	0.082
Fourier	0.046	0.060	0.072	0.066
Haar wavelets	0.052	0.053	0.242	0.059

Table 3. These variances are taken from a similar sample as described in table 1. However a uniformly random amount (maximal 20 samples) of noise is prepended and appended to the 101 samples. These pre- and appended samples also contain additive white Gaussian noise $\mathcal{N}(0,1)$.

parameter	$Var(\hat{\theta}_1)$	$Var(\hat{\theta}_2)$	$Var(\hat{\theta}_3)$	$Var(\hat{\theta}_4)$
monomial	0.022	0.205	0.121	0.558
Chebyshev	0.008	0.009	0.011	0.013
Fourier	0.008	0.009	0.038	0.033
Haar wavelets	0.009	0.008	0.030	0.032

Table 4. Equivalent to table 4 for the shape $\sin(\pi x)$.

3.3 Parameter distance distribution

In order to capture the discriminating power of the representations, an experiment on a set of 30 shapes of each type was conducted. The distances between each pair of shapes from the set was calculated. Then the distances were split in two groups, those which are between shapes of the same shape model, for example $\sin(\pi x)$, and those which are from different shape models. These are shown as histograms in 6.

The histograms show that the Chebyshev and Haar bases perform best, since they have fewest overlap between the blue and red histograms. The Fourier basis, however, lacks discriminating power, since the blue and red histograms overlaps greatly.

3.4 Segmentation of synthetic data

Finally, all segmentation techniques and representations are tested using classification. The classification process is illustrated in figure 8. In (a) and (b), all samples are labelled according to the label of their segment. In (a), each segment is labelled by the synthetic creation of the time series (see section 3.1), whereas in (b) the labels are based on classification. This classification is described by algorithm 4.

Algorithm 4 Classification of a segmented time series

Require: Time series $T = (t_1, \ldots, t_n)$
Segments S
Fitter $fit : \mathbb{R}^n \mapsto \mathbb{R}^d$ ▷ maps to parameters, instead of an approximation of the time series
Shapes $\mathscr{S}$
Ensure: Returns a classification C which assigns for each segment in S a shape in $\mathscr{S}$.

```
C = Mapping S ↦ 𝒮
for s ∈ S do
    mindist = ∞; best = null
    for sh ∈ 𝒮 do
        if ‖fit(s) − fit(sh)‖₂² < mindist then
            mindist = ‖fit(s) − fit(sh)‖₂²; best = sh;
        end if
    end for
    append s ↦ best to C
end for
```

Finally, the performance of the segmentation can be defined as the fraction of samples for which the synthetic and the classification labels match. Figure 7 shows the results of 30 such classifications for each representation, segmentation algorithm and 3 levels of noise.

In almost all cases, the bottom up method outperforms the other two. This is in line with the conclusion in [4]. The classification test empirically shows that the polynomial approximations outperform the Haar wavelet and the Fourier based approximations. Moreover, the Chebyshev basis has slightly better results.

4 Conclusion and future work

This document has empirically compared three methods for segmenting and four representations. This work is similar to the work in [4], yet instead of only considering linear approximations, other bases were tested. This paper empirically shows that the polynomial representations perform best. Regarding the segmentation method, the classification shows that the bottom up technique performs best.

This paper also shows how the GAP statistic can be used to determine the optimal number of segments for bottom up and top down segmentation. Future work should at least include testing of all methods and representations on a real world dataset, like the IJdijk dataset.

Reproducible results statement In the interests of competitive scientific inquiry, all datasets and code used in this work are available for Matlab by emailing the first author.

References

[1] J. Cooley, P. Lewis, and P. Welch. The finite fourier transform. *Audio and Electroacoustics, IEEE Transactions on*, 17(2):77–85, 1969.

[2] E. Fuchs, T. Gruber, H. Pree, and B. Sick. Temporal data mining using shape space representations of time series. *Neurocomputing*, 74(1):379–393, 2010.

[3] C. F. Gauss. *Theory of the motion of the heavenly bodies moving about the sun in conic sections: a translation of Carl Frdr. Gauss" Theoria motus": With an appendix. By Ch. H. Davis*. Little, Brown and Comp., 1857.

[4] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[5] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *SIGKDD'02*, pages 102–111, 2002.

[6] H. W. Sorenson. Least-squares estimation: from gauss to kalman. *Spectrum, IEEE*, 7(7):63–68, 1970.

[7] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.