2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Fast and smooth surface B-spline interpolation for regularly spaced data used in system modeling to make MPC real-time feasible*

R. Mitze[1], D. Dillkötter[1], S. Gros[2], A. Schild[3] and M. Mönnigmann[1]

Abstract— **Advanced control applications require accurate system models. Obviously, these models must be evaluated sufficiently fast in order for a model-based controller to be real-time feasible. This holds for methods that are based on online optimizations, such as model predictive control (MPC), in particular. It is common to describe nonlinear static parts of system models with interpolated look-up tables, because they are computationally efficient and they can be designed to provide the required accuracy. Since the underlying data are often determined with measurements or simulations, the location of data points can be chosen by the user to some extent. We use data on regular grids and B-splines with uniform knotvectors located at the data grid points, because this results in smooth interpolated look-up tables that can be evaluated very fast. The algorithm for the online evaluation and interpolation can be extended to efficiently provide first and second order derivatives, which are, for example, needed in MPC. We illustrate the use of the implemented methods with the look-up table of the aerodynamic power coefficient of a wind turbine generator and compare computation times for an implementation on a CPU and on an FPGA.**

I. INTRODUCTION

Interpolated look-up tables are often used in system modeling to describe nonlinear static system behavior. They have several advantages: Complex parts of models can be replaced to reduce model complexity, subsystems that are difficult to model can be described by measured or simulated data points, and the model behavior can be modified systematically by simply changing the shape of the look-up table.

Since the computational demand of a look-up table increases exponentially with the number of input parameters, they are only suitable for low-dimensional problems. Almost all of the look-up tables used in practice have just two input parameters; more than ten years ago modern combustion engine control units already contained more than 100 two-dimensional look-up tables [1].

The data the look-up table is based on need to be interpolated. In control applications such as MPC, the interpolation must be done online. The computational time required for the interpolation must obviously be small to ensure real-time feasibility of these controllers. Depending on the optimization algorithm that is used to solve the optimization problem of the MPC, it is also necessary to provide derivatives up to second order.

Several interpolation methods exist. Uniform bicubic surface B-splines are well suited for a fast and accurate interpolation of two-dimensional regularly spaced look-up table data. The shape of a B-spline interpolated surface is defined by knots. The desired smoothness† can be achieved by placing the knots appropriately. A cubic degree guarantees the spline to be twice continuously differentiable, and it allows a fast interpolation, because no more than 4^2 knots are relevant for the determination of the interpolated value and its associated derivatives. Equidistant, so-called uniform, knot vectors also contribute to a fast interpolation algorithm (see Sec. II and Sec. III-B).

The choice of knots has been subject of much research (see [2] and references therein). Assuming that sufficient memory exists, it is a good choice to place the knots at the same grid-positions as the given data [3], [4]. We use a standard optimization [5], [6] problem that penalizes the deviation between original and interpolated data and the curvature of the interpolated data to determine the optimal values at the knots (also referred to as height of the knots).

Sec. II summarizes some facts about uniform bicubic surface B-spline interpolation that are needed for the paper. Sec. III-A states the QP that generates knots for given two-dimensional regularly spaced data for smooth and fast interpolation. The interpolation algorithm is stated in Sec. III-B. Sec. IV illustrates the use of the presented knot algorithm and gives computational times of the interpolation algorithm of a CPU and FPGA implementation. The look-up table of the aerodynamic power coefficient of a wind turbine generator serves as an example. Brief conclusions are given in Sec. V.

II. BACKGROUND AND NOTATION

We give a brief overview on B-splines with a focus on uniform bicubic surface B-splines. Further information can be found in [7] and [4].

The shape of a B-spline interpolated surface depends on the position of the knots. We assume that the knots x_i, y_j, c_{ij} are given in this section, where x_i and y_j are the elements of the two knot vectors and c_{ij} refers to the value at (x_i, y_j). Furthermore, we assume that the x_i and the y_j are

*This work was supported by the German Federal Ministry for Economic Affairs and Energy under grant 0324125C

[1]R. Mitze, D. Dillkötter and M. Mönnigmann are with Dep. of Mechanical Engineering, Ruhr-Universität Bochum, 44801 Bochum, Germany. E-mail: ruth.mitze@rub.de, david.dillkoetter@rub.de and martin.moennigmann@rub.de, [2]S. Gros is with Dep. of Electrical Engineering, Chalmers University of Technology, 41296 Göteborg, Sweden. E-mail: grosse@chalmers.se, [3]A. Schild is with IAV GmbH, 38518 Gifhorn, Germany. E-mail: axel.schild@iav.de

†'Smoothness' has a somewhat vague meaning in the context of interpolation, which refers to dampening measurement or simulation errors in the data. This can be achieved by imposing an upper bound on the absolute value of second order derivatives, or their approximations.

equidistant, but the two step sizes are not equal in general. The desired interpolated function $s(x, y)$ reads

$$s(x,y) = \sum_j \sum_i c_{ij} \cdot B_i^p(x) \cdot B_j^p(y) \tag{1}$$

where the B-splines $B_i^p(x)$, $B_j^p(y)$ are piecewise polynomials of degree p that serve as basis functions. We use B-splines of degree $p = 3$ and write $B_i(x)$, $B_j(y)$ for brevity. In case of a uniform knot vector the shape of the B-splines is identical for every knot and merely shifted so that their maximum is located at the corresponding knot. Cubic B-splines are nonzero within five knots. As a consequence, $s(x, y)$ can be evaluated locally around (x, y), because splines centered around a point far from (x, y) do not contribute. More precisely, $s(x, y)$ in (1) results from

$$\begin{aligned} s(x,y) = &\sum_{j=l-1}^{l+2} \sum_{i=k-1}^{k+2} c_{ij} \cdot B_i(x) \cdot B_j(y), \\ &\text{if } x \in [x_k, x_{k+1}) \text{ and } y \in [y_l, y_{l+1}). \end{aligned} \tag{2}$$

As mentioned before we place the knots at the same positions as the given data. Assume that $(n + 1) \cdot (m + 1)$ data points located at the positions $x_0, ..., x_n$ and $y_0, ..., y_m$ with the constant step sizes h_x and h_y are given. According to (2) we need one more knot on each side to cover the first and last intervals (see Fig. 1). It follows that the two knot vectors have $n + 3$ and $m + 3$ entries x_i, $i = -1, ..., n + 1$ and y_j, $j = -1, ..., m + 1$.

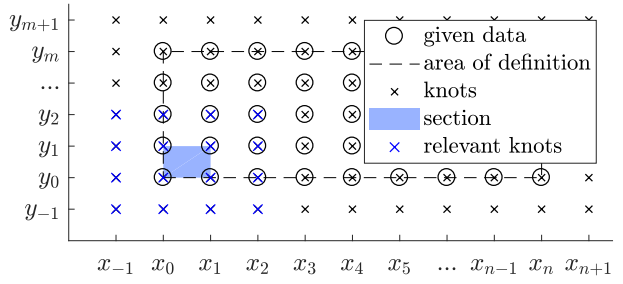

Fig. 1. Knot positions and the area of definition.

Fig. 2 shows the B-splines of the relevant knots over an arbitrary interval $[x_k, x_{k+1})$ in the knot vector in x direction. We introduce the coordinate

$$a_k = \frac{x - x_k}{h_x}, \quad x \in [x_k, x_{k+1}), \tag{3}$$

which gives us the normalized distance between x and the previous knot x_k, hence $a_k \in [0, 1)$. The third derivatives of the B-splines are constant over a_k. Integration yields

$$\begin{aligned} {B_i}'''(a_k) =& {B_i}'''(a_k = 0) = {B_i}'''|_0 \\ {B_i}''(a_k) =& {B_i}'''|_0 \cdot a_k + {B_i}''|_0 \\ {B_i}'(a_k) =& \frac{1}{2} \cdot {B_i}'''|_0 \cdot a_k^2 + {B_i}''|_0 \cdot a_k + {B_i}'|_0 \\ B_i(a_k) =& \frac{1}{6} \cdot {B_i}'''|_0 \cdot a_k^3 + \frac{1}{2} \cdot {B_i}''|_0 \cdot a_k^2 \\ &+ {B_i}'|_0 \cdot a_k + B_i|_0 \end{aligned} \tag{4}$$

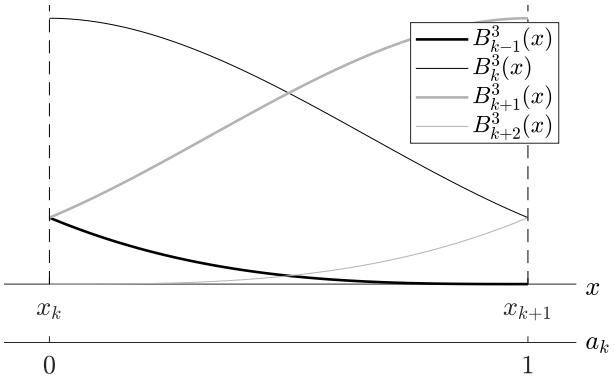

Fig. 2. B-splines in an interval $[x_k, x_{k+1})$ within the knot vector and the associated normalized coordinate a_k.

The integration constants can be determined with the recursive deBoor algorithm [7]. For the normalized coordinate a_k the results are constant numbers. Tab. I lists them for a degree 3. In order to rescale these functions from a_k to x, each μth derivative function in (4) must be divided by $(h_x)^\mu$. The same approach can be used to determine the B-splines

TABLE I
UNIFORM NORMALIZED CUBIC B-SPLINES AT A KNOT x_k

i	${B_i}\|_0$	${B_i}'\|_0$	${B_i}''\|_0$	${B_i}'''\|_0$
⋮	0	0	0	0
$k-1$	$\frac{1}{6}$	$-\frac{1}{2}$	1	-1
k	$\frac{2}{3}$	0	-2	3
$k+1$	$\frac{1}{6}$	$\frac{1}{2}$	1	-3
$k+2$	0	0	0	1
⋮	0	0	0	0

in y-direction using the coordinate $a_l = \frac{y - y_l}{h_y}$, which is valid over $y \in [y_l, y_{l+1})$.

For later use we define the basis vectors $\boldsymbol{b}(x) \in \mathbb{R}^{n+3}$ and $\boldsymbol{b}(y) \in \mathbb{R}^{m+3}$

$$\begin{aligned} \boldsymbol{b}(x) &= \begin{pmatrix} B_{-1}(x) & B_0(x) & \cdots & B_{n+1}(x) \end{pmatrix}, \\ \boldsymbol{b}(y) &= \begin{pmatrix} B_{-1}(y) & B_0(y) & \cdots & B_{m+1}(y) \end{pmatrix} \end{aligned}$$

containing the B-splines and the knot vector $\boldsymbol{c} \in \mathbb{R}^{(n+3)\cdot(m+3)}$

$$\begin{aligned} \boldsymbol{c} = \big(& c_{-1,-1} \quad \cdots \quad c_{-1,m+1} \quad c_{0,-1} \quad \cdots \quad c_{0,m+1} \\ & \cdots \quad c_{n+1,-1} \quad \cdots \quad c_{n+1,m+1} \big)^T \end{aligned}$$

containing the knot values. Equation (1) can then be rewritten as

$$s(x,y) = [\boldsymbol{b}(x) \otimes \boldsymbol{b}(y)] \cdot \boldsymbol{c}. \tag{5}$$

III. FAST AND SMOOTH SURFACE B-SPLINE INTERPOLATION

Sec. III-A and Sec. III-B present the algorithm for generating the knots and the algorithm for a fast interpolation, respectively.

A. KNOT GENERATION

We need to generate knot heights c_{ij}, $i = -1, \dots, n+1$, $j = -1, \dots, m+1$ for a given set of data x_i, y_j, z_{ij} ($i = 0, \dots, n$, $j = 0, \dots, m$). This is achieved with a QP that penalizes two contributions, the deviation between the given data and the interpolated look-up table, and large curvatures, which are represented by the second derivative of the interpolated look-up table. More precisely, we solve

$$\min_{\boldsymbol{c}} \; Q + rR, \tag{6}$$

where Q and R represent the deviation and curvature explained below, and $r \in \mathbb{R}$, $r > 0$ is a weighting parameter.

The deviation can be expressed as

$$Q = \sum_{j=0}^{m} \sum_{i=0}^{n} \left(z_{ij} - s(x_i, y_j) \right)^2. \tag{7}$$

With equation (5) this becomes

$$Q = \sum_{j=0}^{m} \sum_{i=0}^{n} \left(z_{ij} - [\boldsymbol{b}(x_i) \otimes \boldsymbol{b}(y_j)] \cdot \boldsymbol{c} \right)^2. \tag{8}$$

Since the knots are placed at the same positions as the given data points, the basis vectors become (see Tab. I)

$$\boldsymbol{b}(x_i) = \frac{1}{6} \cdot \Bigg(\underbrace{0 \;\; \cdots \;\; 0}_{i \text{ times}} \;\; 1 \;\; 4 \;\; 1 \;\; \underbrace{0 \;\; \cdots \;\; 0}_{n-i \text{ times}} \Bigg),$$

$$\boldsymbol{b}(y_j) = \frac{1}{6} \cdot \Bigg(\underbrace{0 \;\; \cdots \;\; 0}_{j \text{ times}} \;\; 1 \;\; 4 \;\; 1 \;\; \underbrace{0 \;\; \cdots \;\; 0}_{m-j \text{ times}} \Bigg).$$

Introducing the data vector $\boldsymbol{z} \in \mathbb{R}^{(n+1)\cdot(m+1)}$ and the matrix $\boldsymbol{A} \in \mathbb{R}^{(n+1)\cdot(m+1)\times(n+3)\cdot(m+3)}$

$$\boldsymbol{z} = \begin{pmatrix} z_{0,0} \\ \vdots \\ z_{0,m} \\ z_{1,0} \\ \vdots \\ z_{1,m} \\ \vdots \\ z_{n,0} \\ \vdots \\ z_{n,m} \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} \boldsymbol{b}(x_0) \otimes \boldsymbol{b}(y_0) \\ \vdots \\ \boldsymbol{b}(x_0) \otimes \boldsymbol{b}(y_m) \\ \boldsymbol{b}(x_1) \otimes \boldsymbol{b}(y_0) \\ \vdots \\ \boldsymbol{b}(x_1) \otimes \boldsymbol{b}(y_m) \\ \vdots \\ \boldsymbol{b}(x_n) \otimes \boldsymbol{b}(y_0) \\ \vdots \\ \boldsymbol{b}(x_n) \otimes \boldsymbol{b}(y_m) \end{pmatrix}$$

Q can be written as

$$Q = \boldsymbol{c}^T \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{c} - 2\boldsymbol{z}^T \boldsymbol{A} \boldsymbol{c} + \boldsymbol{z}^T \boldsymbol{z}. \tag{9}$$

Since the last term in the sum does not depend on the optimization variables, dropping this term does not alter the value of $\boldsymbol{c}$ that results from (17).

The second term R of the objective function penalizes curvatures by minimizing

$$R = \int_{y_0}^{y_m} \int_{x_0}^{x_n} \lambda_1(x,y)^2 + \lambda_2(x,y)^2 \, \mathrm{d}x\mathrm{d}y \tag{10}$$

where $\lambda_1(x,y)$ and $\lambda_2(x,y)$ refer to the eigenvalues of the Hessian matrix of s. It is easy to show that

$$\begin{aligned} R = \int_{y_0}^{y_m} \int_{x_0}^{x_n} & \left(\frac{\partial^2 s(x,y)}{\partial x^2} \right)^2 \\ & + 2 \cdot \left(\frac{\partial^2 s(x,y)}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 s(x,y)}{\partial y^2} \right)^2 \mathrm{d}x\mathrm{d}y. \end{aligned} \tag{11}$$

We briefly summarize how to express R in terms of $\boldsymbol{b}(x)$, $\boldsymbol{b}(y)$ and $\boldsymbol{c}$. Let

$$S_{\mu,\nu} = \int_{y_0}^{y_m} \int_{x_0}^{x_n} \left(\frac{\partial^{(\mu+\nu)} s(x,y)}{\partial x^{(\mu)} \partial y^{(\nu)}} \right)^2 \mathrm{d}x\mathrm{d}y. \tag{12}$$

The piecewise structure of the B-splines can be exploited to simplify (12) to

$$S_{\mu,\nu} = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} \left(\frac{\partial^{(\mu+\nu)} s(x,y)}{\partial x^{(\mu)} \partial y^{(\nu)}} \right)^2 \mathrm{d}x\mathrm{d}y. \tag{13}$$

Using (5) the derivatives in (13) can be expressed as

$$\frac{\partial^{(\mu+\nu)} s(x,y)}{\partial x^{(\mu)} \partial y^{(\nu)}} = [\boldsymbol{b}^{(\mu)}(x) \otimes \boldsymbol{b}^{(\nu)}(y)] \cdot \boldsymbol{c}$$

where derivatives of vectors refer to the derivatives of their elements. Rearranging the integrals in (13) yields

$$\begin{aligned} S_{\mu,\nu} = & \boldsymbol{c}^T \cdot \Bigg[\sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \left(\int_{x_i}^{x_{i+1}} \boldsymbol{b}^{(\mu)}(x)^T \cdot \boldsymbol{b}^{(\mu)}(x) \mathrm{d}x \right) \\ & \otimes \left(\int_{y_j}^{y_{j+1}} \boldsymbol{b}^{(\nu)}(y)^T \cdot \boldsymbol{b}^{(\nu)}(y) \mathrm{d}y \right) \Bigg] \cdot \boldsymbol{c}, \end{aligned} \tag{14}$$

where integrals of matrices are understood to be integrals of their elements. A change of coordinates according to (3) and the corresponding change of coordinates for y yields

$$\begin{aligned} S_{\mu,\nu} = & \boldsymbol{c}^T \cdot \Bigg[\sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \left(\int_0^1 \boldsymbol{b}^{(\mu)}(a_i)^T \cdot \boldsymbol{b}^{(\mu)}(a_i) \mathrm{d}a_i \right) \\ & \otimes \left(\int_0^1 \boldsymbol{b}^{(\nu)}(a_j)^T \cdot \boldsymbol{b}^{(\nu)}(a_j) \mathrm{d}a_j \right) \Bigg] \cdot \boldsymbol{c}. \end{aligned} \tag{15}$$

More details on (15) are given in the appendix. We refer to the matrix in brackets in (15) as $\boldsymbol{A_{xx}}$, $\boldsymbol{A_{xy}}$ and $\boldsymbol{A_{yy}}$ for the cases $(\mu,\nu) = (2,0)$, $(1,1)$ and $(0,2)$, respectively. R can then be expressed as

$$R = \boldsymbol{c}^T \cdot (\boldsymbol{A_{xx}} + 2\boldsymbol{A_{xy}} + \boldsymbol{A_{yy}}) \cdot \boldsymbol{c}. \tag{16}$$

Substituting (9) and (16) into the QP (6) yields

$$\min_{\boldsymbol{c}} \; \boldsymbol{c}^T \cdot (\boldsymbol{A}^T \boldsymbol{A} + r\boldsymbol{A_{xx}} + 2r\boldsymbol{A_{xy}} + r\boldsymbol{A_{yy}}) \cdot \boldsymbol{c} - 2\boldsymbol{z}^T \boldsymbol{A} \boldsymbol{c}. \tag{17}$$

We introduce the diagonal matrix $\boldsymbol{G} \in \mathbb{R}^{(n+1)\cdot(m+1)\times(n+1)\cdot(m+1)}$ to control the accuracy weight

of each data point

$$\min_{\boldsymbol{c}} \boldsymbol{c}^T \cdot (\boldsymbol{A}^T\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{A} + r\boldsymbol{A}_{\boldsymbol{xx}} + 2r\boldsymbol{A}_{\boldsymbol{xy}} + r\boldsymbol{A}_{\boldsymbol{yy}}) \cdot \boldsymbol{c} - 2\boldsymbol{z}^T\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{A}\boldsymbol{c}. \tag{18}$$

Note this optimization problem may be augmented by constraints to reflect physical properties of the look-up table.

B. ONLINE INTERPOLATION

The interpolation algorithm is designed such that the computational time needed to evaluate the B-spline interpolated value and the associated Jacobian and Hessian is small. In order to minimize the computational effort, (2) can be rewritten as

$$s(x,y) = \sum_{j=l-1}^{l+2} B_j(y) \cdot \sum_{i=k-1}^{k+2} c_{ij} \cdot B_i(x), \quad \text{if } x \in [x_k, x_{k+1}) \text{ and } y \in [y_l, y_{l+1}). \tag{19}$$

The elements of the Jacobian and Hessian of $s(x,y)$ can be expressed in terms of the respective derivatives of the B-splines in (19). The evaluation of (19) can be divided into three major steps:

1) finding the intervals k and l such that $x \in [x_k, x_{k+1})$ and $y \in [y_l, y_{l+1})$,
2) evaluating the B-splines and their derivatives up to second order for all relevant knots,
3) summing the weighted knot values

The first step merely requires to evaluate

$$k = \lfloor \frac{x - x_0}{h_x} \rfloor, \quad l = \lfloor \frac{y - y_0}{h_y} \rfloor. \tag{20}$$

Note that this particularly simple solution to step 1 results, because the knots have been placed at constant distances. Because of this uniform placement, multistep search algorithms such as binary trees are not needed in step 1.

Step 2 is carried out as explained in Sec. II: First we evaluate the normalized distance to the previous knot (see equation (3)), then we use (4) and Tab. I to evaluate the normalized B-spline values and finally, in order to rescale the B-splines, we divide the first derivatives by the step size and the second derivatives by the squared step size. Tab. II lists the resulting expressions for the B-splines in x direction. The computational time can be reduced by identifying common subexpressions and reusing their values. Tab. III lists the entries of Tab. II in way that makes the common subexpressions explicit. The evaluation with common subexpressions can then be carried out as stated in Tab. IV. Multiplications with powers of two can efficiently be implemented with bit shifts. The factors $\frac{1}{6}$, $-\frac{1}{2h_x}$ and $\frac{1}{h_x^2}$ are factored out in the evaluation of the B-splines. They are multiplied after summing the weighed knots.

The B-splines in y-direction can be evaluated in the same manner.

TABLE II
B-SPLINES AT RELEVANT KNOTS

i	$B_i(x)$	$B_i'(x)$	$B_i''(x)$
$k-1$	$\frac{1}{6}\left(-a_k^3 + 3a_k^2 - 3a_k + 1\right)$	$\frac{1}{2h_x}\left(-a_k^2 + 2a_k - 1\right)$	$\frac{1}{h_x^2}\left(-a_k + 1\right)$
k	$\frac{1}{6}\left(3a_k^3 - 6a_k^2 + 4\right)$	$\frac{1}{2h_x}\left(3a_k^2 - 4a_k\right)$	$\frac{1}{h_x^2}\left(3a_k - 2\right)$
$k+1$	$\frac{1}{6}\left(-3a_k^3 + 3a_k^2 + 3a_k + 1\right)$	$\frac{1}{2h_x}\left(-3a_k^2 + 2a_k + 1\right)$	$\frac{1}{h_x^2}\left(-3a_k + 1\right)$
$k+2$	$\frac{1}{6}a_k^3$	$\frac{1}{2h_x}a_k^2$	$\frac{1}{h_x^2}a_k$

TABLE III
REWRITTEN B-SPLINES AT RELEVANT KNOTS

i	$B_i(x) = \frac{1}{6}\cdot$	$B_i'(x) = -\frac{1}{2h_x}\cdot$	$B_i''(x) = \frac{1}{h_x^2}\cdot$
$k-1$	$(1-a_k)^3$	$(1-a_k)^2$	$(1-a_k)$
k	$(2-a_k)^3$ $-4(1-a_k)^3$	$(2-a_k)^2$ $-4(1-a_k)^2$	$(2-a_k)$ $-4(1-a_k)$
$k+1$	$(1+a_k)^3$ $-4a_k^3$	$-(1+a_k)^2$ $+4a_k^2$	$(1+a_k)$ $-4a_k$
$k+2$	a_k^3	$-a_k^2$	a_k

TABLE IV
EVALUATIONS WITH COMMON SUBEXPRESSIONS

name	partial term	operation $+/-$	operation $\cdot$	B-spline
p_0	$\frac{1}{h_x}\cdot(x - x_k)$	1	1	$\cdot\frac{1}{h_x^2} = B_{k+2}''(x)$
p_1	$1 - p_0$	1	0	$\cdot\frac{1}{h_x^2} = B_{k-1}''(x)$
p_2	$2 - p_0$	1	0	
p_3	$1 + p_0$	1	0	
p_4	$p_0 \cdot p_0$	0	1	
p_5	$p_1 \cdot p_1$	0	1	$\cdot\left(-\frac{1}{2h_x}\right) = B_{k-1}'(x)$
p_6	$p_2 \cdot p_2$	0	1	
p_7	$p_3 \cdot p_3$	0	1	
p_8	$p_4 \cdot p_0$	0	1	$\cdot\frac{1}{6} = B_{k+2}(x)$
p_9	$p_5 \cdot p_1$	0	1	$\cdot\frac{1}{6} = B_{k-1}(x)$
p_{10}	$p_6 \cdot p_2$	0	1	
p_{11}	$p_7 \cdot p_3$	0	1	
	$-p_4$	0	1	$\cdot\left(-\frac{1}{2h_x}\right) = B_{k+2}'(x)$
	$p_2 - 4p_1$	1	0	$\cdot\frac{1}{h_x^2} = B_k''(x)$
	$p_3 - 4p_0$	1	0	$\cdot\frac{1}{h_x^2} = B_{k+1}''(x)$
	$p_6 - 4p_5$	1	0	$\cdot\left(-\frac{1}{2h_x}\right) = B_k'(x)$
	$4p_4 - p_7$	1	0	$\cdot\left(-\frac{1}{2h_x}\right) = B_{k+1}'(x)$
	$p_{10} - 4p_9$	1	0	$\cdot\frac{1}{6} = B_k(x)$
	$p_{11} - 4p_8$	1	0	$\cdot\frac{1}{6} = B_{k+1}(x)$

IV. APPLICATION TO CHARACTERISTICS FOR WIND TURBINE MODELING

In wind turbine generator modeling, the aerodynamic power coefficient c_P and the aerodynamic thrust coefficient c_T play an important role. Both c_P and c_T depend on the pitch angle β and the tip-speed ratio λ. This dependency can be represented by characteristics with data points obtained from fluid dynamic computations. Data from a sample c_P look-up table are shown in Fig. 3.

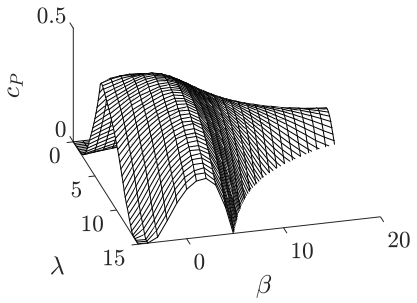

Fig. 3. Data of the c_P look-up table.

In order to adapt a nonlinear model predictive control (NMPC) for wind turbine generators and solve the underlying optimization problem (which involves Newton steps and Hessian approximations), we need the interpolation of these aerodynamic coefficients and the associated Jacobian and Hessian thousands of times in each time step [8, Table III]. The interpolation needs to be fast to ensure real-time feasibility, and smooth to guarantee local convergence. We refer to [8], [9] for more details on this application.

A. KNOT GENERATION

The data for c_P and c_T come from simulations that are subject to numerical errors. These simulation errors result in an artificial non-smoothness which can be removed in the interpolated characteristic with the parameter r. Fig. 4 illustrates this effect for c_P.

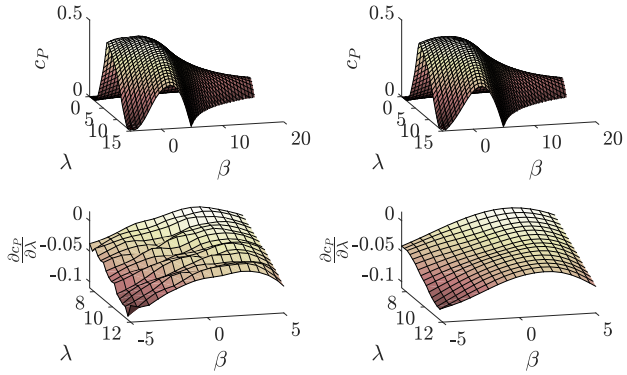

Fig. 4. Interpolated c_P look-up tables and their first derivative with respect to λ for smoothing weight $r = 0.05$ (left) and $r = 1.5$ (right). Derivatives are shown on a subdomain for better visibility. The curvature measure R (see equation (11)) reduces from 0.22 (left) to 0.16 (right).

The aerodynamic power coefficient has negative values for high β and high λ. Due to heuristics embedded in the control scheme, we require all look-up table values to be positive. Hence, all negative data points are set to zero before the spline interpolation is carried out.

We augment the QP by the constraint $\boldsymbol{c} \geq 0$ to prevent an undershoot in the interpolation due to the strong curvature in the vicinity of the resulting kink in the map. The desired effect of the constraint is illustrated in Fig. 5 by showing the interpolation of the zeroed data before and after adding the constraint $\boldsymbol{c} \geq 0$.

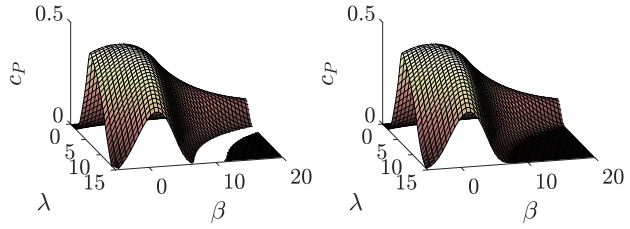

Fig. 5. Interpolated zeroed c_P look-up tables before (left) and after (right) adding the constraint $\boldsymbol{c} \geq 0$.

The interpolation at the kink causes high local second derivatives. Fig. 6 shows this effect can be mitigated with the weight matrix $\boldsymbol{G}$.

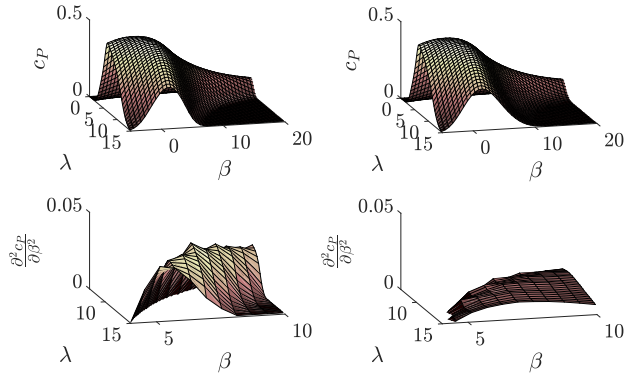

Fig. 6. Interpolated zeroed c_P look-up tables and their second derivative with respect to β before (left) and after (right) lowering the accuracy weight of data points located in the zeroed area by 80%. Second derivatives are shown on a subdomain for better visibility. The curvature measure R introduced in (11) reduces from 0.19 (left) to 0.11 (right).

B. ONLINE IMPLEMENTATION

In the described application c_P and c_T have identical knot vectors. Since we always need the values of both aerodynamic coefficients, step 1 and 2 in Sec. III-B need to be carried out only once. Step 3 needs to be carried out separately for c_P and c_T.

We implemented both the algorithm proposed in [9] and the algorithm presented in Sec. III-B on a CPU (central processing unit, Intel I5-7360U) in ANSI C for comparison. Additionally, we implemented the algorithm presented in Sec. III-B on an FPGA (field programmable gate array, Xilinx Zynq XC7Z010) in VHDL (very high speed integrated circuit hardware description language).

The C-implementation led to computational times of 303ns, which is about two times faster than the reference algorithm presented in [9] (640ns). To further improve the computational time of our algorithm, we implemented parts as multiplication and addition of vectors. Making use of intrinsic functions of current CPUs, we were able to reduce the computational time to 280ns.

In contrast to ANSI C, VHDL allows arbitrary levels of parallelism. This feature can be used to evaluate several of the intermediate expressions listed in Tab. IV simultaneously.

We briefly compare a hand-written implementation to an implementation that was synthesized with a high-level synthesis tool, where pipelining was enforced.

The hand-written VHDL implementation lead to a computational time of 10ns. The code generated with the high-level synthesis tool resulted in a computational time of 214ns. Due to pipelining, however, the high-level synthesis code provides a new result every clock cycle, which here amounts to 3.1ns. Pipelining may therefore be advantageous if multiple evaluations of the characteristics are required simultaneously and a truly concurrent evaluation is ruled out, for example, on a small FPGA. In case of an MPC which uses multiple shooting, many simultaneous evaluations are needed.

Note that times for transmission of data between a CPU and the FPGA have not been accounted for here. Furthermore, we stress all reported times are the times required for c_P, c_T and the associated Jacobians and Hessians.

V. CONCLUSIONS

We presented an offline algorithm which generates knots for a smooth and fast interpolation with bicubic uniform B-splines. We showed the ability to control the smoothness and the shape of the interpolated surface illustrated by the look-up table of the aerodynamic power coefficient used in a wind turbine generator model. Additionally, we demonstrated the online implementation allows computational times down to 280 ns for a ANSI C implementation on a typical CPU, down to 10ns and 3.1ns on an FPGA without and with pipelining, respectively.

APPENDIX

The integrals in (15) result in a symmetric matrix with dimension $(n+3)\times(n+3)$ for a B-spline with cubic degree and a uniform knot vector. The calculations described in Sec. II result in

$$\int_0^1 \boldsymbol{b}^{(\mu)}(a_i)^T \cdot \boldsymbol{b}^{(\mu)}(a_i)\mathrm{d}a_i = \begin{pmatrix} 0 & \cdots & 0 & & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{D} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \underbrace{0 & \cdots & 0}_{i \text{ times}} & & \underbrace{0 & \cdots & 0}_{n-1-i \text{ times}} \end{pmatrix} \tag{21}$$

with

$$\boldsymbol{D} = \frac{1}{36}\cdot\begin{pmatrix} \frac{1}{7} & \frac{129}{140} & \frac{3}{7} & \frac{1}{140} \\ \frac{129}{140} & \frac{297}{35} & \frac{933}{140} & \frac{3}{7} \\ \frac{3}{7} & \frac{933}{140} & \frac{297}{35} & \frac{129}{140} \\ \frac{1}{140} & \frac{3}{7} & \frac{129}{140} & \frac{1}{7} \end{pmatrix}, \qquad \text{for } \mu = 0$$

$$\boldsymbol{D} = \frac{1}{4}\cdot\begin{pmatrix} \frac{1}{5} & \frac{7}{30} & -\frac{2}{5} & -\frac{1}{30} \\ \frac{7}{30} & \frac{17}{15} & -\frac{29}{30} & -\frac{2}{5} \\ -\frac{2}{5} & -\frac{29}{30} & \frac{17}{15} & \frac{7}{30} \\ -\frac{1}{30} & -\frac{2}{5} & \frac{7}{30} & \frac{1}{5} \end{pmatrix}, \qquad \text{for } \mu = 1$$

$$\boldsymbol{D} = \begin{pmatrix} \frac{1}{3} & -\frac{1}{2} & 0 & \frac{1}{6} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{6} & 0 & -\frac{1}{2} & \frac{1}{3} \end{pmatrix}, \qquad \text{for } \mu = 2.$$

ACKNOWLEDGMENT

This work was supported by the German Federal Ministry for Economic Affairs and Energy under grant 0324125C.

REFERENCES

[1] M. Vogt, N. Müller, and R. Isermann, "On-Line Adaption of Grid-Based Look-up Tables Using a Fast Linear Regression Technique," *Journal of Dynamic Systems, Measurement and Control*, pp. 732–739, 2004.

[2] P. H. Eilers and B. D. Marx, "Flexible Smoothing with B-splines and Penalties," *Statistical Science*, pp. 89–121, 1996.

[3] L. Biagiotti and C. Melchiorri, *Trajectory Planning for Automatic Machines and Robots*. Berlin Heidelberg: Springer Verlag, 2008.

[4] P. Dierckx, *Curve and Surface Fitting with Splines*. New York: Oxford University Press, 1993.

[5] C. H. Reinsch, "Smoothing by Spline Functions," *Numerische Mathematik 10*, pp. 177–183, 1967.

[6] I. J. Schoenberg, "Spline functions and the problem of graduation," in *Proceedings of the National Academy of Sciences, Vol. 52*, 1964, pp. 947–950.

[7] C. de Boor, *A Practical Guide to Splines*. New York: Springer Verlag, 1978.

[8] S. Gros, R. Quirynen, and M. Diehl, "An Improved Real-time Economic NMPC Scheme for Wind Turbine Control Using Spline-Interpolated Aerodynamic Coefficients," in *53rd IEEE Conference on Decision and Control*, Los Angeles, USA, 2014, pp. 935–940.

[9] S. Gros and A. Schild, "Real-time Economic Nonlinear Model Predictive Control for Wind Turbine Control," *International Journal of Control*, pp. 1–14, 2017.