Engineering Science and Technology, an International Journal 20 (2017) 28–34

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full Length Article

# Recursive B-spline approximation using the Kalman filter

Jens Jauch *, Felix Bleimund, Stephan Rhode, Frank Gauterin

*Institute of Vehicle System Technology, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany*

ARTICLE INFO

*Article history:*
Received 14 April 2016
Revised 15 September 2016
Accepted 20 September 2016
Available online 28 November 2016

*Keywords:*
Recursive
B-spline
Approximation
Kalman filter
Shift
Online

ABSTRACT

This paper proposes a novel recursive B-spline approximation (RBA) algorithm which approximates an unbounded number of data points with a B-spline function and achieves lower computational effort compared with previous algorithms. Conventional recursive algorithms based on the Kalman filter (KF) restrict the approximation to a bounded and predefined interval. Conversely RBA includes a novel shift operation that enables to shift estimated B-spline coefficients in the state vector of a KF. This allows to adapt the interval in which the B-spline function can approximate data points during run-time.


## 1. Introduction

A B-spline function is a piecewise defined polynomial function with several beneficial properties such as numerical stability of computations, local effects of coefficient changes and built-in smoothness between neighboring polynomial pieces [2, chap. 1]. A common application of B-spline functions, curves and surfaces is fitting of data points. Fitting can either be interpolation or approximation. An interpolating B-spline function passes through the data points, whereas an approximating B-spline function minimizes the residuals between the function and the data but does not pass through the data points in general. The representation using B-splines is popular in computer-aided design, modeling and engineering as well as computer graphics for the geometry of curves, objects and surfaces [3]. It is also used for planning trajectories of computer controlled industrial machines [4] and robots [5,6].

Fitting B-spline functions can be determined by the weighted least squares (WLS) method. It is often used in offline applications, where all data points are available at once.

The Kalman filter (KF) is an established method for estimating the state of a dynamic system. Applications include tracking, navigation, sensor data fusion and process control [7, pp. 4f.]. The KF can be seen as a generalization of the recursive least squares (RLS) method [8, p. 129]. RLS can compute an approximating B-spline function recursively meaning that the approximation is updated with each new data point. This is desired in online applications, in which data points are observed one after another.

### 1.1. Problem statement

The value of a B-spline function is given by the sum of basis functions (B-splines) weighted with their corresponding coefficients. Each B-spline is only nonzero within a certain bounded interval which causes that the definition range of a B-spline is bounded as well. If the magnitude of the data points is not exactly known or changes over time, data points can be outside the definition range. Such data points cannot be taken into account. Thereby the problem arises that the approximation might not reflect the data anymore.

Publications concerning the recursive data approximation with a B-spline function have not addressed this issue but have assumed a constant definition range. For example, the approaches based on the KF in [9,10] require that the KF state vector contains all coefficients that are estimated during the whole approximation procedure. Therefore the number of coefficients has to be bounded and specified in advance. As a result, these algorithms can only approximate data points that are within the bounded definition range determined at the beginning.

### 1.2. Contribution

We propose a novel B-spline approximation (RBA) algorithm that solves the approximation problem iteratively using a KF.

* Corresponding author.
*E-mail addresses:* jens.jauch@kit.edu (J. Jauch), felix.bleimund@kit.edu (F. Bleimund), stephan.rhode@kit.edu (S. Rhode), frank.gauterin@kit.edu (F. Gauterin).
Peer review under responsibility of Karabuk University.
☆ The associated MATLAB source code can be downloaded from [1].

http://dx.doi.org/10.1016/j.jestch.2016.09.015
2215-0986/

RBA overcomes the current limitation of recursive algorithms based on the KF concerning the fixed approximation interval. The main contribution is to use the time update of the KF for a shift of estimated B-spline coefficients in the KF state vector in combination with a shift in the B-spline knot vector. The shift operation enables to shift the definition range such that it is always possible to take into account the latest data point for the approximation.

In online and offline applications, the shift operation allows to reduce the size of the state vector. As smaller state vector causes less computational effort. Table 1 displays the relevant features of different B-spline approximation methods.

### 1.3. Fitting algorithms for B-spline functions

Fitting B-spline functions can be computed by least squares (LS) methods [2,11,12]. With the standard formula in batch form, all data points have to be collected and then processed simultaneously. Therefore the number of data points $n$ needs to be bounded. The computation usually involves a Cholesky or QR factorization and requires $\mathcal{O}(n)$ operations if one takes advantage of the banded matrix structure [13, pp. 327–331]. Such algorithms are stated in [13, pp. 117–121] and [14, pp. 152–160]. With the LS algorithm each data point influences the result to the same extend. The WLS algorithm allows to weight measurements relative to each other [2, pp. 119–123].

In online applications an ever-growing amount of data is common. LS algorithms for online applications can be subdivided into two groups: First, growing memory LS algorithms apply a weighting that forgets old data exponentially. Second, sliding window LS algorithms discard old data completely and require only finite storage [15]. Sliding window LS and sliding window WLS algorithms are proposed in [15–18], respectively. Re-computing the fitting function from scratch with each new data point is costly. Rank update and rank downdate methods allow to re-use an already known factorization for an efficient update of a solution when observations have been added or deleted [19–21].

With WLS the bounded definition range of B-spline functions does not present a problem because the number and position of B-splines can be changed if the fitting function is re-computed from scratch. Moreover, rank modification methods support adding or deleting matrix columns [20]. This allows to extend, shrink or shift the definition range of the B-spline function.

Recursive algorithms such as RLS (see [8, pp. 84–88]) usually require less computational power than batch algorithms because they use smaller matrices and vectors whose sizes do not depend on the number of data points. The recursive computation is also referred to as progressive, iterative or sequential. In [22] fitting B-spline curves and surfaces are iteratively constructed based on the idea of profit and loss modification without solving a linear system. The authors of [23] build on the progressive and iterative approximation technique for B-spline curve and surface fitting and prove that the proposed algorithm achieves a least squares fit to the data points. A recursive algorithm for optimal smoothing B-spline surfaces inspired by the RLS method is presented in [24]. Algorithms that involve a KF are stated in [9,10]. All recursive approaches mentioned assume a constant definition range.

### 1.4. Structure of the data set

$\{(s_t, y_t)\}_{t=1,2,\dots,n}$ is a set of $n$ data points. $t$ denotes the time step at which data point $(s_t, \boldsymbol{y}_t)$ has been measured or observed. $s_t$ is the value of the independent variable $s$ at time step $t$. $\boldsymbol{y}_t = (y_{t,1}, y_{t,2}, \dots, y_{t,v}, \dots, y_{t,V_t})^\top$ is a vector of $V_t$ measurements $y$ that refer to $s_t$ and may come from different sensors. $^\top$ denotes the transpose operation. $V_t \in \mathbb{N}$ is allowed to be different for each $\boldsymbol{y}_t$. We assume that $V_t \ll n \forall t$. The vector of all measurements $\boldsymbol{y}$ is composed as follows:

$$\boldsymbol{y}^\top = (\underbrace{y_{1,1}, \dots, y_{1,V_1}}_{=:\boldsymbol{y}_1^\top}, \dots, \boldsymbol{y}_t^\top, \dots, \underbrace{y_{n,1}, \dots, y_{n,V_n}}_{=:\boldsymbol{y}_n^\top}) \tag{1}$$

### 1.5. Outline

The remainder of this article is structured as follows: In Section 2.1 we introduce a B-spline function definition in matrix form. Section 2.2 describes the WLS approach followed by the KF algorithm in Section 2.3. Section 2.4 presents the novel RBA algorithm. Its effectiveness is demonstrated in comparison with the WLS solution by numerical examples in Section 3. We summarize the characteristics of RBA and draw our conclusions in Section 4.

## 2. Methods

### 2.1. B-spline functions

A B-spline function is a piecewise defined function. Its value is given by the weighted sum of $J$ polynomial basis functions (B-splines) of degree $d$. The knot vector is $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \dots, \kappa_{J+d+1})$. We assume strictly increasing knot values $(\kappa_k < \kappa_{k+1}, k = 1, 2, \dots, J+d)$. $\boldsymbol{\kappa}$ and $d$ determine the number and shape of B-splines. The $j$-th B-spline $b_j(s), j = 1, 2, \dots, J$ is positive only for $s \in (\kappa_j, \kappa_{j+d+1})$ and zero elsewhere [2, pp. 37–42].

The following definitions originate from [2, pp. 47–50 & 65–70]: Let $[\kappa_\mu, \kappa_{\mu+1})$ be a spline interval and let $\mu$ denote the spline interval index with $d+1 \leqslant \mu \leqslant J$. For $s \in [\kappa_\mu, \kappa_{\mu+1})$, the B-splines $b_j(s), j = \mu - d, \dots, \mu$ can be nonzero. Their values for a specific $s \in [\kappa_\mu, \kappa_{\mu+1})$ can be summarized in the B-spline vector $\boldsymbol{b}_{\mu,d}(s) = (b_{\mu-d}(s), b_{\mu-d+1}(s), \dots, b_\mu(s)) \in \mathbb{R}^{1\times(d+1)}$ which can be computed according to (2):

$$\boldsymbol{b}_{\mu,d}(s) = \underbrace{\boldsymbol{B}_{\mu,1}(s)}_{\in\mathbb{R}^{1\times2}} \underbrace{\boldsymbol{B}_{\mu,2}(s)}_{\in\mathbb{R}^{2\times3}} \dots \underbrace{\boldsymbol{B}_{\mu,\delta}(s)}_{\in\mathbb{R}^{\delta\times(\delta+1)}} \dots \underbrace{\boldsymbol{B}_{\mu,d}(s)}_{\in\mathbb{R}^{d\times(d+1)}} \tag{2}$$

The B-spline matrix $\boldsymbol{B}_{\mu,\delta}(s) \in \mathbb{R}^{\delta\times(\delta+1)}$ is defined for each $\delta \in \mathbb{N}$ with $\delta \leqslant d$ and given by

$$\boldsymbol{B}_{\mu,\delta}(s) = \begin{bmatrix} \frac{\kappa_{\mu+1}-s}{\kappa_{\mu-1}-\kappa_{\mu+1-\delta}} & \frac{s-\kappa_{\mu+1-\delta}}{\kappa_{\mu+1}-\kappa_{\mu+1-\delta}} & 0 & \dots & 0 \\ 0 & \frac{\kappa_{\mu+2}-s}{\kappa_{\mu-2}-\kappa_{\mu+2-\delta}} & \frac{s-\kappa_{\mu+2-\delta}}{\kappa_{\mu+2}-\kappa_{\mu+2-\delta}} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \dots \\ 0 & 0 & \dots & \frac{\kappa_{\mu+\delta}-s}{\kappa_{\mu+\delta}-\kappa_\mu} & \frac{s-\kappa_\mu}{\kappa_{\mu+\delta}-\kappa_\mu} \end{bmatrix}. \tag{3}$$

**Table 1**
Comparison of different B-spline approximation methods.

| Feature | WLS (single call) | WLS (multiple calls) | RLS/KF | RBA |
|---|---|---|---|---|
| Number of processable data points n | bounded | unbounded | unbounded | unbounded |
| Time complexity | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ |
| Approximation interval | fixed | variable | fixed | variable |
| Determination of total number of coefficients being estimated | at beginning | during run-time | at beginning | during run-time |

The B-spline function $f : \mathcal{D} \to \mathbb{R}, s \mapsto f(s)$ has the definition range $\mathcal{D} = [\kappa_{d+1}, \kappa_{J+1})$. For $s \in [\kappa_\mu, \kappa_{\mu+1})$, the B-spline function is given by

$$f(s) = \boldsymbol{b}_{\mu,d}(s)\boldsymbol{x}_{\mu,d} \tag{4}$$

with coefficient vector

$$\boldsymbol{x}_{\mu,d} = (x_{\mu-d}, x_{\mu-d+1}, \ldots, x_\mu)^\top. \tag{5}$$

Fig. 1 illustrates the construction of a B-spline vector.

The B-spline function is $d-1$ times continuously differentiable. For $r \in \mathbb{N}_0$, the $r$-th derivative $\frac{\partial^r}{\partial s^r} f(s)$ of the B-spline function with respect to $s$ is given by

$$\frac{\partial^r}{\partial s^r} f(s) = \frac{\partial^r}{\partial s^r} \boldsymbol{b}_{\mu,d}(s)\boldsymbol{x}_{\mu,d} \tag{6}$$

with B-spline vector

$$\frac{\partial^r}{\partial s^r}\boldsymbol{b}_{\mu,d}(s) = \begin{cases} \frac{d!}{(d-r)!}\boldsymbol{B}_{\mu,1}(s)\ldots\boldsymbol{B}_{\mu,d-r}(s)\boldsymbol{B}'_{\mu,d-r+1}\ldots\boldsymbol{B}'_{\mu,d}, & \text{if } r \leqslant d \\ \boldsymbol{0}_{1\times(d+1)}, & \text{else} \end{cases} \tag{7}$$

$\boldsymbol{0}_{1\times(d+1)}$ denotes a $1 \times (d+1)$ zero matrix. The matrix $\boldsymbol{B}'_{\mu,\delta} \in \mathbb{R}^{\delta\times(\delta+1)}$ is obtained by differentiating all entries in $\boldsymbol{B}_{\mu,\delta}(s)$ with respect to $s$:

$$\boldsymbol{B}'_{\mu,\delta} = \begin{bmatrix} \frac{-1}{\kappa_{\mu+1}-\kappa_{\mu+1-\delta}} & \frac{1}{\kappa_{\mu+1}-\kappa_{\mu+1-\delta}} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \frac{-1}{\kappa_{\mu+\delta}-\kappa_\mu} & \frac{1}{\kappa_{\mu+\delta}-\kappa_\mu} \end{bmatrix} \tag{8}$$

### 2.2. Weighted least squares

The linear weighted least squares (WLS) method estimates the constant state vector $\boldsymbol{x} \in \mathbb{R}^{L\times 1}$ of a linear system

$$\boldsymbol{y} = \mathbf{C}\boldsymbol{x} + \boldsymbol{\upsilon}. \tag{9}$$

$\boldsymbol{y} \in \mathbb{R}^{N\times 1}$ is the vector of measurements and $\mathbf{C}$ denotes the measurement matrix that relates $\boldsymbol{x}$ to $\boldsymbol{y}$. The measurement noise $\boldsymbol{\upsilon} \in \mathbb{R}^{N\times 1}$ is assumed to be an uncorrelated white noise process with mean zero. This implies that the covariance matrix of measurement noise $\mathbf{R}$ is a diagonal matrix and $\mathbf{R}_{i,i}$, $i = 1, \ldots, N$ is the variance of measurement $y_i$ which can differ from the variances of other measurements [8]. The assumptions for $\{\upsilon\}$ can be generalized to a correlated noise process. This is termed generalized linear model [25, p. 143]. Then $\mathbf{R}$ is a positive definite matrix [13, p. 374]. Furthermore, [13] states LS algorithms for nonlinear problems.

The linear WLS estimate $\hat{\boldsymbol{x}}$ minimizes the sum of squared errors between the measurements $\boldsymbol{y}$ and the vector $\mathbf{C}\boldsymbol{x}$ which are weighted with the reciprocals of the variances of the measurements:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} (\boldsymbol{y} - \mathbf{C}\boldsymbol{x})^\top \mathbf{R}^{-1} (\boldsymbol{y} - \mathbf{C}\boldsymbol{x}) \tag{10}$$

The solution to optimization problem (10) is given by the closed-form estimator

$$\hat{\boldsymbol{x}} = (\mathbf{C}^\top\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{C}^\top\mathbf{R}^{-1}\boldsymbol{y} \text{ [8]}. \tag{11}$$

From (4) follows that the value of a B-spline function is a linear combination of its coefficients. Therefore WLS can be used to determine the coefficients such that the function approximates the set of data points defined in Section 1.4. Then $\mathbf{C}$ is a $(\sum_{t=1}^{n} V_t) \times J$ matrix because $\boldsymbol{y}$ comprises $\sum_{t=1}^{n} V_t$ scalar components $y_{t,v}$, $t = 1, \ldots, n, v = 1, \ldots, V_t$ (c.f. (1)) and there are $J$ B-splines. $y_{t,v}$ is the $\tilde{v}$-th component of $\boldsymbol{y}$ ($\tilde{v} = \sum_{\hat{t}=1}^{t-1} V_{\hat{t}} + v$) and provides information about $\frac{\partial^r}{\partial s^r} f(s_t)$ with $s_t \in [\kappa_\mu, \kappa_{\mu+1})$ and an $r \in \mathbb{N}_0$. The $\tilde{v}$-th row $\mathbf{C}$ is given by $\mathbf{C}_{\tilde{v};1,\ldots,J} = \boldsymbol{c}$ with

$$\boldsymbol{c} = (\boldsymbol{0}_{1\times(\mu-(d+1))}, \frac{\partial^r}{\partial s^r}\boldsymbol{b}_{\mu,d}(s_t), \boldsymbol{0}_{1\times(J-\mu)}). \tag{12}$$

### 2.3. Kalman filter

The linear Kalman filter (KF) estimates the state vector $\boldsymbol{x}_t \in \mathbb{R}^{L\times 1}$ of a linear time-discrete system

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{A}}_t\boldsymbol{x}_{t-1} + \boldsymbol{\mathcal{B}}_t\boldsymbol{u}_t + \boldsymbol{\omega}_t \qquad \text{(state equation)} \tag{13}$$

$$\boldsymbol{y}_t = \boldsymbol{\mathcal{C}}_t\boldsymbol{x}_t + \boldsymbol{\upsilon}_t \qquad \text{(measurement equation)} \tag{14}$$

where $t \in \mathbb{N}$ denotes the time step. $\boldsymbol{\mathcal{A}}_t$ is the state transition matrix that relates $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$, $\boldsymbol{u}_t \in \mathbb{R}^{M\times 1}$ an input signal vector with known influence on $\boldsymbol{x}_t$ and $\boldsymbol{\mathcal{B}}_t$ the input matrix that relates $\boldsymbol{u}_t$ to $\boldsymbol{x}_t$. The vector of measurements is denoted by $\boldsymbol{y}_t \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{\mathcal{C}}_t$ is the measurement matrix that relates $\boldsymbol{x}_t$ to $\boldsymbol{y}_t \cdot \boldsymbol{\omega}_t \in \mathbb{R}^{N\times 1}$ is the process noise with covariance matrix $\boldsymbol{\mathcal{Q}}_t$ and $\boldsymbol{\upsilon}_t \in \mathbb{R}^{N\times 1}$ is the measurement noise with covariance matrix $\boldsymbol{\mathcal{R}}_t$. Both $\{\boldsymbol{\omega}_t\}$ and $\{\boldsymbol{\upsilon}_t\}$ are uncorrelated white noise processes with mean zero which implies that $\boldsymbol{\mathcal{Q}}_t$ and $\boldsymbol{\mathcal{R}}_t$ are diagonal matrices [8, p. 124].

The KF consists of a sequel of equations, which are computed for each time step and summarized in Algorithm 1 in which $\boldsymbol{I}$ denotes the identity matrix with appropriate dimensions. The KF performs a time update followed by a measurement update. During the time update, the state estimate is updated based on the knowledge about the system specified by (13). Both the a priori estimate $\hat{\boldsymbol{x}}_t^-$ and the covariance $\mathcal{P}_t^-$ of the a priori estimation error are calculated. During the measurement update, the Kalman gain $\mathcal{K}_t$ is computed and used together with the information provided by measurement $\boldsymbol{y}_t$ for the calculation of the corresponding a posteriori quantities $\hat{\boldsymbol{x}}_t^+$ and $\mathcal{P}_t^+$ [8, pp. 124–129]. KF generalizations

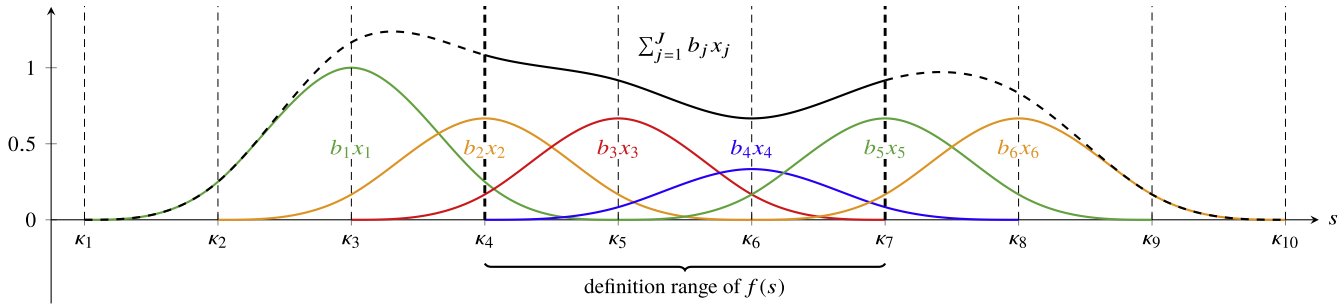


**Fig. 1.** Construction of a cubic ($d = 3$) B-spline function: Equidistant knots $\kappa_1, \kappa_2, \ldots, \kappa_{10}$ (indicated by vertical straight lines) and $J = 6$ resulting B-splines $b_j$, $j = 1, 2, \ldots, J$ weighted with coefficients $x_1 = 1.5, x_4 = 0.5, x_2 = x_3 = x_5 = x_6 = 1$. The black line indicates the sum of all weighted B-splines ($\sum_{j=1}^{J} b_j(s)x_j$). The B-spline function is given by $f(s) = \sum_{j=1}^{J} b_j(s)x_j$ but only defined over $[\kappa_4, \kappa_7)$ (solid part of the black line.)

for correlated or colored noise processes are stated in [8, pp. 183 – 193].

If the state vector $\boldsymbol{x}_t$ is constant, then $\boldsymbol{\mathcal{A}}_t = \boldsymbol{I}$, $\boldsymbol{\omega}_t = \boldsymbol{0}$ and $\boldsymbol{u}_t = \boldsymbol{0}$, where $\boldsymbol{0}$ is the zero matrix with appropriate dimensions. In this case, the time update is redundant and the KF simplifies to the RLS algorithm [8, p. 129]. An application of RLS to the data approximation problem with a polynomial is described in [8, pp. 92–93].

In the known recursive approaches to the data approximation problem, the KF estimates a state vector $\boldsymbol{x}_t$ that comprises all required coefficients. Then $\boldsymbol{x}_t$ is constant, only the estimation $\hat{\boldsymbol{x}}_t$ may change and therefore RLS is sufficient for solving the problem. In contrast, the novel algorithm proposed in Section 2.4 takes advantage of the time update that only the KF provides.

**Algorithm 1.** Kalman Filter

**Input**: $\hat{\boldsymbol{x}}^+_{t-1}, \boldsymbol{\mathcal{P}}^+_{t-1}, \boldsymbol{u}_t, \boldsymbol{y}_t, \boldsymbol{\mathcal{A}}_t, \boldsymbol{\mathcal{B}}_t, \boldsymbol{\mathcal{C}}_t, \boldsymbol{\mathcal{Q}}_t, \boldsymbol{\mathcal{R}}_t$

/* Time update */

1 $\hat{\boldsymbol{x}}^-_t \leftarrow \boldsymbol{\mathcal{A}}_t \hat{\boldsymbol{x}}^+_{t-1} + \boldsymbol{\mathcal{B}}_t \boldsymbol{u}_t$

2 $\boldsymbol{\mathcal{P}}^-_t \leftarrow \boldsymbol{\mathcal{A}}_t \boldsymbol{\mathcal{P}}^+_{t-1} \boldsymbol{\mathcal{A}}_t^\top + \boldsymbol{\mathcal{Q}}_t$

/* Measurement update */

3 $\boldsymbol{\mathcal{K}}_t \leftarrow \boldsymbol{\mathcal{P}}^-_t \boldsymbol{\mathcal{C}}_t^\top (\boldsymbol{\mathcal{C}}_t \boldsymbol{\mathcal{P}}^-_t \boldsymbol{\mathcal{C}}_t^\top + \boldsymbol{\mathcal{R}}_t)^{-1}$

4 $\hat{\boldsymbol{x}}^+_t \leftarrow \hat{\boldsymbol{x}}^-_t + \boldsymbol{\mathcal{K}}_t (\boldsymbol{y}_t - \boldsymbol{\mathcal{C}}_t \hat{\boldsymbol{x}}^-_t)$

5 $\boldsymbol{\mathcal{P}}^+_t \leftarrow (\boldsymbol{I} - \boldsymbol{\mathcal{K}}_t \boldsymbol{\mathcal{C}}_t) \boldsymbol{\mathcal{P}}^-_t (\boldsymbol{I} - \boldsymbol{\mathcal{K}}_t \boldsymbol{\mathcal{C}}_t)^\top + \boldsymbol{\mathcal{K}}_t \boldsymbol{\mathcal{R}}_t \boldsymbol{\mathcal{K}}_t^\top$

**Output**: $\hat{\boldsymbol{x}}^+_t, \boldsymbol{\mathcal{P}}^+_t$

### 2.4. Recursive B-spline approximation

The novel recursive B-spline approximation (RBA) algorithm computes an approximating B-spline function $f(s)$ of degree $d$ for the set of data points from Section 1.4 iteratively using the KF. Algorithm 2 summarizes the calculations. $I \in \mathbb{N}$ denotes the constant number of spline intervals of $f(s)$. The KF state estimate $\hat{\boldsymbol{x}}_t = (\hat{x}_{t_1}, \hat{x}_{t_2}, \ldots \hat{x}_{t_J})^\top$ comprises $J = d + I$ components which are the estimated coefficients of $f(s)$. The knot vector $\boldsymbol{\kappa}_t = (\kappa_{t_1}, \kappa_{t_2}, \ldots, \kappa_{t_K})$ for time step $t$ has to contain $K = J + d + 1$ knots. $\mathcal{D}_t = [\kappa_{t_{d+1}}, \kappa_{t_{J+1}})$ is the definition range of $f(s)$ at $t$.

#### 2.4.1. Initialization

We initialize $\hat{\boldsymbol{x}}_t$ with $\hat{\boldsymbol{x}}^+_0 = \bar{x}\boldsymbol{1}_{J\times 1}$, where $\boldsymbol{1}_{J\times 1}$ denotes a $J \times 1$ matrix of ones and $\bar{x}$ a scalar quantity of the magnitude of measurements $y_{t,v}$ that refer to $\frac{\partial^r}{\partial s^r} f(s)$ with $r = 0$.

The covariance matrix of a posteriori estimation error $\boldsymbol{\mathcal{P}}^+$ is initialized with $\boldsymbol{\mathcal{P}}^+_0 = \bar{p}\boldsymbol{I}_{J\times J}$, where $\boldsymbol{I}_{J\times J}$ is a $J \times J$ identity matrix. The scalar $\bar{p}$ should be chosen large (e.g. $10^4$) because then $\hat{\boldsymbol{x}}_t$ will quickly deviate from its initial value $\hat{\boldsymbol{x}}^+_0$ in such a way that $f(s)$ approximates the data points. If the elements in $\boldsymbol{\mathcal{P}}^-_t$ are small, this signals to the KF that the state estimate $\hat{\boldsymbol{x}}^-_t$ is very certain and therefore it will hardly be updated using the measurements. If the KF updates $\hat{\boldsymbol{x}}^+_t$ as intended, the elements in $\boldsymbol{\mathcal{P}}^+_t$ become smaller as t increases.

In the long-run, $\boldsymbol{\mathcal{P}}^-_t$ is strongly influenced by the covariance matrix of process noise $\boldsymbol{\mathcal{Q}}_t$ because of line 2 of Algorithm 1. If the elements in $\boldsymbol{\mathcal{Q}}_t$ are large, the elements in $\boldsymbol{\mathcal{P}}^-_t$ remain large too. This can lead to volatile states estimates $\hat{\boldsymbol{x}}_t$ that do not converge to a certain value. Then $f(s)$ will not approximate the data points well. For that reason we choose $\boldsymbol{\mathcal{Q}}_t = \bar{q}\boldsymbol{I}_{J\times J}$ with a very small positive value $\bar{q}$ (e.g. $\bar{q} = 10^{-12}$).

#### 2.4.2. Time update with shift operation

RBA compares the knot vector $\boldsymbol{\kappa}_{t-1}$ with $s_t$ in order to determine whether $s_t \in \mathcal{D}_{t-1}$. If necessary, a shift of $\mathcal{D}_{t-1}$ is performed during the time update of the KF such that $s_t \in \mathcal{D}_t$. The variable s indicates the shift direction of $\mathcal{D}_{t-1}$ and the number of positions by which elements of $\boldsymbol{\kappa}_{t-1}$ are shifted. $\sigma > 0$ means a right shift of $\mathcal{D}_{t-1}$, $\sigma < 0$ a left shift of $\mathcal{D}_{t-1}$ and for $\sigma = 0$ no shift is performed because $s_t \in \mathcal{D}_{t-1}$. Algorithm 2 computes $\sigma$ from line 5 to line 18.

For example, assume $d = 3$, $I = 3$ and $\boldsymbol{\kappa}_{t-1} = (1, 2, \ldots, 10)$, then $\mathcal{D}_{t-1} = [4, 7)$ . If $s_t = 8.5$, we need two additional knots to be able to perform a right shift by two elements ($\sigma = 2$). With 11 and 12 as additional knots we get $\boldsymbol{\kappa}_t = (3, 4, \ldots, 12)$ and hence $s_t \in \mathcal{D}_t = [6, 9)$.

Algorithm 2 distinguishes between $\sigma \geqslant 0$ and $\sigma < 0$. It assumes that the $|\sigma|$ additional knots are the $\sigma$ last components of the knot vector $\bar{\boldsymbol{\kappa}}_t = (\bar{\kappa}_{t_1}, \bar{\kappa}_{t_2}, \ldots, \bar{\kappa}_{t_K})$ in case of $\sigma > 0$ and that they are the -$\sigma$ first components of $\bar{\boldsymbol{\kappa}}_t$ in case of $\sigma < 0$.

*Case 1:* $\sigma \geqslant 0$.

The new knot vector is

$$\boldsymbol{\kappa}_t \leftarrow (\kappa_{t-1_{\sigma+1}}, \kappa_{t-1_{\sigma+2}}, \ldots, \kappa_{t-1_K}, \bar{\kappa}_{t_{K-\sigma+1}}, \bar{\kappa}_{t_{K-\sigma+2}}, \ldots, \bar{\kappa}_{t_K}). \tag{15}$$

The elements in $\hat{\boldsymbol{x}}^+_{t-1}$ are shifted by line 1 of Algorithm 1 using the state transition matrix

$$\boldsymbol{\mathcal{A}}_t \in \mathbb{R}^{J\times J} \text{ with } \boldsymbol{\mathcal{A}}_{t_{g;h}} = \begin{cases} 1, & \text{if } h = g + \sigma \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

$\hat{\boldsymbol{x}}^-_t \leftarrow \boldsymbol{\mathcal{A}}_t \hat{\boldsymbol{x}}^+_{t-1}$ updates the old estimate $\hat{\boldsymbol{x}}^+_{t-1}$ to

$$\hat{\boldsymbol{x}}^-_t = (\hat{\boldsymbol{x}}_{t-1_{\sigma+1}}, \hat{x}_{t-1_{\sigma+2}}, \ldots, \hat{\boldsymbol{x}}_{t-1_{J-\sigma}}, \boldsymbol{0}_{1\times\sigma})^T \tag{17}$$

With the second part of the instruction ($\hat{\boldsymbol{x}}^-_t \leftarrow \hat{\boldsymbol{x}}^-_t + \boldsymbol{\mathcal{B}}_t \boldsymbol{u}_t$), input matrix $\boldsymbol{\mathcal{B}}_t = \boldsymbol{I}_{J\times J}$ and input signal vector $\boldsymbol{u}_t = (\boldsymbol{0}_{1\times(J-\sigma)}, \bar{x}\boldsymbol{1}_{1\times\sigma})^\top$ we can have arbitrary initial estimates $\bar{x}$ in $\hat{\boldsymbol{x}}^-_t$:

$$\hat{\boldsymbol{x}}^-_t = (\hat{x}_{t-1_{\sigma+1}}, \hat{x}_{t-1_{\sigma+2}}, \ldots, \hat{x}_{t-1_{J-\sigma}}, \bar{x}\boldsymbol{1}_{1\times\sigma})^\top \tag{18}$$

$\boldsymbol{\mathcal{P}}^+_{t-1}$ is updated during the time update as well by line 2 of Algorithm 1. The first part of the instruction ($\boldsymbol{\mathcal{P}}^-_t \leftarrow \boldsymbol{\mathcal{A}}_t \boldsymbol{\mathcal{P}}^+_{t-1} \boldsymbol{\mathcal{A}}_t^\top$) leads to $\boldsymbol{\mathcal{P}}^+_{t_{1,\ldots,J-\sigma;1,\ldots,J-\sigma}} \leftarrow \boldsymbol{\mathcal{P}}^-_{t-1_{\sigma+1,\ldots,J;\sigma+1,\ldots,J}}$ and all elements in the rows or columns $J-\sigma+1, J-\sigma+2, \ldots, J$ of $\boldsymbol{\mathcal{P}}^-_t$ equal zero. Especially zeros on the main diagonal prevent that $\hat{x}_{t_{J-\sigma+1}}, \hat{x}_{t_{J-\sigma+2}}, \ldots, \hat{x}_{t_J}$ become different from the initial value $\bar{x}$. Large values in $\boldsymbol{\mathcal{P}}^-_t$ can be achieved by the second part of the instruction, $\boldsymbol{\mathcal{P}}^-_t \leftarrow \boldsymbol{\mathcal{P}}^-_t + \boldsymbol{\mathcal{Q}}_t$. During a right shift we overwrite the elements $\boldsymbol{\mathcal{Q}}_{t_{J-\sigma+1;J-\sigma+1}}, \boldsymbol{\mathcal{Q}}_{t_{J-\sigma+2;J-\sigma+2}}, \ldots, \boldsymbol{\mathcal{Q}}_{t_{J;J}}$ with $\bar{p}$ in line 24 of Algorithm 2 in order to get large entries $\boldsymbol{\mathcal{P}}^-_{t_{J-\sigma+1;J-\sigma+1}}, \boldsymbol{\mathcal{P}}^-_{t_{J-\sigma+2;J-\sigma+2}}, \ldots, \boldsymbol{\mathcal{P}}^-_{t_{J;J}}$. Fig. 2 depicts different states of $\boldsymbol{\mathcal{P}}^+$ and $\boldsymbol{\mathcal{P}}^-$, respectively.

*Case 2:* $\sigma < 0$.

The new knot vector is

$$\boldsymbol{\kappa}_t \leftarrow (\bar{\kappa}_{t_1}, \bar{\kappa}_{t_2}, \ldots, \bar{\kappa}_{t_{-\sigma}}, \kappa_{t-1_1}, \kappa_{t-1_2}, \ldots, \kappa_{t-1_{K+\sigma}}) \tag{19}$$

We choose $\boldsymbol{\mathcal{A}}_t$ as in (16), $\boldsymbol{\mathcal{B}}_t = \boldsymbol{I}_{J\times J}, \boldsymbol{u}_t = (\bar{x}\boldsymbol{1}_{1\times(-\sigma)}, \boldsymbol{0}_{1\times(J+\sigma)})^\top$ and set the elements $\boldsymbol{\mathcal{Q}}_{t_{1;1}}, \boldsymbol{\mathcal{Q}}_{t_{2;2}}, \ldots, \boldsymbol{\mathcal{Q}}_{t_{-\sigma;-\sigma}}$ to $\bar{p}$.

The shift operation is the distinguishing feature of RBA compared to the algorithms in [9,10]. Due to the shift operation, the total number of coefficients that will be estimated during the application of RBA does neither have to be known in advance nor to be bounded. Using $\boldsymbol{\kappa}_t$ and $\hat{\boldsymbol{x}}^+_t$ we can evaluate the determined B-spline function $f(s)$ at any $s \in [\kappa_{t_{d+1}}, \kappa_{t_{d+I+1}})$.

By shifting the entries in $\boldsymbol{\kappa}_{t-1}$, $\hat{\boldsymbol{x}}^+_{t-1}$ and $\boldsymbol{\mathcal{P}}^+_{t-1}$, the required storage is held constant. In Algorithm 2 we choose $|\sigma|$ just large enough that the current data point can be taken into account during the measurement update because the shift operation comes at the cost that parts of an already computed approximation are forgotten unless the values of $\hat{\boldsymbol{x}}^+_{t-1}$ and $\boldsymbol{\mathcal{P}}^+_{t-1}$ are saved separately from Algorithm 2 before they are removed from the vectors or matrix, respectively.

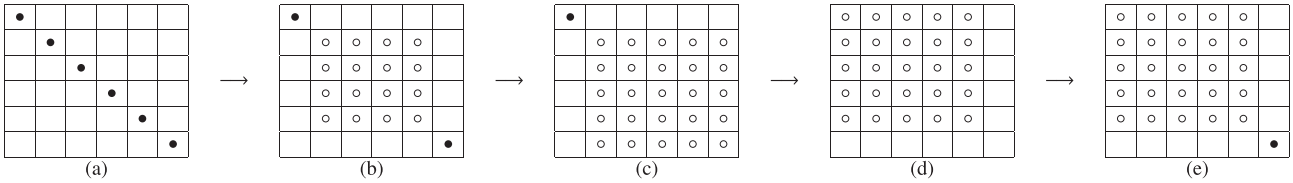


**Fig. 2.** Changes of the elements in $\mathcal{P}^+$ and $\mathcal{P}^-$, respectively: $d = 3$ and $I = 3$ leads to a $6 \times 6$ matrix $\mathcal{P}^+$. We initialize the diagonal values of $\mathcal{P}_0^+$ with a large positive value indicated by • as depicted in (a). All other elements of $\mathcal{P}_0^+$ are initialized with zeros indicated by empty cells. After some data points in the second spline interval have been processed, different comparatively small values denoted by ∘ are in the submatrix $\mathcal{P}^+_{2,\dots,5;2,\dots,5}$. (b). After data points that fall into the third spline interval have been processed, only the elements in the first row and column of $\mathcal{P}^+$ have still their initialization values (c). If only the first part of the update instruction is executed during a right shift by one element ($\sigma = 1$), the elements in the last row and column of $\mathcal{P}^-$ become zero (d). With the second part of the instruction, these elements can be set to nonzero values. For $\mathcal{Q}_{6;6} =$ • and all other elements of $\mathcal{Q}$ equal zero, we obtain matrix (e).

#### 2.4.3. Measurement update

During the measurement update, the information provided by $(s_t, y_t)$ is used to update $f(s)$. With the covariance matrix of measurement noise $\mathcal{R}_t$, different components of the measurement vector $\boldsymbol{y}_t$ can be weighted relative to each other. $\mathcal{R}_t \in \mathbb{R}^{V_t \times V_t}$ is a diagonal matrix with positive elements on its diagonal. The smaller an entry $\mathcal{R}_{t_{v;v}}$ is, the greater is the effect of the $v$-th component of the measurement error $(\boldsymbol{y}_t - \mathcal{C}_t \hat{\boldsymbol{x}}_t^-)$ on $\hat{\boldsymbol{x}}_t^-$.

The measurement matrix $\mathcal{C}_t$ is a $V_t \times J$ matrix. $y_{t,v}$ is the $v$-th component of $\boldsymbol{y}_t$ and provides information about $\frac{\partial^r}{\partial s^r} f(s_t)$ with $s_t \in [\kappa_\mu, \kappa_{\mu+1})$ and an $r \in \mathbb{N}_0$. The $v$-th row of $\mathcal{C}_t$ is given by

$$\mathcal{C}_{t_{v;1,\dots,J}} = \boldsymbol{c} \tag{20}$$

with $\boldsymbol{c}$ from (12).

According to $\mathcal{C}_t$, $(s_t, y_t)$ influences only the estimates $\hat{x}_{t_{\mu-d}}, \hat{x}_{t_{\mu-d+1}}, \dots, \hat{x}_{t_\mu}$. However, other estimates can still be updated by the KF using the information stored in $\mathcal{P}_t^-$.

**Algorithm 2:** Recursive B-Spline Approximation

**Input**: $\boldsymbol{\kappa}_{t-1}, \hat{\boldsymbol{x}}_{t-1}^+, \mathcal{P}_{t-1}^+, \mathcal{R}_t, s_t, \boldsymbol{y}_t, \bar{\boldsymbol{\kappa}}_t, \bar{x}, \bar{p}, \bar{q}$

$J \leftarrow$ **length**$(\hat{\boldsymbol{x}}_{t-1}^+)$
$K \leftarrow$ **length**$(\boldsymbol{\kappa}_{t-1})$
$d \leftarrow K - J - 1$
$I \leftarrow J - d$
$\sigma \leftarrow 0$
**if** $s_t \geq \kappa_{t-1_{J+1}}$ **then**
  **if** $s_t \geq \kappa_{t-1_K}$ **then**
    $\sigma \leftarrow d + 1$
  **else**
    $\sigma$ such that $s_t \in [\kappa_{t-1_{d+I+1+\sigma}}, \kappa_{t-1_{d+I+2+\sigma}})$
  **end**
**else if** $s_t < \kappa_{t-1_{d+1}}$ **then**
  **if** $s_t < \kappa_{t-1_1}$ **then**
    $\sigma \leftarrow -(d+1)$
  **else**
    $\sigma$ such that $s_t \in [\kappa_{t-1_{d+1+\sigma}}, \kappa_{t-1_{d+2+\sigma}})$
  **end**
**end**
$\mathcal{A}_t \in \mathbb{R}^{J \times J}$ from (16)
$\boldsymbol{Q}_t \leftarrow \bar{q} \boldsymbol{I}_{J \times J}$
**if** $\sigma \geq 0$ **then**
  $\boldsymbol{\kappa}_t$ from (15)
  $\boldsymbol{u}_t = (\boldsymbol{0}_{1\times(J-\sigma)}, \bar{x}\boldsymbol{1}_{1\times\sigma})^\top$
  $\boldsymbol{Q}_{t_{m;m}} \leftarrow \bar{p},\ m = J - \sigma + 1, J - \sigma + 2, \dots, J$
**else**
  $\boldsymbol{\kappa}_t$ from (19)
  $\boldsymbol{u}_t = (\bar{x}\boldsymbol{1}_{1\times(-\sigma)}, \boldsymbol{0}_{1\times(J+\sigma)})^\top$
  $\boldsymbol{Q}_{t_{m;m}} \leftarrow \bar{p},\ m = 1, 2, \dots, -\sigma$
**end**
$\mu$ such that $s_t \in [\kappa_{t_\mu}, \kappa_{t_{\mu+1}})$
$V_t \leftarrow$ **length**$(\boldsymbol{y}_t)$
$\boldsymbol{C}_t \in \mathbb{R}^{V_t \times J}$ from (20)
$[\hat{\boldsymbol{x}}_t^+, \mathcal{P}_t^+] \leftarrow$ Algorithm 1$(\hat{\boldsymbol{x}}_{t-1}^+, \mathcal{P}_{t-1}^+, \boldsymbol{u}_t, \boldsymbol{y}_t, \mathcal{A}_t, \boldsymbol{I}_{J\times J}, \boldsymbol{C}_t, \boldsymbol{Q}_t, \mathcal{R}_t)$

**Output**: $\boldsymbol{\kappa}_t, \hat{\boldsymbol{x}}_t^+, \mathcal{P}_t^+$

## 3. Numerical experiments

We demonstrate the effectiveness of Algorithm 2 in three examples and compare its results with the corresponding WLS solution.

### 3.1. General experimental setup

We choose $\{(s_t, \boldsymbol{y}_t)\}_{t=1,2,\dots,n}$ with $n = 5000$, where

$$s_t = 0.01 + 0.02(t-1), \tag{21}$$

$$y_{t,1} = \begin{cases} 20, & \text{if } 30 \leqslant s_t < 70 \\ 10, & \text{otherwise} \end{cases} \quad \text{and} \tag{22}$$

$$y_{t,2} = y_{t,3} = 0 \;\forall\, t \tag{23}$$

By the term thinned out data set we refer to a second data set that differs from the full data set defined above only by $s_t = 0.5 + (t-1)$ and $n = 100$.

We approximate the data sets with a cubic ($d = 3$) B-spline function $f(s)$ whereby we assume that the measurements $y_{t,2}$ refer to the slope $\frac{\partial}{\partial s} f(s)$ of the B-spline function and the measurements $y_{t,3}$ to the curvature $\frac{\partial^2}{\partial s^2} f(s)$. The reciprocals of the relative weights between the different target criteria are specified by the diagonal measurement covariance matrix $\mathcal{R} \in \mathbb{R}^{3\times3}$ with $\mathcal{R}_{t_{1;1}} = 1, \mathcal{R}_{t_{2;2}} = 10^{-2}$ and $\mathcal{R}_{t_{3;3}} = 10^{-3}$. The diagonal measurement covariance matrix of WLS $\mathbf{R} \in \mathbb{R}^{3n\times3n}$ has analogous values:

$$\begin{aligned} &\mathbf{R}_{i+1;i+1} = 1,\ \mathbf{R}_{i+2;i+2} = 10^{-2},\ \mathbf{R}_{i+3;i+3} = 10^{-3}, \\ &i = 3(r-1),\ r = 1, 2, \dots, n \end{aligned} \tag{24}$$

The chosen weighting helps to prevent overshoots and oscillations of $f(s)$ and leads to a B-spline function that smooths the jumps of $y_{t,1}$ in the data sets. For all experiments we use $\bar{q} = 10^{-12}$, $\bar{x} = 0$ and $\bar{p} = 10^4$.

### 3.2. Experiment 1: Setup

We require that the definition range of $f$ comprises $I = 5$ spline intervals and define $K = 12$ knots. We initialize $\boldsymbol{\kappa}$ with $\boldsymbol{\kappa}_0 = (-10, -6.6, -3, -1, 25, 35, 60, 80, 100, 101, 103.2, 110)$, hence $\mathcal{D}_0 = [-1, 100)$. As $s_t \in [s_1, s_n] = [0.01, 99.99] \subset \mathcal{D}_0 \forall t$, no shift operation is needed.

### 3.3. Experiment 1: Results & discussion

Fig. 3 shows the results. The first 25% of the data points $(s_t, \boldsymbol{y}_t)$ all belong to the first spline interval $[-1, 25)$. Therefore they only influence the first four of the $J = 8$ coefficients. The remaining four

coefficient are still at their initial values $\bar{x}$, hence $f(s) \to 0$ for increasing $s$.

With increasing amount of processed data points, the RBA solution converges to the WLS solution. This demonstrates how RBA computes the coefficients for the approximating function iteratively like RLS and how it finds a solution close to the one determined by WLS. Although the data set is symmetrical to a vertical straight line through $s = 50$, both the WLS solution and the end result obtained with RBA are not. This is caused by the asymmetrical knot vector.

All coefficient values of the RBA end result are lower than the corresponding values of the WLS solution. The first coefficient value differs the most with $4.2020 \times 10^{-6}$ for the full data set and with $2.0742 \times 10^{-4}$ for the thinned out data set. For the thinned out data set the deviation is larger because less time steps are available to update each coefficient estimate. If coefficient values are initialized with $\bar{x} > 0$, the sign of the differences can change.

### 3.4. Experiment 2: Setup

In this experiment we perform four runs of RBA with $I = 1, 3, 7$ and 20, respectively, in order to investigate the effect of the choice of $I$. We set $\boldsymbol{\kappa}_0 = (-15, 10, 15, 20)$ for $I = 1$, $\boldsymbol{\kappa}_0 = (-15, 10, \ldots, 30)$ for $I = 3$, $\boldsymbol{\kappa}_0 = (-15, 10, \ldots, 50)$ for $I = 7$, and $\boldsymbol{\kappa}_0 = (-15, 10, \ldots, 115)$ for $I = 20$. For $I = 20$ the resulting $\mathcal{D}_0$ comprises all $s_t$ of the data set and therefore no shift operation is needed. For $I = 1, 3$ and 7, RBA has to perform several right shifts by one element in order to be able to process all data points. For each shift operation, we have to define an additional knot $\bar{\kappa}_{t_K}$ in the vector $\bar{\boldsymbol{\kappa}}_t$. We choose $\bar{\kappa}_{t_k} = 25, 30, \ldots, 115$ for $I = 1$, $\bar{\kappa}_{t_K} = 35, 40, \ldots, 115$ for $I = 3$ and $\bar{\kappa}_{t_K} = 55, 60, \ldots, 115$ for $I = 7$. For evaluation purposes we save $\hat{x}_{t-1_1}$ and $\kappa_{t-1_1}$ separately RBA before a shift operation is performed.

### 3.5. Experiment 2: Results & discussion

Fig. 4 displays the results. As both the data set and knot vector are symmetrical to a vertical straight line through $s = 50$, the WLS solution is symmetrical as well. The RBA solution converges to the WLS solution as $I$ increases. For $I = 1$ and $I = 3$, the resulting B-spline function is asymmetrical with respect to a straight vertical line through $s = 50$. For $I = 7$, RBA provides almost the same result as for $I = 20$. Consequently, we can reduce $I$ from 20 to 7 without noticeably worsening the quality of the approximation. Lowering $I$ leads to less computational effort in the KF because $\mathcal{P}_t^-, \mathcal{P}_t^+$ and $\mathcal{Q}_t$ are $(d + I) \times (d + I)$ matrices and therefore the asymptotic time complexity of Algorithm 1 (one iteration) is $\mathcal{O}((d + I)^3)$ if the standard method for matrix multiplication is used [26,27]. Under the same conditions both RLS and the known methods based on the KF need $20 \times 20$ matrices because shift operations are not possible. If the thinned out data set is used, the determined coefficients differ by about $10^{-3}$ from corresponding coefficients obtained using the full data set.

### 3.6. Experiment 3: Setup

We test the influence of the shift operation on the result. We use the same settings as in Experiment 2 but select $\boldsymbol{\kappa}_0$ such that the first data points lie in the rightmost spline interval. We initial-

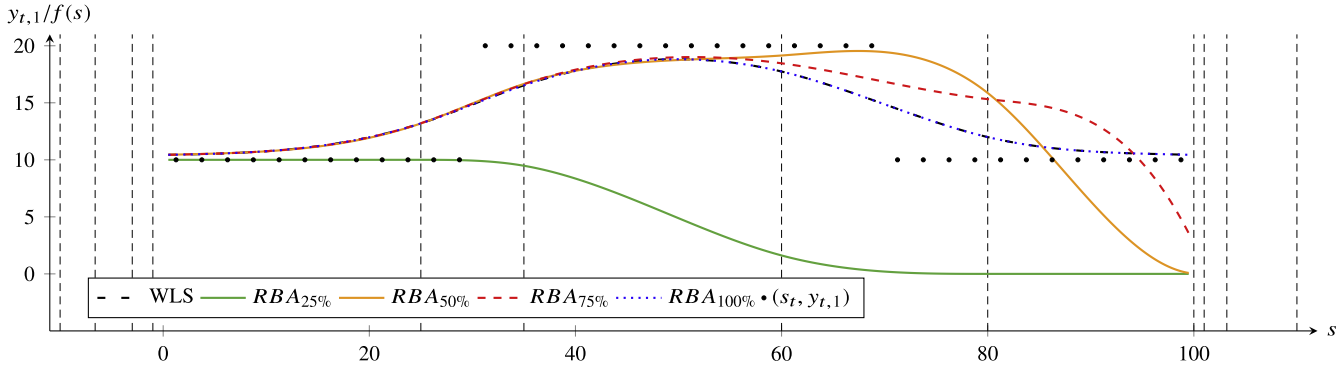


**Fig. 3.** Experiment 1: 40 of the 5000 data points $(s_t, y_{t,1})$ (black dots) and all knots (vertical dashed lines) are shown as well as the B-spline function $f(s)$ determined by WLS and RBA with $I = 5$. For RBA the intermediate results are depicted for time steps at which 25%, 50% and 75% of the data points have been processed as well as the end result.

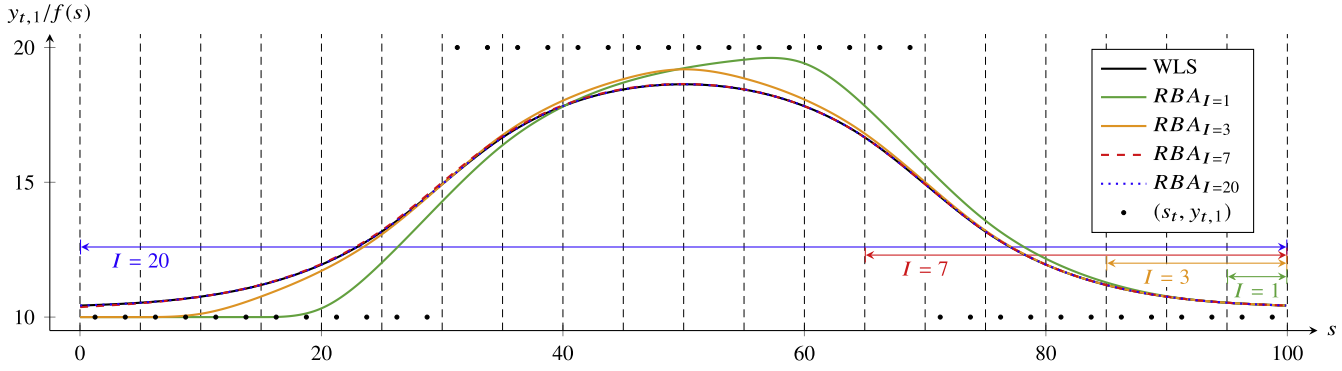


**Fig. 4.** Experiment 2: 40 of the 5000 data points $(s_t, y_{t,1})$ (black dots) and knots 0,5,. . ., 100 (vertical dashed lines) are shown as well as the B-spline function $f(s)$ determined by WLS and RBA with $I = 1$, $I = 3$, $I = 7$ and $I = 20$. The double-headed arrows visualize the definition range of $f(s)$ while data points that lie in the interval $[95, 100)$ are processed.

ize the last eight knots with $\boldsymbol{\kappa}_{0_{k-7,\ldots,K}} = (-15, -10, \ldots, 20)$. For $I = 3, I = 7$ and $I = 20$ additional preceding knots are needed. These can have arbitrary values that fulfill the requirement of strictly increasing knots in $\boldsymbol{\kappa}_0$. In order to approximate all data points, 20 shift operations have to be performed in each run. For $I = 1$, this has already been the case in Experiment 2 and therefore we get an identical result. However, for $I = 3, I = 7$ and $I = 20$, we have increased the number of required shift operations by 3, 7 and 20, respectively. The additional knot values we specify during each run are $\bar{\kappa}_{t_k} = 25, 30, \ldots, 115$.

### 3.7. Experiment 3: Results & discussion

We performed runs for $I = 3, I = 7$ and $I = 20$ with both the full and thinned out data set and evaluated the absolute values of the differences of corresponding coefficient values between Experiment 2 and Experiment 3. The maximum was $8.8818 \times 10^{-15}$, which is close to the machine accuracy of $2.2204 \times 10^{-16}$.

## 4. Conclusions

We proposed a novel recursive B-spline approximation (RBA) algorithm for the approximation of an unbounded set of data points with a B-spline function. The coefficients of the B-spline function are sequentially estimated using a Kalman filter (KF). The total computational effort increases linearly with the number of approximated data points. RBA enables to shift estimated B-spline coefficients within the state vector of the KF during its time update in combination with a shift of the components of the B-spline knot vector. This shift operation allows to shift the definition range of the B-spline function during run-time.

RBA is advantageous in online applications in which the magnitude of the data points is not exactly known or changes over time. Then data points can be outside the definition range. In contrast to previous algorithms based on the KF, RBA shifts the definition range in order to approximate these data points.

RBA is also beneficial when a tradeoff between low computational effort and high approximation quality is needed because the shift operation of RBA allows to reduce the size of the KF state vector in online and offline applications.

Numerical experiments show that the RBA result converges to the weighted least squares solution as the size of the state vector is increased. Moreover, the experimental results reveal that few spline intervals are sufficient for good approximation results.

The number of required shift operations can increase when the size of the state vector is reduced to lower the computational effort. The experiments indicate that effects of shift operations on the resulting approximation function are negligible. Each shift operations comes at the expense that a part of the approximation result is forgotten in order to keep the sizes of matrices and vectors constant. A growing approximation interval can be realized by storing matrix and vector elements separately from RBA before they are overwritten.

## Acknowledgment

The work presented in this publication has been performed as part of the research project *e-volution* which is supported by the German Federal Ministry of Education and Research (BMBF), FKZ: 16EMO0071. The authors also wish to thank the anonymous reviewers for their valuable comments and suggestions.

## References

[1] J. Jauch, (Sep. 2016). doi:http://dx.doi.org/10.5281/zenodo.157732, <http://github.com/JensJauch/RBA>.
[2] T. Lyche, K. Mørken, Spline Methods Draft, Department of Informatics Centre of Mathematics for Applications, University of Oslo, 2008.
[3] Y. Kineri, M. Wang, H. Lin, T. Maekawa, B-spline surface fitting by iterative geometric interpolation/approximation algorithms, Comput. Aided Des. 44 (7) (2012) 697–708, http://dx.doi.org/10.1016/j.cad.2012.02.011.
[4] S. He, D. Ou, C. Yan, C.-H. Lee, A chord error conforming tool path B-spline fitting method for nc machining based on energy minimization and LSPIA, J. Comput. Des. Eng. 2 (4) (2015) 218–232, http://dx.doi.org/10.1016/j.jcde.2015.06.002. CNCM 2015 Special Issue.
[5] M. Elbanhawi, M. Simic, R.N. Jazar, Continuous path smoothing for car-like robots using B-spline curves, J. Intell. Rob. Syst. 80 (1) (2015) 23–56, http://dx.doi.org/10.1007/s10846-014-0172-0.
[6] B. Cao, G.I. Doods, G.W. Irwin, Time-optimal and smooth constrained path planning for robot manipulators, in: Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, vol. 3, 1994, pp. 1853–1858. doi:http://dx.doi.org/10.1109/ROBOT.1994.351191.
[7] M.S. Grewal, A.P. Andrews, Kalman Filtering, John Wiley & Sons Inc, 2008, http://dx.doi.org/10.1002/9780470377819.
[8] D. Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, Wiley, 2006.
[9] M. Harashima, L. Ferrari, P. Sankar, Spline approximation using Kalman filter state estimation, IEEE Trans. Circuits Syst. II: Analog Digital Signal Process. 44 (5) (1997) 421–424, http://dx.doi.org/10.1109/82.580860.
[10] K.H. Lim, K.P. Seng, L.-M. Ang, River flow lane detection and kalman filtering-based B-spline lane tracking, Int. J. Veh. Technol. (2012), http://dx.doi.org/10.1155/2012/465819.
[11] M. Shahrbanozadeh, G.-A. Barani, S. Shojaee, Simulation of flow through dam foundation by isogeometric method, Eng. Sci, Technol. Int. J. 18 (2) (2015) 185–193, http://dx.doi.org/10.1016/j.jestch.2014.11.001.
[12] X.-D. Chen, W. Ma, J.-C. Paul, Cubic B-spline curve approximation by curve unclamping, Comput. Aided Des. 42 (6) (2010) 523–534, http://dx.doi.org/10.1016/j.cad.2010.01.008.
[13] Å. Björck, Numerical Methods in Matrix Computations, Texts in Applied Mathematics, vol. 59, Springer International Publishing, 2015, http://dx.doi.org/10.1007/978-3-319-05089-8.
[14] G. Golub, C. Van Loan, Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences, third ed., Johns Hopkins University Press, 1996.
[15] K. Zhao, F. Ling, H. Lev-Ari, J.G. Proakis, Sliding window order-recursive least-squares algorithms, IEEE Trans. Signal Process. 42 (8) (1994) 1961–1972, http://dx.doi.org/10.1109/78.301835.
[16] J. Jiang, Y. Zhang, A revisit to block and recursive least squares for parameter estimation, Comput. Electron. Eng. 30 (5) (2004) 403–416, http://dx.doi.org/10.1016/j.compeleceng.2004.05.002.
[17] B.Y. Choi, Z. Bien, Sliding-windowed weighted recursive least-squares method for parameter estimation, Electron. Lett. 25 (20) (1989) 1381–1382, http://dx.doi.org/10.1049/el:19890924.
[18] W. Wang, J. Zhao, Inverse updating and downdating for weighted linear least squares using m-invariant reflections, Linear Algebra App. 291 (1) (1999) 185–199, http://dx.doi.org/10.1016/S0024-3795(99)00007-5.
[19] S.J. Olszanskyj, J.M. Lebak, A.W. Bojanczyk, Rank-k modification methods for recursive least squares problems, Numer. Algorithms 7 (2) (1994) 325–354, http://dx.doi.org/10.1007/BF02140689.
[20] S. Hammarling, C. Lucas, Updating the QR Factorization and the Least Squares Problem, Manchester Institute for Mathematical Sciences, School of Mathematics, MIMS EPrint 2008.111, <http://www.eprints.ma.man.ac.uk/1192/01/covered/MIMSep2008111.pdf>.
[21] O. Olsson, T. Ivarsson, Using the QR Factorization to swiftly update least squares problems, student paper (2014).
[22] H. Lin, G. Wang, C. Dong, Constructing iterative non-uniform B-spline curve and surface to fit data points, Sci. Chin. Ser.: Inf. Sci. 47 (3) (2004) 315–331, http://dx.doi.org/10.1360/02yf0529.
[23] C. Deng, H. Lin, Progressive and iterative approximation for least squares B-spline curve and surface fitting, Comput. Aided Des. 47 (2014) 32–44, http://dx.doi.org/10.1016/j.cad.2013.08.012.
[24] H. Fujioka, H. Kano, Recursive construction of smoothing spline surfaces using normalized uniform b-splines, in: Proceedings of the 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009, 2009, pp. 5550–5555. doi: http://dx.doi.org/10.1109/CDC.2009.5399523.
[25] C. Radhakrishna Rao, Shalabh, H. Toutenburg, C. Heumann, Linear Models and Generalizations: Least Squares and Alternatives, third ed., Springer Series in Statistics, Springer, Berlin Heidelberg, 2008, http://dx.doi.org/10.1007/978-3-540-74227-2.
[26] J.R. Bunch, J.E. Hopcroft, Triangular factorization and inversion by fast matrix multiplication, Math. Comput. 28 (125) (1974) 231–236, http://dx.doi.org/10.2307/2005828.
[27] F. Le Gall, Faster Algorithms for Rectangular Matrix Multiplication, in: IEEE 53rd Annual Symposium on, 2012. Foundations of Computer Science (FOCS), 2012, pp. 514–523. doi:http://dx.doi.org/10.1109/FOCS.2012.80.