EFFICIENT KNOT OPTIMIZATION FOR ACCURATE B-SPLINE-BASED DATA APPROXIMATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Raine Yeh

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Xavier M. Tricoche, Chair

Department of Computer Science

Dr. Voicu S. Popescu

Department of Computer Science

Dr. Elisha Sacks

Department of Computer Science

Dr. Tom Peterka

Argonne National Laboratory

Dr. Alex Pothen

Department of Computer Science

Approved by:

Dr. Kihong Park

Department of Computer Science

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Yeh, Raine Ph.D., Purdue University, December 2020. Efficient Knot Optimization for Accurate B-spline-based Data Approximation. Major Professor: Xavier Tricoche.

Many practical applications benefit from the reconstruction of a smooth multivariate function from discrete data for purposes such as reducing file size or improving analytic and visualization performance. Among the different reconstruction methods, tensor product B-spline has a number of advantageous properties over alternative data representation. However, the problem of constructing a best-fit B-spline approximation effectively contains many roadblocks. Within the many free parameters in the B-spline model, the choice of the knot vectors, which defines the separation of each piecewise polynomial patch in a B-spline construction, has a major influence on the resulting reconstruction quality. Yet existing knot placement methods are still ineffective, computationally expensive, or impose limitations on the dataset format or the B-spline order. Moving beyond the 1D cases (curves) and onto higher dimensional datasets (surfaces, volumes, hypervolumes) introduces additional computational challenges as well. Further complications also arise in the case of undersampled data points where the approximation problem can become ill-posed and existing regularization proves unsatisfactory.

This dissertation is concerned with improving the efficiency and accuracy of the construction of a B-spline approximation on discrete data. Specifically, we present a novel B-splines knot placement approach for accurate reconstruction of discretely sampled data, first in 1D, then extended to higher dimensions for both structured and unstructured formats. Our knot placement methods take into account the feature or complexity of the input data by estimating its high-order derivatives such that the resulting approximation is highly accurate with a low number of control points. We

demonstrate our method on various 1D to 3D structured and unstructured datasets, including synthetic, simulation, and captured data. We compare our method with state-of-the-art knot placement methods and show that our approach achieves higher accuracy while requiring fewer B-spline control points. We discuss a regression approach to the selection of the number of knots for multivariate data given a target error threshold. In the case of the reconstruction of irregularly sampled data, where the linear system often becomes ill-posed, we propose a locally varying regularization scheme to address cases for which a straightforward regularization fails to produce a satisfactory reconstruction.

1. INTRODUCTION

In the past decades, due to rapid advances in data collection technology and high-performance computer architectures, the amount of scientific data produced by physical experiments and numerical simulation has been growing at an unprecedented rate. With this growth comes the need to efficiently manage, analyze, and visualize the data. Therefore, it has become essential to adopt data representations that are optimized for storage, data retrieval, and post-processing. However, data are often produced or acquired in a format that is suitable for the specific measure processes and numerical simulations. These initial data representation formats are often inefficient for storage or not optimal for the analysis tasks. Subsequent steps of the data life cycle can benefit from replacing the initial data representation with one that is instead optimized for the tasks in the later pipeline. Such data representation should satisfy several important properties:

1. Is accurate - captures the original data such that subsequent analysis produces equivalent results as the original.

2. Is storage efficient - representing the data with the least storage cost.

3. Supports fast random access - ability to locate the data of interest quickly.

4. Permits efficient data analysis - common computational tasks, such as interpolation, evaluating properties like derivatives or curvatures, etc., can be done easily.

Irrespective of the chosen data model, its application requires that the model is fitted to the original data.

The problem of data fitting is a widely studied subject in approximation theory. Data fitting falls into one of two categories: interpolation, which is an exact fit to the

source data, or approximation, which produces a model close to the data while providing other benefits such as smoothing, denoising, or compression. Both interpolation or approximation are used for reconstruction, which yields a continuous representation of the original data. In our work, we focus on the approximation problem, though we use the words "approximation" and "reconstruction" interchangeable to refer to the fitted model. In this case, fitting a data model typically consists of approximating the input data within a prescribed margin of error while reducing the model size, or the memory used. Typically, a data model with few parameters is easily optimized to fit a source data, whereas more complicated data models can be computationally expensive to apply, sometimes even intractable.

While many data representation models have their pros and cons, we focus on multivariate B-splines in our work. A B-spline is a piecewise polynomial function with a knot vector that defines where the polynomial pieces meet. High-order nonuniform tensor product B-splines have been widely used for curve and surface representation for their flexibility and geometric intuition in computer-aided design and computer graphics, and later expanded to wider usage in computational science. B-splines meet the aforementioned requirements as a suitable data fitting model while providing additional benefits. A well-chosen set of knots can produce a set of B-splines basis functions that represent data with high accuracy while reducing storage space of the original data. Existing methods for evaluating B-spline basis function are extremely fast, thereby enabling highly efficient reconstruction in the post-processing analytic pipeline. High-order B-splines readily support arbitrarily smooth data reconstruction and high-order derivative evaluations. Existing storage schemes that transform discrete data points to a compressed format to reduce storage space would need to undergo a decompression step in order for the data to be used for analysis. A B-spline model, on the other hand, is designed to enable most spatio-temporal analyses without reverting back to the original discrete layout, while usually occupying less storage space. This provides an advantage over other data compression techniques where evaluations and interpolations are delayed by a necessary decompression step. The

choice of B-splines as the basis for reconstruction provides additional advantages such as local support, partition of unity, strong convex hull property, geometric intuition of control points, and easy high-order derivative evaluation.

While the original usage of B-splines focused on univariate curves, it was subsequent extended to the multidimensional setting via tensor product. Tensor product B-splines yield a structured grid that is a tessellation of space by rectangles in 2D, cuboids in 3D, and so on. Compared to unstructured grids that exhibit arbitrary connectivity patterns with arbitrary cell types, structured grids are highly storage-efficient due to the implicit connectivity between elements. The nonuniformity of B-splines brings about varying grid spacing, allowing for flexible spatial resolution.

All in all, the choice of tensor product B-spline as a reconstruction basis provides many advantages. However, utilizing B-splines as a general-purpose data representation presents significant challenges. In our work, we focus on the challenge of finding an efficient and accurate fitting of various source data onto B-splines models. To find a B-spline fit of an input data, the first challenge lies in the proper selection of the parameters of the B-spline model to effectively approximate the considered data. Parameters to construct an approximating B-spline function include the degree of B-splines, the underlying grid resolution, and the grid coordinates, also called the knot vectors. While the knot vectors define the local complexity of the B-spline reconstruction, the control coefficients, or control points, determines the shape of the reconstruction. A well-selected set of B-spline parameters produces an accurate approximation of the input data while minimizing the number of knots, or the total number of control points used, which is the main contributor to the storage size of a B-spline model. If a set of knot vectors are chosen, the control coefficients of the B-splines are typically computed as a least squares solution. Because the knot vectors of a B-spline defines the resolution of the structured grid it lies on and the shape of the underlying B-spline basis, nonuniform B-splines, where knots are not placed in an equidistant manner, offers much higher flexibility for the shape formed by the resulting B-splines based on the locations of the knots. It is, therefore, no surprise

that the choice of the knot vectors has a vast influence on the shape of the resulting B-spline reconstruction accuracy, leading to the problem of knot optimization.

Given the polynomial order of the B-spline, knot optimization consists of finding the placement of knots for a B-spline (using as few knots as possible) that achieves the required accuracy. Existing knot placement methods propose various heuristics, optimization, or artificial intelligence-based methods. Unfortunately, these methods suffer from problems such as requiring extensive trial-and-error, relying on heuristics that yield suboptimal results, imposing restrictions on the order of the B-spline used, or limiting the dimension of the input data. Other methods become prohibitively expensive with the growth of the dataset size or dimension, or break down with an irregular sampling of the input dataset.

Furthermore, in the case of input data with high spatial variations in point density, the resulting linear least squares system often becomes ill-posed. A widely used method to overcome this problem is to regularize the linear system with additional constraints such as smoothness. While the additional constraint addresses the ill-posedness of the system, they typically reduce the accuracy of the fit, and in particular can blur out sharp features in the reconstruction. Because of this, balancing smoothness and accuracy is a challenging problem.

The target of this dissertation is to present solutions for the challenges posed by the reconstruction of geometric or time-series data through a B-spline model, using properties of the input data for an effective heuristic approach. We propose a novel knot placement method for accurate B-splines reconstruction of datasets in various layouts and dimensions. Our approach makes use of high-order derivatives of the input data to construct a novel feature measure of the input data, which is used to place knots that result in high accuracy approximation with a low number of knots. We develop our approach to apply for various synthetic, simulated, and captured data of structured and unstructured format ranging from 1D to 3D. We compare with state-of-the-art knot placement methods and show that our method achieves higher accuracy while requiring fewer B-spline control points for the various datasets.

In the case of reconstruction for irregularly sampled data, where the inverse problem becomes ill-posed, we further propose a locally adjusted variational formulation when a straight-forward regularization fails to produce a quality reconstruction. We show that this solution accurately reconstructs details in the data while maintaining the desired smoothness.

In Chapter 2, we give a mathematical overview of the fundamentals of B-splines and the data approximation problem.

In Chapter 3, we present a framework for constructing a B-spline approximation to an input data curve. The derivatives of the input data are estimated via the finite difference method. A feature function of the curve is generated from the estimated derivatives that dictate local knot density in the knot placement step. Given a choice of the number of knots, a knot vector is generated with the guidance of the feature function, and the resulting B-spline curve is generated by solving a linear least square system for the B-spline control coefficients. We demonstrate the effectiveness of our knot placement compared with state-of-the-art methods for a variety of 1D datasets. We also discuss the choice of the number of knots guided by the integral of the feature function.

In Chapter 4, we extend the previous knot placement method on multidimensional data using tensor product expansion of the B-spline basis. We propose derivative estimation methods for high dimension dataset using finite difference and moving least square methods. As a knot vector is needed for each dimension of the input dataset, we propose a method for estimating the optimal number of knots in each dimension. We compare our multidimensional knot placement method with the state-of-the-art for a variety of 2D and 3D datasets.

In Chapter 5, we consider the problem of B-spline reconstruction for input with contrasting point density. Using the previously proposed knot placement on these kinds of inputs raises the question of efficient derivative evaluation and numerical issue of the ill-posed linear system when solving for the control coefficients. We propose an octree-based evaluation scheme that naturally leads to the construction of the feature

function of the dataset required for our knot placement method. Furthermore, we propose an improvement to the commonly used regularization technique for ill-posed systems like this. We demonstrate the improvements in our regularization approach for generating smooth reconstruction while preventing an over-smoothing effect that often results from these types of regularization.

2. MATHEMATICAL FOUNDATIONS

This chapter covers the mathematical foundations that are relevant to the work presented in this dissertation. We start with review of the basics of tensor product B-spline and its derivative in Section 2.1, then onto least-squares approximation in Section 2.2.

2.1 B-splines

We assume basic familiarity with the concepts of B-splines, and refer interested readers to Farin [1] for more in-depth material on B-splines.

A B-spline curve of order p is a piecewise polynomial function of order p (degree $p-1$) with n control coefficients, and is defined by

$$C(u) = \sum_{i=1}^{n} N_{i,p}(u)\boldsymbol{c}_i, \quad u \in [\xi_p, \xi_{n+1}], \tag{2.1}$$

where $C : \mathbb{R} \to \mathbb{R}^G$ is the B-spline curve at parameter location u, $\boldsymbol{c}_i \in \mathbb{R}^G$ are the control coefficients, and $N_{i,p}$ are the p^{th} order B-spline basis functions defined over the knot vector $\boldsymbol{\xi} = \{\xi_1, \xi_2, \ldots, \xi_{n+p}\}$. A B-spline curve with n control coefficients has $n+p$ knots.

We define our B-spline knot vector with clamping, so the first and last p knots are the same: $\xi_1 = \xi_2 = \cdots = \xi_p$, and $\xi_{n+1} = \xi_{n+2} = \cdots = \xi_{n+p}$. The B-spline basis function is defined recursively as

$$\begin{aligned} N_{i,1}(u) &= \begin{cases} 1 & \text{if } \xi_i \le u < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \tfrac{u-\xi_i}{\xi_{i+p}-\xi_i} N_{i,p-1}(u) + \tfrac{\xi_{i+p+1}-u}{\xi_{i+p+1}-\xi_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{2.2}$$

The 1-dimensional B-spline curve is extended to D dimensions via tensor product of the basis function. Let $N^{(d)}_{i_d,p}(u_d)$ be the 1D basis function of order p in dimension d

with n_d control coefficients and is associated to knot vectors $\boldsymbol{\xi}^{(d)} = \left\{\xi_1^{(p)}, \ldots, \xi_{n_d+p}^{(p)}\right\}$. In this work we assume the 1D basis functions are all of the same order, although this is not strictly required and is trivial to relax. The multivariate basis function $N_{i,p} : \mathbb{R}^D \to \mathbb{R}$ is the tensor product of the 1D basis functions

$$N_{i,p}(\boldsymbol{u}) = \prod_{d=1}^{D} N_{i_d,p}^{(d)}(u_d) \qquad \boldsymbol{u} = [u_1, \ldots, u_D] \tag{2.3}$$

Here the index i goes from 1 to $\prod_{d=1}^{D} n_d$ and covers all combinations of $i_d = 1, \ldots, n_d$ for $d = 1, \ldots, D$. The multivariate B-spline $C : \mathbb{R}^D \to \mathbb{R}^G$ is then defined similar to the 1D version, but with the multivariate basis function

$$C(\boldsymbol{u}) = \sum_{i=1}^{n} N_{i,p}(\boldsymbol{u})\boldsymbol{c}_i, \quad u_d \in \left[\xi_p^{(d)}, \xi_{n+1}^{(d)}\right] \tag{2.4}$$

with $n = \prod_{d=1}^{D} n_d$ control coefficients and D knot vectors.

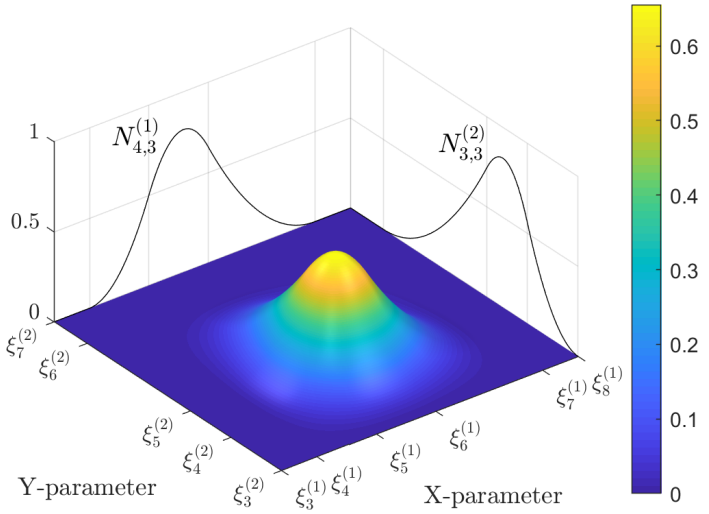


Fig. 2.1.: An order-3 (quadratic) 2D B-spline basis constructed using 2 1D basis functions.

Figure 2.1 shows an example of a tensor product B-spline surface with $D = 2$, constructed with 2 knot vectors, $\boldsymbol{\xi}^{(1)}$ and $\boldsymbol{\xi}^{(2)}$, of length 10 and 11 respectively. The knots shown in the figure are only the non-repeating knots and not the clamped knots.

The derivatives of an order-p B-spline curve C is another B-spline curve of order $p-1$ on the original knot vector

$$\frac{\mathrm{d}}{\mathrm{d}u}C = \sum_{i=1}^{n} N_{i,p-1}(u)\boldsymbol{c}'_i \tag{2.5}$$

with a new set of control points $\boldsymbol{c}'$

$$\boldsymbol{c}'_i = \frac{p}{\xi_{i+p+1} - \xi_{i+1}}\left(\boldsymbol{c}_{i+1} - \boldsymbol{c}_i\right) \tag{2.6}$$

An order-p B-spline curve will have discontinuous order $p-1$ derivatives, assuming no knot multiplicity.

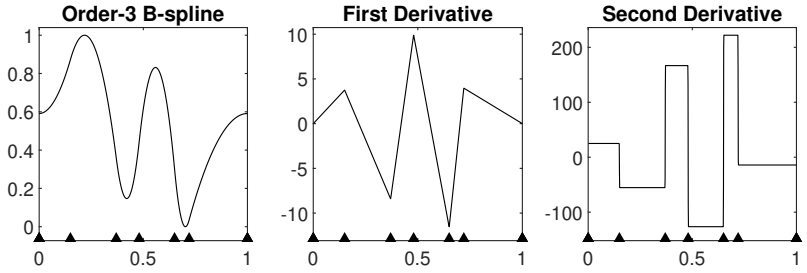


Fig. 2.2.: Points sampled from a quadratic (order-3) B-spline and the first two derivatives. The knots, represented by black triangles, mark the derivative's discontinuities.

Figure 2.2 shows an order-3 B-spline and its first two derivatives, with knots indicated by triangles at the bottom. The first derivative is piecewise linear, and the second derivative is piecewise constant.

2.2 Least Squares Approximation

In this section, we describe the problem of least-squares data approximation using B-splines.

We consider a sequence of m input data points with values $Q = \left\{\boldsymbol{q}_i : \boldsymbol{q}_i \in \mathbb{R}^G\right\}_{i=1}^{m}$ and parameters $\Omega = \left\{\boldsymbol{u}_i : \boldsymbol{u}_i \in \mathbb{R}^D\right\}_{i=1}^{m}$, where each $\boldsymbol{u}_i$ is a parameter for $\boldsymbol{q}_i$. Data parameter can be inferred from the input dataset, or generated using various

parameter optimization methods (e. g., [1,2]). In this work, we assume the input data parameters are given for our datasets.

The B-spline approximation problem is to find a B-spline $C : \mathbb{R}^D \to \mathbb{R}^G$ of order p that approximate the input points Q at parameter locations Ω

$$C(\boldsymbol{u}_i) = \sum_{j=1}^{n} N_{j,p}(\boldsymbol{u}_i)\boldsymbol{c}_j \approx \boldsymbol{q}_i \qquad i = 1, \ldots, m \tag{2.7}$$

where the unknowns are the control coefficients $\boldsymbol{c}_j$ and the knot vectors $\boldsymbol{\xi}$. If the knot vector are known, then the approximation problem becomes a linear problem, expressed as a rectangular linear system

$$\mathbf{A}\boldsymbol{x} \approx \boldsymbol{b} \tag{2.8}$$

where $\boldsymbol{x}$ is an n-vector of unknown control coefficients $\boldsymbol{c}_j$, $\mathbf{A}$ is an $m \times n$ matrix containing the basis of the B-spline at each input parameter locations, and $\boldsymbol{b}$ is the m-vector of input values $\boldsymbol{q}_i$. The resulting A matrix is sparse due to the local support of the B-spline basis.

Typically, the resulting system $\mathbf{A}$ is rectangular (i. e., $m > n$). The linear system (2.8) can be solved in the least square sense, which minimizes the 2-norm of the approximation error:

$$\underset{\boldsymbol{c}}{\operatorname{argmin}} \sum_{i=1}^{m} \left\| \boldsymbol{q}_i - C(\boldsymbol{u}_i) \right\|_2^2 \tag{2.9}$$

If the rank of $\mathbf{A}$ is less than n, the number of columns in $\mathbf{A}$, then the resulting solution $\boldsymbol{x}$ is the minimum norm least-squares solution.

In this work, we measure the accuracy of an approximation using normalized maximum error

$$E_{\max} = \frac{1}{Q_{\text{rng}}} \max_i \left\| \boldsymbol{q}_i - C(\boldsymbol{u}_i) \right\|_2 \tag{2.10}$$

and normalized root mean squared error

$$E_{\text{RMS}} = \frac{1}{Q_{\text{rng}}} \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left\| \boldsymbol{q}_i - C(\boldsymbol{u}_i) \right\|_2^2} \tag{2.11}$$

where $Q_{\text{rng}} = \max_i \left\| \boldsymbol{q}_i \right\|_2 - \min_i \left\| \boldsymbol{q}_i \right\|_2$ is the range of the 2-norm of the data. Other error metrics can be used without loss of generality.

3. FEATURE-GUIDED KNOT PLACEMENT FOR B-SPLINE CURVE APPROXIMATION

This chapter tackles the problem of knot vector optimization for B-spline approximation of 1D signals and parametric curves in two or more dimensions, including the selection of the number of knots, and the location of these knots. We review a number of 1D knot optimization solutions proposed in the literature, and present a novel method that advances the state-of-the-art in terms of efficiency and/or accuracy. The proposed method allows for B-spline of arbitrary order, and automatically determines a knot vector that achieves high approximation quality. At the core of our approach is a feature function definition that quantifies the amount and spatial distribution of geometric details in the input curve by estimating its derivatives. Knots are then selected in such a way as to evenly distribute the feature contents across their intervals. A comparison to the state-of-the-art for a wide variety of curves shows that our method is faster and achieves more accurate reconstruction results, while typically reducing the number of necessary knots.

3.1 Introduction

Curve fitting using B-splines is a fundamental problem in many applications such as computer-aided design (CAD), geometric modeling, and reverse engineering [3, 4]. High-accuracy fitting is also being explored in data analysis and compression for large scale simulations [5]. The problem of B-spline curve fitting involves finding a B-spline curve that minimizes the least squares error between a sequence of input data points and the fitted spline [6, 7]. In this type of fitting problem, variables include the order of the B-spline, the number of knots, the knot locations, and the control point values.

Often, a uniform knot vector with a predetermined number of knots is used. Fixing the knot vector and optimizing only the control points reduces the B-spline fitting problem to a linear least squares problem. However, using uniform knots may fail to capture details of the input dataset. In contrast, solving for the knot vector in addition to the control points can improve the fitting result dramatically [8], leading to the problem of knot optimization.

Knot optimization consists of finding the placement of as few knots as possible for a B-spline curve that fits some desired approximation error criterion. This is a challenging problem for two reasons. First, the unknown number and locations of knots result in a large and nonlinear optimization problem, which is computationally difficult. Second, analytic expressions for optimal knot locations, or even for general characteristics of optimal knot distributions for a desired error criterion, are not easy to derive [8].

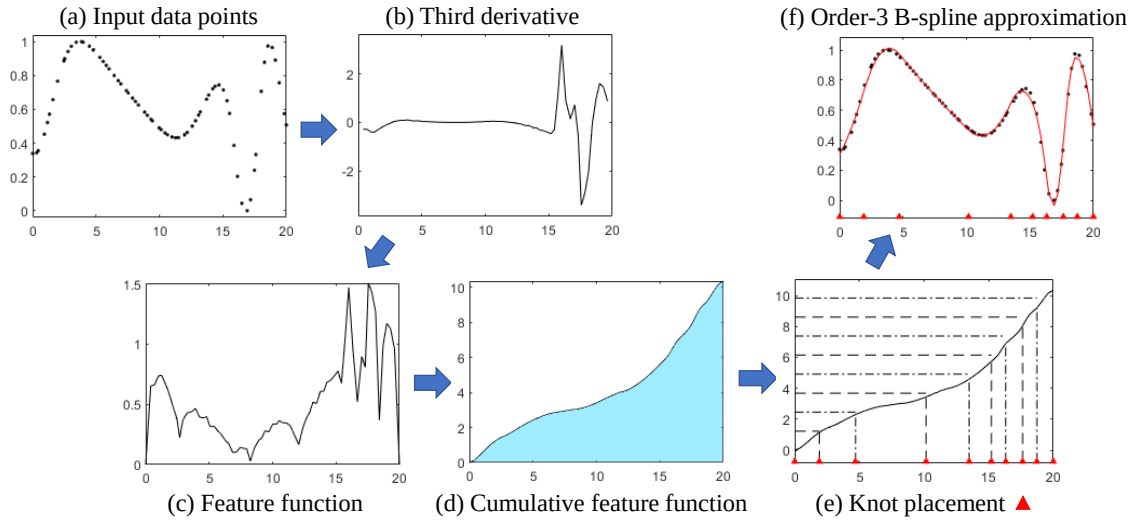


Fig. 3.1.: Overview of our method using order-3 B-spline to approximate an input curve.

In this work, we use the derivatives of the input data to calculate a feature function that captures the amount and distribution of detail in the data, where higher feature values indicate that a higher knot density is needed to capture the detail and reach

the desired approximation error tolerance. Using the cumulative distribution function (CDF) of the feature function, interior knots are distributed, such that higher feature values result in more knots. This approach affords the fitted B-spline more flexibility in those locations, thereby reducing the approximation error. Figure 3.1 shows an overview of our knot placement method for an approximation using an order-3 B-spline. Given an input dataset (Figure 3.1a) and its third derivative (Figure 3.1b), our method calculates a feature function (Figure 3.1c) using the third derivative, and the CDF of the feature function (Figure 3.1d). Knot locations, indicated by the red triangles, are determined by a set amount of variation of the feature function (Figure 3.1e). The approximation is solved using the resulting knot vector, successfully capturing the detail of the input data points (Figure 3.1f).

Our approach works for 1-dimensional input data and parametric curves in 2 or higher dimensions. It is fast and produces high accuracy approximation for a wide range of input data that are smooth and sampled densely enough for high-order derivatives to be estimated from the data. The order of the derivatives depends on the order of the B-spline used for the approximation. The knot placement method works for a B-spline approximation of any order, with the resulting approximation error close to a user-supplied target error.

The remainder of this chapter starts with a presentation of related work in Section 3.2. Then, we present our method in Section 3.3, and compare it with prior work in Section 3.4. Finally, conclusions are drawn, and possible extensions are discussed in Section 3.5.

3.2 Related Work

Knot optimization for B-spline curve fitting is a well studied topic. Many approaches can be found in the literature for both interpolation and approximation. Initial approaches consider only the parameter values of the input data into account. Piegl and Tiller [6] proposed their new knot placement method (NKTP) that places

knots with averages of representative parameters for groups of input points. This leads to a stable system of equations and a uniform-like distribution of knots along the parameter domain. This method can be used for interpolation or approximation. Interpolation occurs when the number of control points is the same as the number of input points, then the resulting interpolating B-splines would overlap with all the input data points, whereas using fewer control points than the number of input points would result in an approximating B-spline curves. When NKTP is used for approximation, it may not capture details of the data because knots are placed using only the parameter information, and not the data values.

An improvement to that problem is to iterative refine knot spans that contains the highest error. Liang *et al.* [9] described an iterative knot insertion (IKI) method that starts with the fewest knots possible, finds knot segments whose approximation error is higher than a given error tolerance, and adds a new knot in the middle of these segments. Although this refinement method typically provide good approximation result, the computational time is high due to its iterative nature.

Dung and Tjahjowidodo [10] proposed a fast method for knot placement that locally searches for the largest knot spans under an error threshold using binary search. This method is fast and uses few knots, but often produces a discontinuous approximation. On the opposite direction of the iterative process are the knot removal techniques, which start with a set of dense knots, and iteratively remove knots while maintaining the approximation tolerance [11, 12]. These class of methods can still be resource consuming, and often terminate early without removing redundant knots.

Jupp [8] and Loach and Wathen [13] describe local optimization techniques that transform the constrained optimization problem into an unconstrained problem; then a local gradient-based or Gauss-Newton method is employed for minimization. Global optimization can avoid the drawbacks of local methods, but it is computationally more expensive [14].

Kang *et al.* [15] and Loock *et al.* [16] treat knot placement as a convex optimization problem, where the norm of jump of the $(p-1)^{\text{th}}$ derivative of a p-order B-spline is

minimized. These optimization methods compute the number and positions of the knots simultaneously and achieve low approximation error with few knots, but are typically computationally more expensive than other approaches.

A machine learning approach using support vector machines for knot placement is described by Laube *et al.* [17]. The performance of their approach depends strongly on the training dataset, limiting the applicability to a different dataset than the training dataset. Yuan *et al.* [18] select knots by extracting optimal subsets from a multiresolution B-spline basis using regression analysis.

There also exists a body of work using genetic algorithms for knot vector optimization [19–23], meta-heuristics such as the firefly algorithm [24], and elitist clonal selections [25]. Such methods are typically computationally expensive, and often produce globally suboptimal solutions.

Another body of literature proposes heuristic methods that use specific properties of the input dataset to guide knot placement. Curvature, in particular, is a widely-used criterion in heuristic methods. Park and Lee [26] select knots using dominant points, which are points of interest of the input dataset. Initial dominant points are set to points of high curvature; then, additional dominant points are added in segments of high approximation error using the input data curvature. Li *et al.* [27] place initial knots at zero crossings of the curvature, then iteratively add new knots to balance the integral of curvature of the new knot segments. Aguilar *et al.* [28] use curvature peaks as initial knots, then iteratively add new knots or adjust existing knots to reduce curvature deviation. Razdan [29] uses curvature and arc length of the input dataset to select points of interest, and construct the B-spline approximation by interpolating those points. This interpolation, however, only used the points of interest and does not take into account the points that were not chosen, which can lead to overall higher error.

Derivatives are also used by some heuristic methods for knot placement. Corresponding techniques approximate the derivatives of the input data using a piecewise low-order polynomial function; the connecting points of the piecewise polynomial are

then used as knot locations. Tjahjowidodo *et al.* [30] find knots for a cubic B-spline approximation by using piecewise linear approximation of the second derivative of the input data. Conti *et al.* [31] find a smooth fit of noisy input data, calculate the third derivative of the smooth fit, and find the piecewise constant approximation of the derivative.

Finally, other heuristic methods use wavelet decomposition [32].

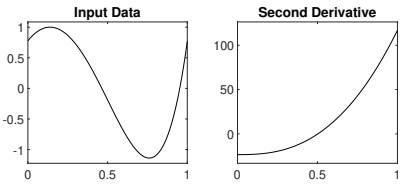
Though our work also follows a heuristic approach using the derivatives of the input dataset, it produces more accurate approximation in a shorter time compared to the existing heuristic approaches, as we will show in Section 3.4.

3.3 Methodology

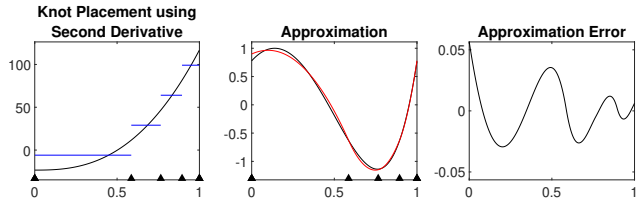
Our work is motivated by the idea that an order-p B-spline has a piecewise constant $(p-1)^{\text{th}}$ derivative, where derivative discontinuities mark the knot locations, as shown in Figure 2.2. Given a set of points sampled from an order-p B-spline, one can recover the original B-spline knots by locating the discontinuities in the $(p-1)^{\text{th}}$ derivative. In practice, the input data points will not be sampled from a B-spline, and will not, in general, have distinct discontinuities in their derivatives. Nonetheless, we show in the following that the derivative information of the input data can be used to guide the knot placement such that resulting knots align with the properties of the input data, thus yielding a better approximation.

Figure 3.2a shows an example input dataset and its gradually increasing second derivative. More knots are needed on the right side to reduce the approximation error where second derivative is steeper. A knot placement method that does not take the data complexity into account and allocates knots uniformly (such as the NKTP method [33]) will result in insufficient knots on the right side, and therefore exhibit higher error there (as shown in Figure 3.2c with the NKTP method).

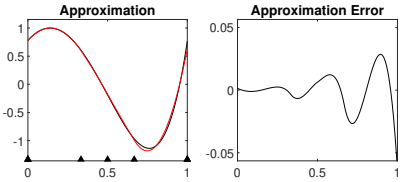
Using the $(p-1)^{\text{th}}$ derivative directly does not solve this problem either. Figure 3.2b shows a piecewise constant function approximating the second derivative



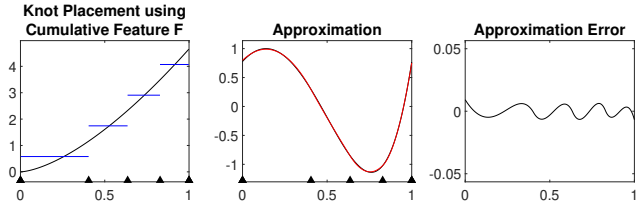
(a) Left: Input data. Right: Second derivative that gets increasingly steeper.



(b) Left: Four constant functions (blue) approximate the second derivative (black). Knots are set to be the breakpoints of the blue lines. Middle: Approximation using those knots. Right: Approximation error.



(c) Left: Approximation using NKTP knot placement method. Right: The resulting approximation error.



(d) Left: Four constant functions (blue) approximates the cumulative feature function (black). Knots are placed at the breakpoints of the blue lines. Middle: Resulting approximation. Right: Approximation error.

Fig. 3.2.: Demonstration of different knot placement methods and the respective approximation error. The approximated curve is red. The knots used are indicated by black triangles. Each approximation uses the same number of knots, varying only the location of the interior knots.

such that the maximum absolute difference between the derivative and the piecewise constant function is minimized. The knots are derived from the breakpoints of the piecewise constant approximation. Similar ideas were attempted in prior work [30,31]. The right plot in Figure 3.2b shows the approximation error, where higher error occurred on the left while most knots gather on the right side where the second derivative is steeper.

In this work, instead of using the $(p-1)^{\text{th}}$ derivative directly, we calculate a feature function (Section 3.3.2) that better captures the amount of detail present throughout the dataset. We then place knots such that each knot segment has the same integral of the feature function (Section 3.3.3). Figure 3.2d shows the approximation of the same dataset using our method. The resulting approximation exhibits an error that is evenly spread across the domain, producing a higher accuracy approximation using the same number of knots.

Following this idea, our approach for finding the knot vector for B-spline curve fitting is comprised of the following steps:

1. Calculate the derivatives of the input data.
2. Calculate a feature curve for the input data.
3. Determine a parameter that decides the number of knots to use.
4. Adjust the feature curve to avoid a rank deficient system.
5. Determine the knot vector using the feature curve.
6. Use least squares minimization to obtain a B-spline approximation of the input data.

The above steps are explained in greater detail below.

3.3.1 Derivative Calculation

We use central differences to calculate an approximation of the derivatives of the input points. Central differences is second-order accurate in the parameter spacing. With a given set of m input points $Q = \left\{q_i : q_i \in \mathbb{R}^d\right\}_{i=1}^{m}$ and parameters $U = \left\{u_i : u_i \in \mathbb{R}, u_i < u_{i+1}\right\}_{i=1}^{m}$, we define $Q^{(k)} = \left\{q_j^{(k)} \in \mathbb{R}^d\right\}_{j=1}^{m-k}$ to be the set that approximates k^{th} derivatives of the input points at parameters $U^{(k)} = \left\{u_j^{(k)} \in \mathbb{R}\right\}_{j=1}^{m-k}$. We let $Q^{(0)} = Q$, $U^{(0)} = U$, then for $k > 0$, we use central differences to find $q_j^{(p)}$

$$q_j^{(k+1)} = \frac{q_{j+1}^{(k)} - q_j^{(k)}}{u_{j+1}^{(k)} - u_j^{(k)}} \tag{3.1}$$

with parameter

$$u_j^{(k+1)} = \frac{1}{2}\left(u_j^{(k)} + u_{j+1}^{(k)}\right). \tag{3.2}$$

Note that each level of derivatives has its own set of parameters, which are midpoints of the parameters of the previous level derivatives.

Figure 3.3 shows an example of calculating the first and second derivatives of the starting segment of a dataset.

Other methods of derivative calculations can also be applied here. In fact, if the analytical derivatives are known, the proposed method will automatically benefit from the additional accuracy.

Care must be taken when calculating derivatives of noisy data because differentiation in general, and central differences in particular, are known to amplify the noise. However, the present work does not consider the specific issue of fitting noisy data, a problem for which different solutions have been proposed. For instance, Conti *et al.* [31] fit a smoothing B-spline with dense knots to the noisy input data, and estimate the derivatives using the smoothing B-spline.

3.3.2 Calculating Feature Function

The feature function $f(u)$ measures the amount of detail in the input data points, and is later used for knot placement in the B-spline approximation step. The feature

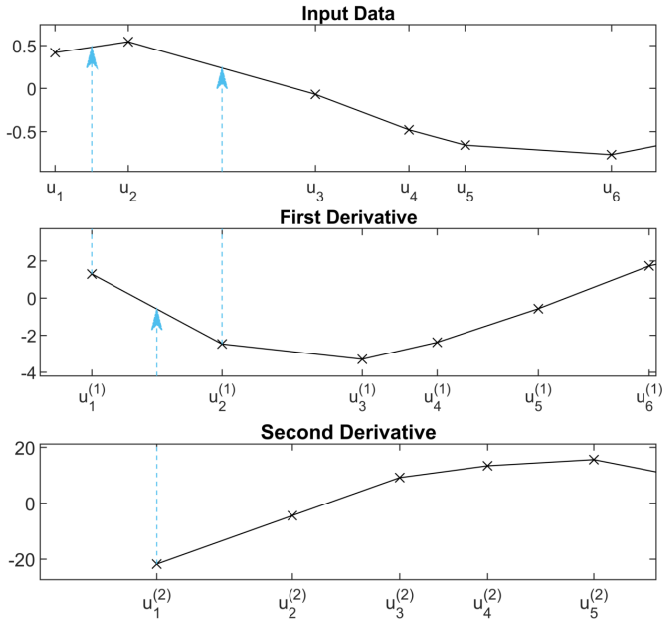
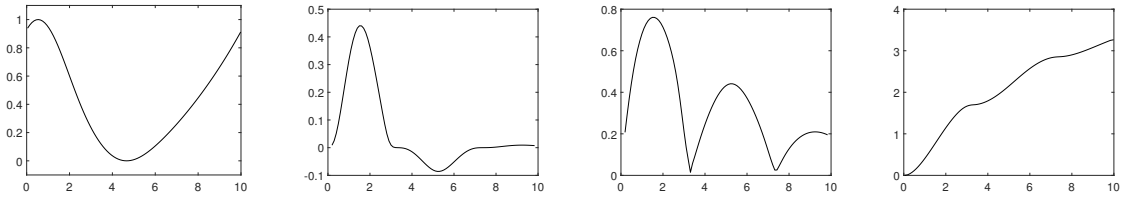


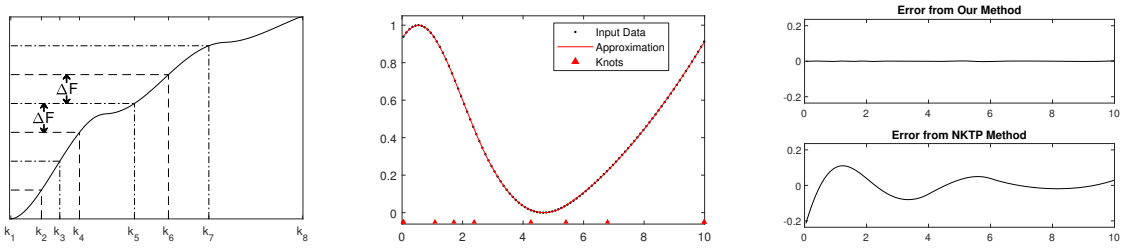
Fig. 3.3.: Example of derivative calculation. The parameter location of each derivative value is the midpoint of two points from the previous level, as indicated by the blue dashed arrows.



(a) Input data (b) Third derivative (c) Feature function f (d) Cumulative feature function F

Fig. 3.4.: Example of calculating the feature function f and the cumulative feature function F

function is defined using a set of *feature points* $\{f_i\}$, calculated from the p^{th} derivative of the input dataset via a normalization function Φ, where p is the order of B-spline used to approximate the data.



(a) Cumulative feature function and the knot placement.

(b) Resulting Approximation using 12 knots (8 unique knots).

(c) Error of approximation from our knot placement method vs. NKTP method

Fig. 3.5.: Placement of knots using the cumulative feature function F

We define the set of feature points $\{f_i\}$ at parameter locations $\{\bar{u}_i\}$, $0 \leq i \leq m-p+1$, as

$$(\bar{u}_i, f_i) = \begin{cases} (u_1, 0)\,, & i = 0 \\ \left(u_i^{(p)}, \Phi\left(\left\|q_i^{(p)}\right\|_2\right)\right), & 1 \leq i \leq m-p \\ (u_m, 0)\,, & i = m-p+1 \end{cases} \tag{3.3}$$

We take the magnitude of the p^{th} derivatives of the input data points, and use a normalization function Φ to balance the knot distribution and prevent the case in Figure 3.2b, where too many knots are allocated at steeper derivatives.

The definition of f_i for $i = 1, \ldots, m-p$ is

$$f_i = \left(\left\|q_i^{(p)}\right\|_2\right)^{1/p}. \tag{3.4}$$

Empirically, we find that defining Φ to be the p^{th} root of the p^{th} derivative produces the best error distribution. The mathematical proof behind this feature definition is provided by Lenz *et al.* based on spline theory where the approximation error is shown to be the p^{th} power of the knot span width in a neighborhood of knots [34]. The continuous feature function $f(u)$ is then defined as the piecewise linear interpolant of the set of feature measure $\{f_i\}$. In other words, for parameter u where $\bar{u}_i \leq u \leq \bar{u}_{i+1}$,

$$f(u) = \frac{u - \bar{u}_{i+1}}{\bar{u}_i - \bar{u}_{i+1}} f_i + \frac{u - \bar{u}_i}{\bar{u}_{i+1} - \bar{u}_i} f_{i+1} \tag{3.5}$$

for $i = 0, \ldots, m - p + 1$, and $f(u) = 0$ for u outside of the range $\left[\bar{u}_0, \bar{u}_{m-p+1}\right]$.

The feature function $f(u)$ represents the amount of detail at parameter location u. The higher the value of $f(u)$, the smaller the knot span at the parameter location u.

Figures 3.4a to 3.4c show the feature set calculation process for a sample input. For an order-3 B-spline approximation, the third derivative is calculated in Figure 3.4b, and the feature function is shown in Figure 3.4c. The cumulative curve of Figure 3.4d is described in the next subsection.

3.3.3 Knot Placement

We now consider the problem of how to distribute the knots using the feature function $f(u)$.

We want to place knots such that the integral of f over each knot span is the same. To do so, we calculate $F(u)$, the cumulative distribution function (CDF) of $f(u)$.

$$F(u) = \int_{-\infty}^{u} f(v)\,\mathrm{d}v, \tag{3.6}$$

which is equivalent to F being a linear interpolant of the set $\left\{F_i\right\}$ at parameter locations $\left\{\bar{u}_i\right\}$, $0 \leq i \leq m - p + 1$, where $F_0 = 0$, and

$$F_i = \sum_{j=1}^{i} f_{j,\text{trap.}} \tag{3.7}$$

for $i = 1, \ldots, m-p+1$, and $f_{j,\text{trap.}}$ is the finite integral approximation of $f(u)$ between $\bar{u}_{j-1}$ and $\bar{u}_j$ calculated using the trapezoid rule

$$f_{j,\text{trap.}} = \frac{1}{2}\Big(f_j + f_{j-1}\Big)\Big(\bar{u}_j - \bar{u}_{j-1}\Big). \tag{3.8}$$

Since F is a cumulative distribution function of the non-negative function f, F starts from zero and is non-decreasing.

Figure 3.4d shows the cumulative feature function F calculated from the feature function f shown in Figure 3.4c.

Next, we define F^{-1} as the inverse of F such that

$$F^{-1}(q) = u \quad \Leftrightarrow \quad F(u) = q. \tag{3.9}$$

The inverse function F^{-1} is well defined if the values $\{F_i\}$ are monotonically increasing, which may not be the case if there are consecutive zeroes in $\{f_i\}$. For the cases where F contains flat spots, we can add a small positive value η to the integration

$$f_{j,\text{trap.}} = \frac{1}{2}\Big(f_j + f_{j-1} + \eta\Big)\Big(u_{j-1}^{(p)} - u_j^{(p)}\Big) \tag{3.10}$$

to ensure that $\{F_i\}$ and therefore F is monotonically increasing, and F^{-1} is uniquely defined in the whole domain.

With F^{-1} defined, we can now find the knot locations. With knot clamping, the first and last knots are repeated p times. The knot vector $\boldsymbol{\xi}$ contains r unique knots, and is defined as

$$\boldsymbol{\xi} = \{\underbrace{\xi_1, \ldots, \xi_1}_{\text{p}}, \xi_2, \ldots, \xi_{r-1}, \underbrace{\xi_r, \ldots, \xi_r}_{\text{p}}\} \tag{3.11}$$

where the range of the knots is the same as the range of the input data parameter, that is, $\xi_1 = u_1$ and $\xi_r = u_m$.

We find the locations of the r unique knots $\{\xi_1, ..., \xi_r\}$ with

$$\xi_i = F^{-1}\big((i-1)\Delta F\big) \tag{3.12}$$

where ΔF is the amount of integrated feature per knot segment. The selection of ΔF determines the number of knots used and the resulting accuracy of the approximation. A smaller ΔF results in greater number of knots with shorter knot spans and a higher approximation accuracy, and conversely for larger ΔF. This knot placement ensures that each knot span has the same amount of increase in F; i. e., $F(\xi_{i+1}) - F(\xi_i) = \Delta F$ for all i.

Figure 3.5a shows the cumulative feature function F split into equal ΔF steps with horizontal dashed lines. The knot locations are shown on the x-axis. If we know r, the number of unique knots, in advance, we can calculate $\Delta F = F_{\max}/(r-1)$,

where $F_{\max}$ is the largest value in F. Otherwise, the selection of ΔF is discussed in Section 3.3.5.

After the knot vector is acquired, the fitting B-spline can be solved with Equation (2.9).

Figure 3.5b shows the approximation using the acquired knot vector, and Figure 3.5c shows the resulting approximation error of our method (top) compared with the approximation error using the NKTP method [33] (bottom).

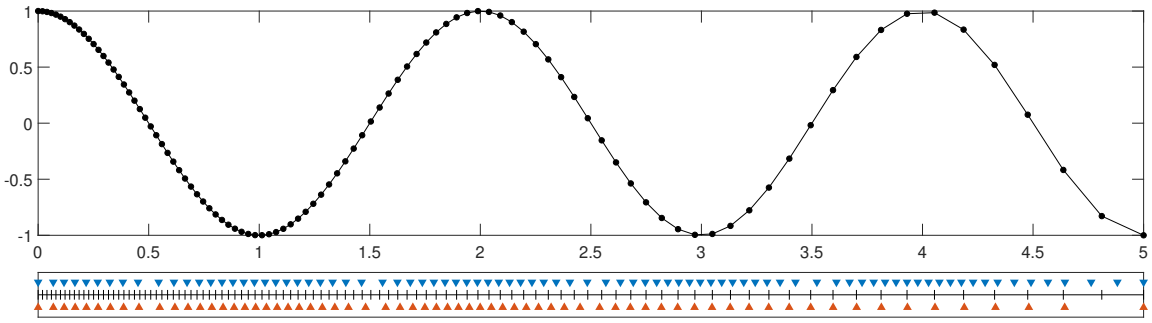


Fig. 3.6.: Input data with nonuniform spacing. Blue triangles indicate old unadjusted knots. Red triangles are adjusted knots. Black bars between the triangles indicate input spacing. At the right side the new knots (red) are adjusted to match the spacing of the input points, whereas old knots (blue) are more densely placed, causing rank deficiency in the least squares system.

3.3.4 Limiting Knot Density

There may be cases where the resulting knot vector has smaller knot spans than the input data point spacing. This can occur when some part of the cumulative feature function F increases too quickly, resulting in knots placed too close together with the given ΔF. This could result in a rank-deficient least squares system and an inefficient usage of knots.

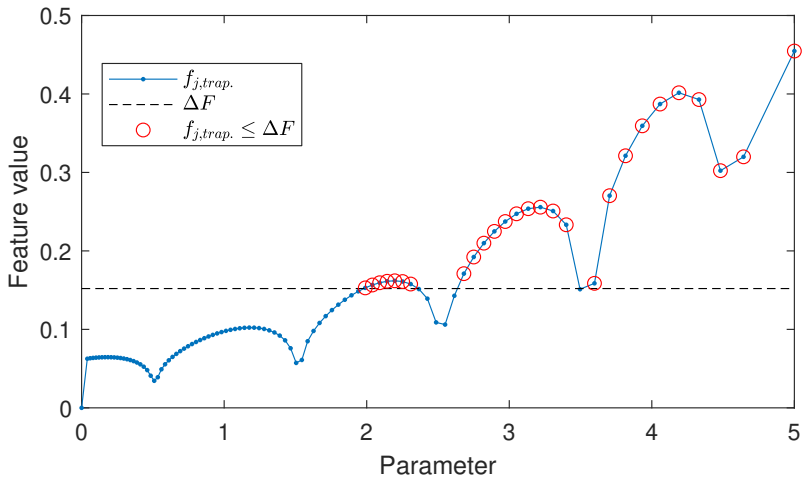


Fig. 3.7.: Finite integral approximation $f_{j,\text{trap.}}$ and the ΔF line that shows the cutoff to prevent a rank-deficient system. All $f_{j,\text{trap.}}$ above the cutoff are circled.

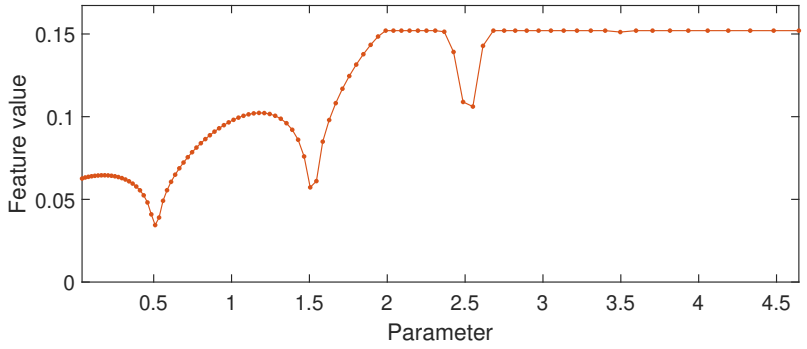


Fig. 3.8.: The plot of $\min(f_{j,\text{trap.}}, \Delta F)$.

We prevent this situation by ensuring that knots are not placed too closely to each other by imposing the condition

$$F_i - F_{i-1} \leq \Delta F \tag{3.13}$$

for all i. We achieve this by limiting F_i to be

$$F_i = \sum_{j=1}^{i} \min\left(\Delta F, f_{j,\text{trap.}}\right). \tag{3.14}$$

We then proceed to place knots as described in Section 3.3.3.

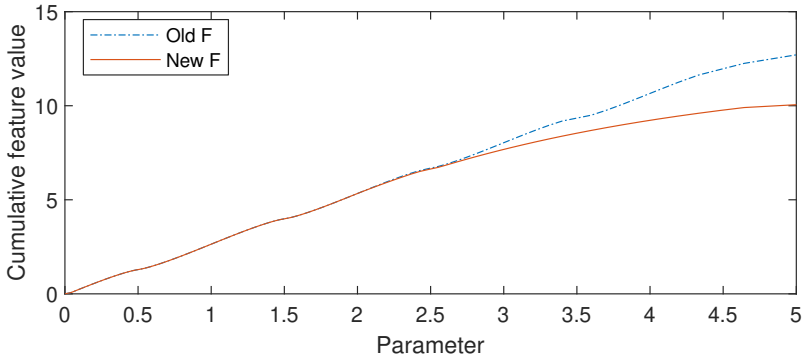


Fig. 3.9.: The old and new cumulative feature functions G diverge where $f_{j,\text{trap.}}$ exceeds ΔF.

Figure 3.6 shows an input dataset sampled from a cosine function, with sparse samples toward the right. Below the cosine plot are the input point parameters shown as vertical bars. Blue downward triangles are knots placed before the adjustment, and red upward triangles are the knots created from the adjusted F function. The pre- and post-adjustment knots match on the left side of the plot where the density of the input points is high enough for the knot density; but on the right side, the pre-adjusted knots are denser than the input points. The post-adjusted knots account for the input point spacing.

Figure 3.7 shows the finite integral approximation $\{f_{j,\text{trap.}}\}$ of the input data's feature function and the chosen ΔF. The $f_{j,\text{trap.}}$ points over the ΔF value are circled. Figure 3.8 shows the finite integral approximation $\{f_{j,\text{trap.}}\}$ limited by the ΔF value, and Figure 3.9 shows the adjusted cumulative feature function F compared with the pre-adjusted F.

3.3.5 Determining Number of Knots

During the placement of knots, ΔF determines the number of knots used. However, usually users do not know how to set ΔF or the number of knots for a desired approximation quality.

In our knot placement method, we approximate the number of knots needed for a desired target error by using regression to find the relationship between the approximation error and ΔF for a specific order-p B-spline.

We tested 10 randomly generated 1D order-9 nonuniform rational B-spline (NURBS) datasets, with random control points, weights, and number of knots ranging from 10 to 40 with randomized knot locations. The range of number of knots results in various feature functions F for each dataset. We use a range of ΔF and B-spline degrees to approximate these datasets. Figure 3.10 plots, in log-log scale, the resulting RMS error of these approximations against the ΔF value used, with a different color for each degree of B-spline used. Exponential convergence can be observed from the plot.

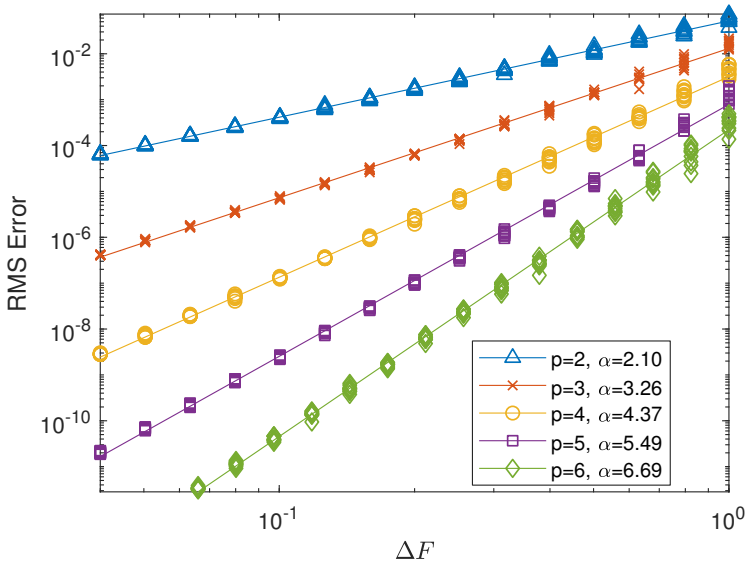
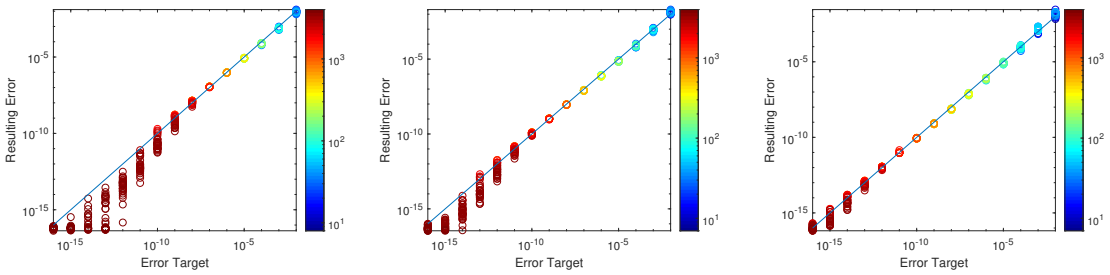


Fig. 3.10.: Plot of ΔF value against RMS error for 10 datasets with 5 different B-spline orders. Lines are fitted via linear regression in the log-log space. The slope of each line is α.

For each order of B-spline, we fitted the data points pertaining to the order in log-log space with the line

$$y = \alpha x + \beta \tag{3.15}$$

where y is the log of the RMS error, x is the log of ΔF; α is the slope of the line, and β is the y-intercept of the line for the order-p B-spline. We solve for the unknowns α and β using linear least squares to minimize the fitting error in the log-log space for each B-spline order. Then given a desired B-spline order p and the desired target approximation RMS error e, we can approximate ΔF with $\Delta F = 10^{(\log(e)-\beta)/\alpha}$.



(a) Using quadratic B-spline (b) Using cubic B-spline (c) Using order-5 B-spline

Fig. 3.11.: Target error vs. resulting error using three different B-spline orders for 32 randomly generated datasets. Number of knots used for each approximation is color-coded.

We evaluate the line model with 32 randomly generated NURBS datasets, different from the 10 datasets used in the regression. The 32 datasets are of order 7 to 10, contain 4000 points, with number of knots in the range of 10 to 60, with random knot locations, control points, and weights. We check the correlation between the input desired tolerance and the actual deviation error for order-3, 4, and 5 B-splines in the correlation plots in Figure 3.11. The $x=y$ line is drawn for reference to see how close the resulting error is to the target error. Each approximation point is colored by the number of knots used. Points below the line indicate a resulting error lower than the target error, whereas points above indicate a resulting error higher than the target error.

The resulting error matches the target error for the most part, except toward the lower error, where a faster convergence is observed for quadratic B-spline, and similarly but less noticeably for the other two higher order B-splines. The drop of

resulting error is caused by the number of knots approaching the number of input points in each data sets, resulting in interpolation of the input datasets. The lowest errors are around 10^{-16}, limited by machine precision.

The integral of the feature function f (in other words, $F_{\max}$, the maximum value in F) can be thought of as a measure of how complicated the input data are. The more complicated the input data are, the more knots should be used to achieve the same error threshold.

This is a heuristic linear regression to guide the choice of the number of knots given a desired error tolerance. It uses a small number of datasets with similar characteristics as the target data, and does not guarantee the resulting approximation error to be under the tolerance. The linear regression model can be improved upon with more sophisticated machine learning approaches that generalize to a wider variety of datasets.

3.4 Experimental Results

The performance of our knot placement algorithm was tested against six prior works. We implemented each of the other methods in MATLAB.

Table 3.1.: Table of all methods

Ref.	Label	Descriptions
	Our	Instant; derivative-guided
[6]	NKTP	Instant; parameter-only
[9]	IKI	Iterative
[30]	LinFit	Locally iterative; derivative-guided
[31]	ConFit	Locally iterative; derivative-guided
[28]	AdpCrv	Locally iterative; curvature-guided
[26]	DOM	Iterative; curvature-guided

Table 3.1 lists the algorithms, their labels used in the comparison plots, and the main idea of each method. "Instant" refers to placing knots without any iterative process. "Iterative" indicates that a repeated adjustment of the knots is done to improve the approximation. "Locally iterative" indicates that, at each iteration, a small local system is solved, whereas (globally) iterative methods solve the fitting equation (2.9) at each iteration. "Parameter-only" refers to methods that only use the parameter information to place knots. These methods do not take into account features of the input data. "Derivative-guided" methods use the derivative information to place knots, whereas "curvature-guided" methods make use of the curvature information. "Our" refers to the method presented in this chapter. Tjahjowidodo *et al.* [30] use piecewise linear polynomial to fit the second derivative, hence their method is dubbed *LinFit*. Similarly, Conti *et al.* [31] use piecewise constants to fit the third derivative, and we refer to their method as *ConFit*. The method by Aguilar *et al.* [28] is adaptive curvature guided (*AdpCrv*), and the method by Park and Lee [26] uses dominant points (*DOM*).

The IKI method [9] refines from a starting knot vector. To prevent the starting knots from influencing the result, our implementation of the IKI method starts with knots only at the end points.

The discrete derivatives discussed in Section 3.3.1 were used for our method for all datasets. Our input data are free of noise generally, but datasets with varying parameter spacing would result in noise in the derivative computation. Therefore, for some of the other methods that are more sensitive to the smoothness of the derivative and curvature calculation, we calculate the derivatives by fitting a cubic smoothing spline over the data. The fitted spline is then used to calculate the approximate derivatives and curvature of the data. We use MATLAB `csaps` function for the cubic smoothing spline. For AdpCrv and DOM methods, datasets with varying spacing (Figure 3.13a and 3.18a) and the simulation dataset (Figure 3.15a) used the smoothing weight of 10^{-5}. For the ConFit method, the smoothing weight used is 10^{-8} for all 1D datasets.

Some of the methods only work with cubic B-splines (order-4); thus, we use cubic B-splines in our experiments to match these methods. However, note that our knot placement method works for B-splines of any order.

3.4.1 Approximation Error

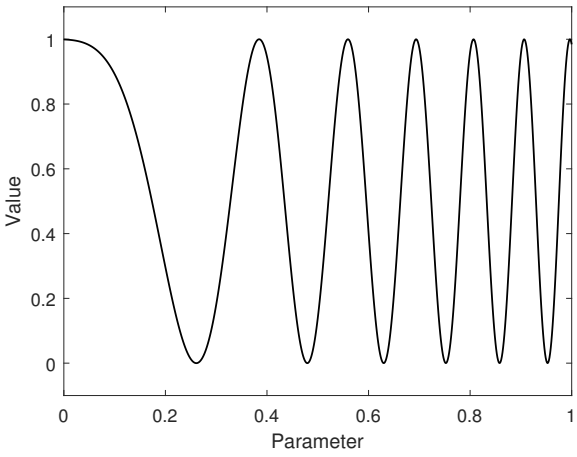
In each example case, the input points are plotted first (a), followed by the maximum error (b) and the root mean squared error (c), plotted against the number of knots used. Each method was run 15 times with different numbers of knots, and the resulting approximations were evaluated. The two exceptions are DOM and ConFit methods. Those methods add one knot at each iteration until a target number of knots is reached. Therefore, the plots for those methods show the progression of error as each knot is added.

The test cases are selected to include a range of fitting complexity, with varying amount of detail. Of the eight datasets, the first four are 1D signals, and the other four are parametric curves. Test datasets are comprised of a combination of synthetic and actual scientific data. Some of the synthetic datasets are generated using high-order NURBS with randomly distributed control points, knots, and weights, such that the data cannot be exactly represented using the approximating B-splines.

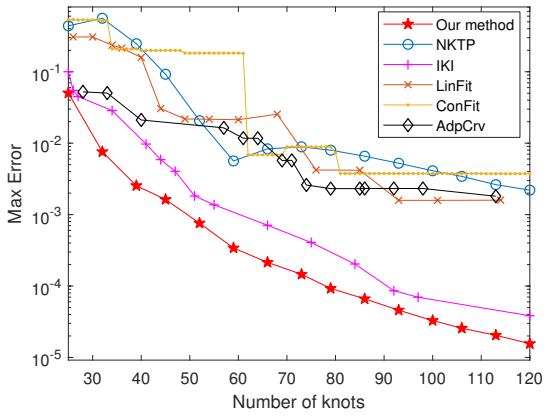
1D datasets

Figures 3.12 to 3.16 compare the approximation results for the four 1D datasets, using all methods except the DOM method, as DOM is described only for parametric curves.

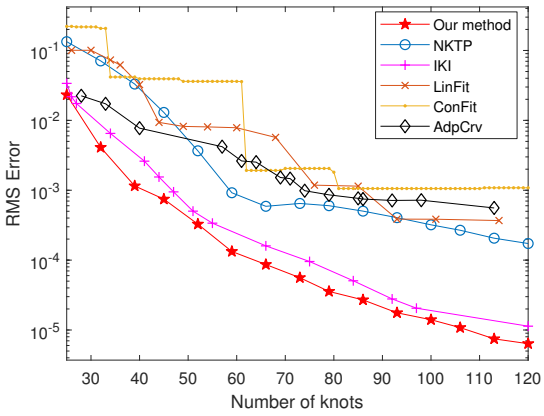
The first dataset shown in Figure 3.12a corresponds to 801 points sampled from a cosine wave with increasing frequency, resulting in progressively higher frequency oscillation toward the right side of the data. More knots need to be allocated toward the right to accurately capture the data. As can be seen from the maximum and RMS error plots (Figure 3.12b and 3.12c), our method achieves the lowest error for all tested



(a) Input 1D data



(b) Max error

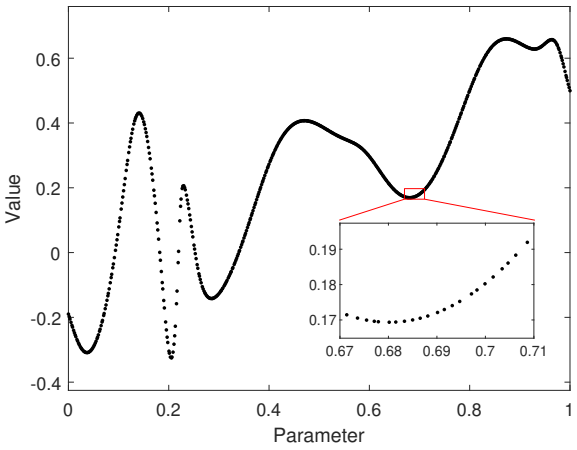


(c) Root mean squared error

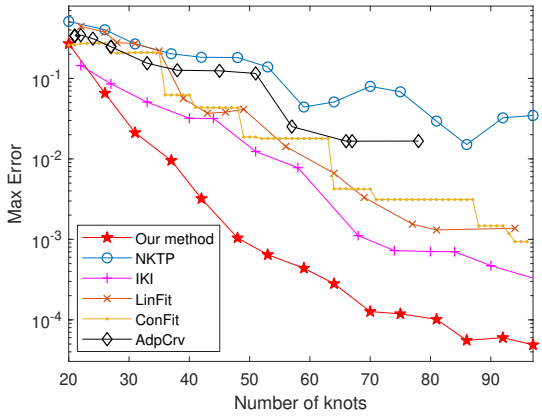
Fig. 3.12.: Methods comparison for 801 1D input points uniformly sampled from a cosine with increasing frequency.

number of knots, followed by the IKI method. The other four methods have higher error for the tested number of knots. The NKTP method does not allocate enough knots toward the right hand side to lower the error sufficiently. The drop at 54 knots for the NKTP method occurs when the knot density is high enough to fit the shape of the right-most oscillation. On the other hand, the three other heuristic methods, LinFit, ConFit, and AdpCrv methods, place too many knots toward the right side, which quickly increases the number of knots without reducing the approximation

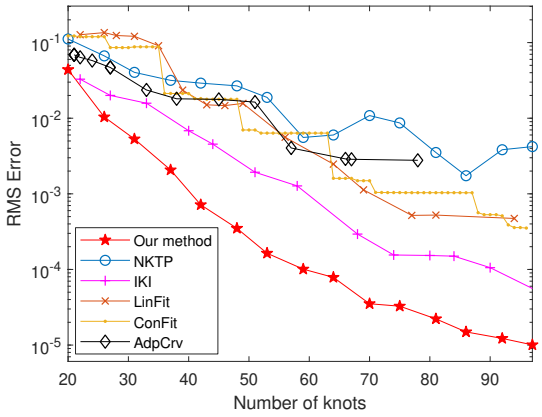
error on the left side. Sharp drops in error for the ConFit method occur when knots are added toward the right side.



(a) Input 1D data



(b) Max error



(c) Root mean squared error

Fig. 3.13.: Methods comparison for 501 1D input points sampled nonuniformly from a randomly generated NURBS curve.

The second dataset shown in Figure 3.13a corresponds to 501 points sampled from a randomly generated NURBS curve of order 6 with 30 knots. The control points, knot locations, and weights are randomly generated. The points are sampled with randomly varying point spacing, as shown in the zoom-in plot. This dataset contains a smooth curve with a sharp dip around parameter value 0.2. Figure 3.14

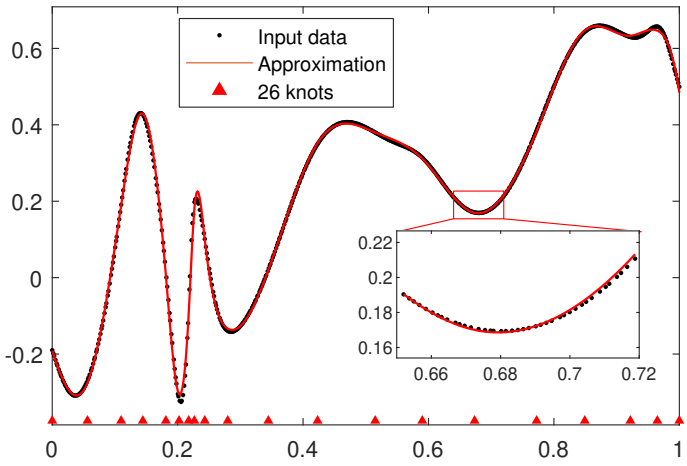
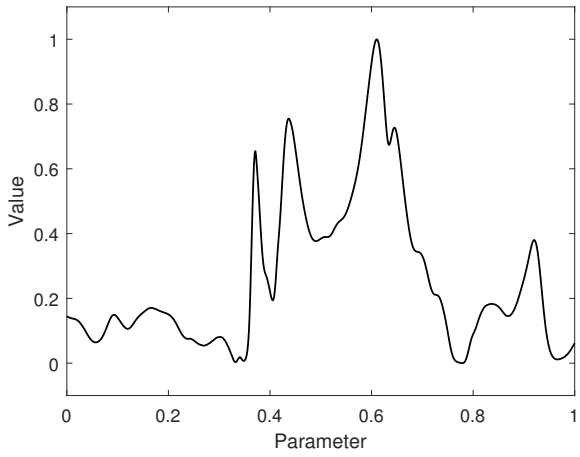


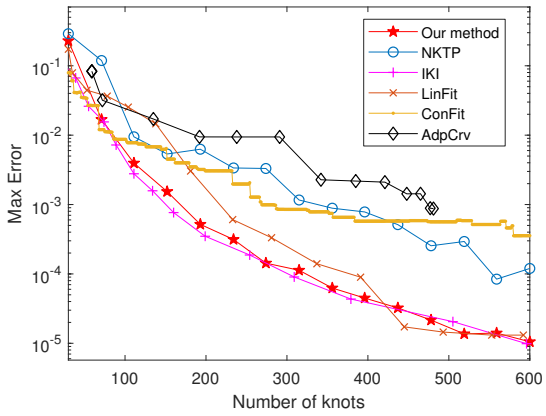
Fig. 3.14.: Approximating a NURBS curve using our knot placement method with 26 knots.

shows the approximation result using our method and the knot placement with 26 knots. Figures 3.13b and 3.13c show the approximation error of all the methods with different numbers of knots. Our method achieves the lowest approximation error for all numbers of knots. The iterative IKI method reaches the second lowest error in the range of the number of knots used, followed by the ConFit and LinFit methods. The NKTP and AdpCrv methods have the highest error for most knot numbers. Since the sampling of the input dataset is uniformly randomized, and not proportional to the amount of variation in the data, the NKTP method does not allocate more knots near the sharp dip at 0.2 parameter.

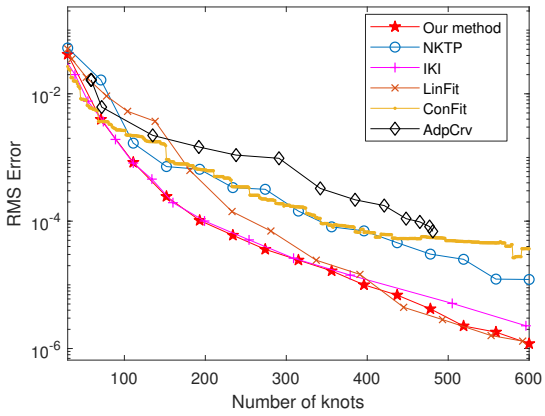
The third dataset shown in Figure 3.15a contains 704 points and is a 1D slice of a 3D dataset, measuring the magnitude of the velocity of a turbulent combustion simulation [35]. This dataset contains sharper features compared with previous datasets. Figures 3.15b and 3.15c show the approximation error for all the methods. For this dataset, our method and the IKI method achieve similar error for the same number of knots. Because of the higher frequency variation present in this dataset, the derivative is noisier compared with the previous datasets. The IKI method tends to perform better with less smooth datasets thanks to its localized refinement step. LinFit also



(a) Input 1D data



(b) Max error

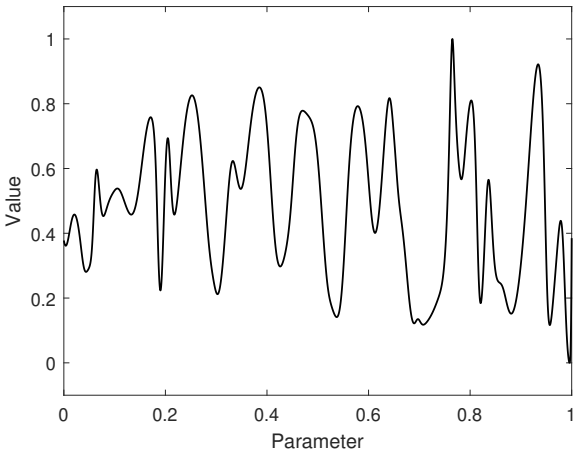


(c) Root mean squared error

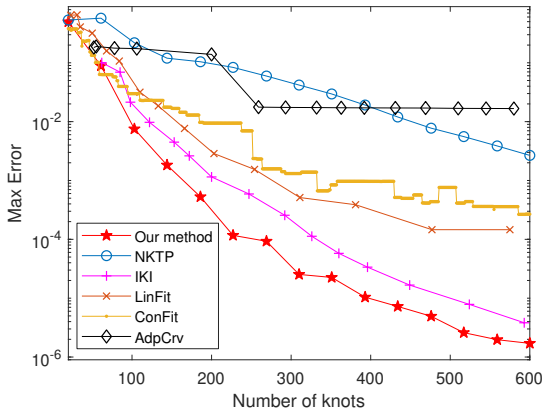
Fig. 3.15.: Methods comparison for 704 1D input points taken from a slice of 3D simulation data.

achieves one of the lowest approximation errors at higher knot count, as it captures the sharper turns. ConFit achieves a low error with fewer knots, but the decrease in error slows as more knots are added to locations with highly varying third derivatives but already low error. AdpCrv does not converge as quickly as the other methods due to the more complex nature of the dataset, since more evenly spread knots are needed to prevent an excessive concentration of knots at high curvature locations.

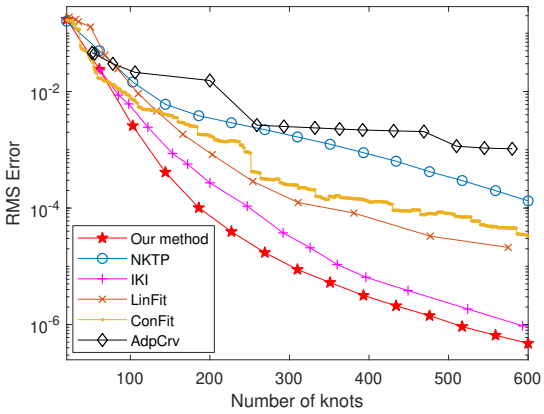
The NKTP method lowers the error at a steady rate with more knots added because of the relatively even detail distribution of the dataset across the parameter space.



(a) Input 1D data



(b) Max error



(c) Root mean squared error

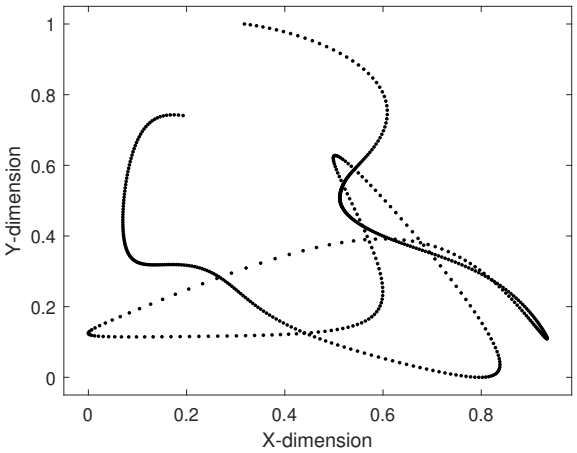
Fig. 3.16.: Methods comparison for a 1D randomly generated NURBS with 4000 points.

The fourth dataset shown in Figure 3.16a is a large randomly generated NURBS dataset with 4000 points and 80 random knots, control points, and weights. The randomized nature and the large number of complex regions represent complicated and unpredictable datasets found in real-world applications. Figures 3.16b and 3.16c show the approximation error for all the methods. Overall, our method achieves the

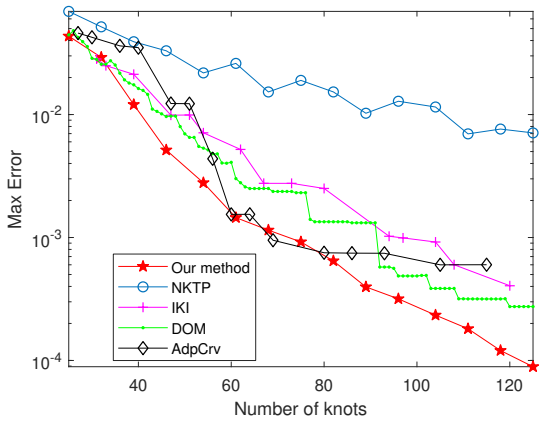
lowest approximation error across all number of knots, followed by the IKI method. The AdpCrv and NKTP methods result in the highest RMS and maximum error.

Parametric datasets

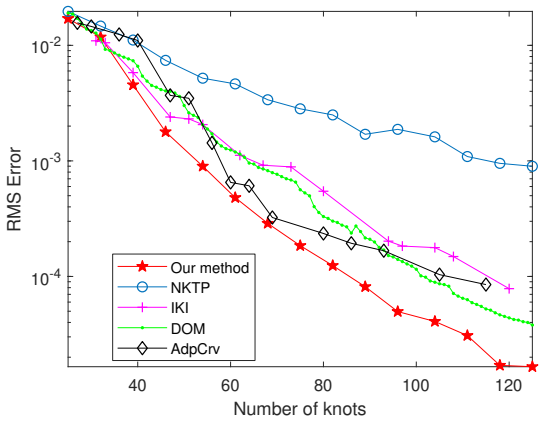
The datasets shown in Figures 3.17-3.20 are curves in 2 dimensions, parameterized by arc length. Methods LinFit and ConFit are described for only 1D datasets, and thus are not used to evaluate these parametric curves.



(a) Input parametric data



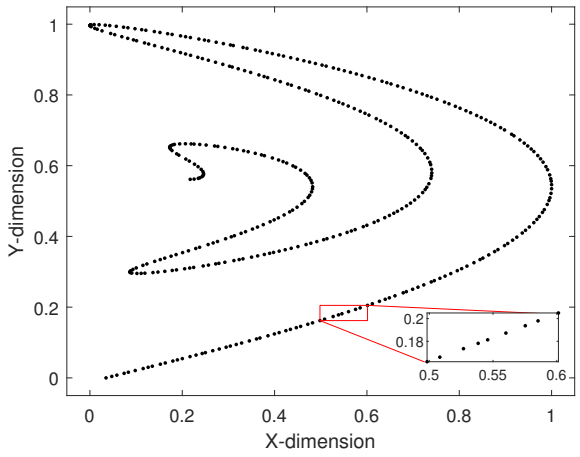
(b) Max error



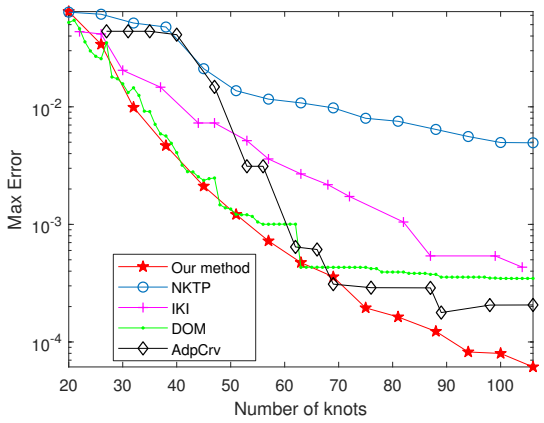
(c) Root mean squared error

Fig. 3.17.: Methods comparison for 601 parametric points sampled from a random NURBS curve.

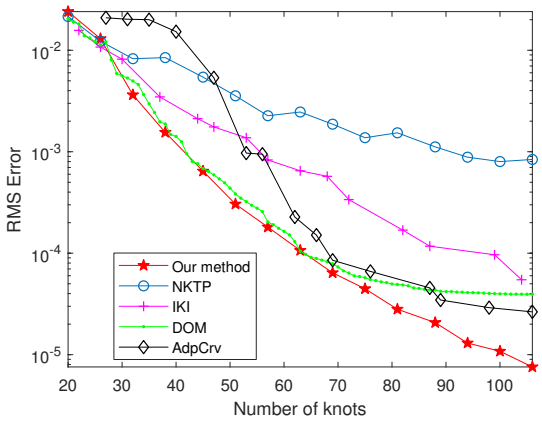
The dataset shown in Figure 3.17a is an order-7 NURBS curve generated using random control points and weights, and 32 random knots. 601 points are uniformly sampled in the parameter space. Figures 3.17b and 3.17c show the approximation error for this dataset across all the methods. Our method achieves the lowest error for both maximum and RMS errors, followed by DOM and AdpCrv methods.



(a) Input parametric data



(b) Max error



(c) Root mean squared error

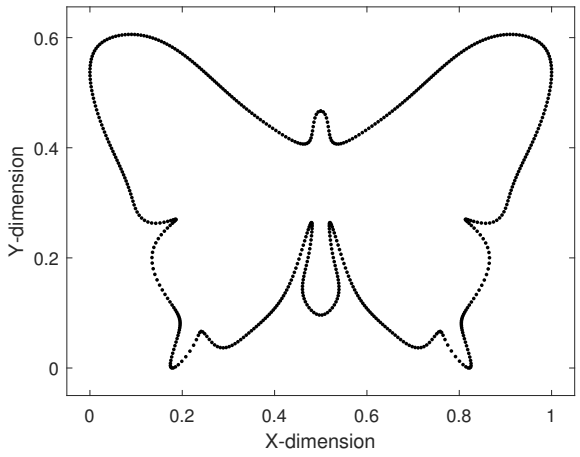
Fig. 3.18.: Methods comparison for 401 points nonuniformly sampled from a parametric function.

The dataset shown in Figure 3.18a is sampled from the parametric equations $x(u) = u(\cos(2u) + 0.5)$ and $y(u) = u\sin(u)$. 401 points are sampled along the arc

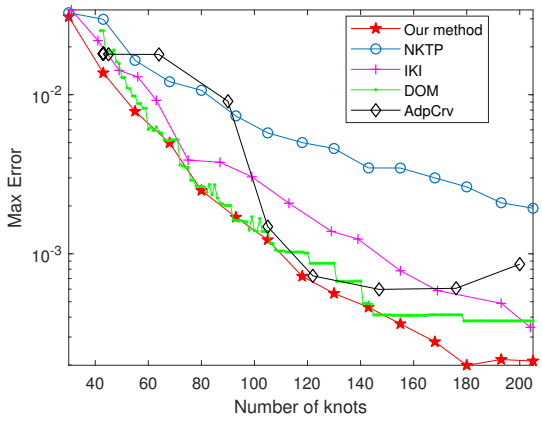
length, with higher sampling density where curvature is higher. The sampling spacing contains a small amount of randomized variation, as shown in the zoomed-in plot. Figures 3.18b and 3.18c show the approximation error for this dataset for all the methods. Our method achieves the lowest RMS error, but has higher maximum error in a few cases compared to the DOM method. Method AdpCrv has high error for a low number of knots, but achieves comparatively lower error with 50 or more knots. Even though higher sampling density at high curvature benefits the NKTP method, it produces the highest error in most cases.

The data of Figure 3.19a is a butterfly contour taken from [36]. 600 points are uniformly sampled in parameter space. It is a more challenging dataset than the cases considered so far, with sharper curves and corners. Our method achieves the lowest error for all numbers of knots, followed by the DOM and AdpCrv methods. The heuristic approach of the DOM method hones in on the sharper corners for knot placement, effectively reducing the approximation error. The IKI method has higher error with the same number of knots because the insertion method always splits at the middle of the segment, which is less effective for this dataset due to the localized high curvature. The NKTP method achieves the highest approximation error among all the methods.

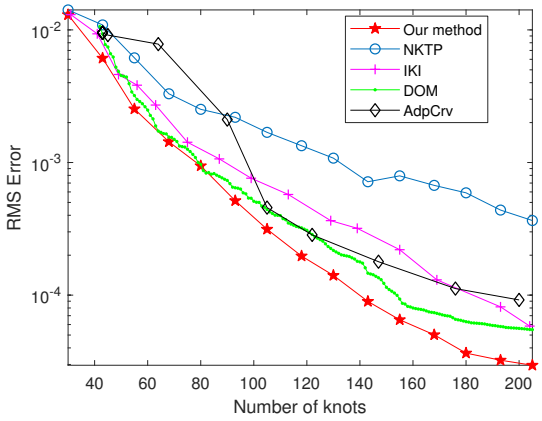
Figure 3.20a shows the last dataset, a parametric curve with 4000 points sampled from a NURBS curve generated using 70 random knots, control points, and weights. This dataset is larger and more complex than the other parametric datasets. Method DOM achieves lower maximum error for low number of knots, but for higher number of knots, our method reaches lower maximum error. For RMS error, our method maintains the lowest error compared with all other methods. The IKI method reduces both maximum and RMS errors consistently as more knots are added, but it does so at a lower rate than DOM and our method. Method AdpCrv does not reduce either error measure beyond around 300 knots. The NKTP method has the highest error compared with all other methods.



(a) Input parametric data



(b) Max error

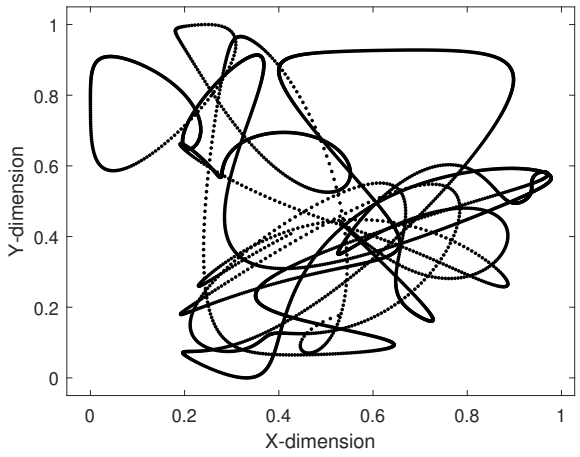


(c) Root mean squared error

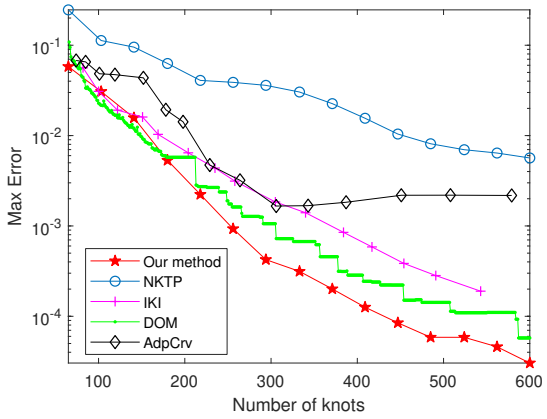
Fig. 3.19.: Methods comparison for 600 points sampled from a butterfly contour taken from [36].

Discussion of each method

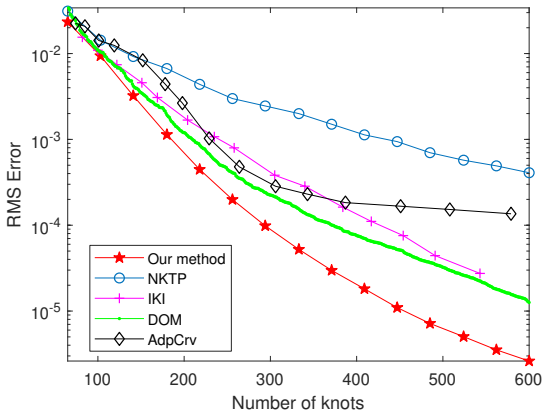
In general, our knot placement method allocates more knots to segments with higher information content, resulting in reduced approximation error. In most cases, our method achieves lower approximation error with the same number of knots compared with other methods.



(a) Input parametric data



(b) Max error



(c) Root mean squared error

Fig. 3.20.: Methods comparison for a randomly generated NURBS parametric curve with 4000 points.

The NKTP method has the highest error for most cases since it ignores the features of the input dataset. The IKI method, due to its adaptive refinement strategy, achieves comparatively lower error in the 1D cases. In particular in the simulation dataset, IKI yields results comparable to our method, focusing on the locations where more knots are needed to reduce error. Due to the chord-length parameterization for curves in 2D, data points may be grouped closely together in a small parameter do-

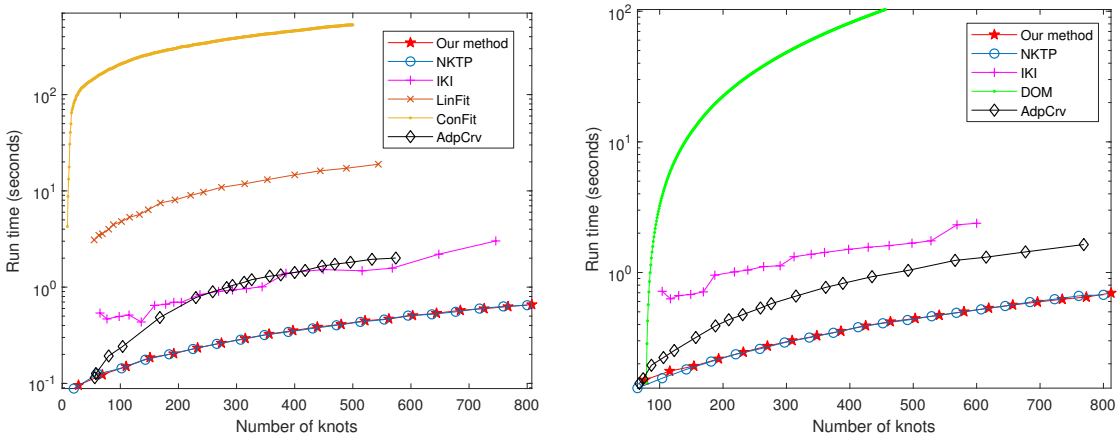
main, requiring more refinement steps and knots to improve the approximation in those regions.

Method LinFit is described only for 1D datasets, and is thus not used on the three parametric datasets. It allocates more knots at locations with highly varying second derivatives to capture the detail in the data, but in some cases could result in redundant knots at the locations with highly varying second derivatives. The method's accuracy directly depends on the fitting parameter used, although the relationship between fitting parameter and approximation error is unclear.

ConFit is described for 1D datasets only, and is not used on the three parametric datasets. This method may perform better in cases where fewer knots are used to capture the general shape of noisy data or data with few samples, as the cubic smoothing spline used for calculating derivatives filters out the noise in the data. In the present context, however, ConFit does not perform as well as other methods because our test datasets are free of noise, and we aim to approximate to a low error tolerance.

Method AdpCrv, being curvature based, achieves lower error on smooth parametric datasets. For datasets with higher curvature, the method can allocate too many knots in the high curvature regions. The simulation dataset shown in Figure 3.15a has many curvature peaks, making it a challenging dataset for AdpCrv method.

The DOM method is described only for parametric curves, and is not evaluated with the three 1D datasets. The method's error-driven knot placement reduces the maximum error quickly for low knot count. However, because existing dominant points are not updated, and there must be a minimum of three input data points between any dominant point pair, the maximum error can plateau as more knots are added.



(a) 1D data with 4000 points

(b) Parametric data with 4000 points

Fig. 3.21.: Timing vs. number of knots for 1D and parametric data.

3.4.2 Timing

We compare the timing of each methods, using a Desktop computer with a 3.3GHz Intel i7-3960X CPU and 32GB RAM. For each parameter used in each method, the timing result reported is the median of ten runs.

Our implementations of the various methods (including our own) are not optimized, thus the resulting timing information is meant to show relative performance for comparison only.

For the algorithms using 1D data, we use the randomly generated dataset shown in Figure 3.16a with 4000 points. The summary of the resulting timing for all methods on 1D data is shown in Figure 3.21a.

Algorithms operating on parametric curves are compared with the randomly generated dataset shown in Figure 3.20a, also with 4000 points. Figure 3.21b shows the timing results for the parametric data.

Since we start with the minimum number of knots for method IKI, it takes many iterations and performs as many least squares solves. We only count the time for iter-

ations that occur after the knot vector becomes non-uniform, for a fairer comparison, which amounts to starting the iterative process with a uniform knot vector.

The fastest run times per knot inserted is obtained with NKTP and our method, as no iterations are needed. These methods have $\mathcal{O}(n)$ runtime complexity where n is the number of input points.

The run times of globally iterative methods (such as DOM and IKI) and locally iterative methods (AdpCrv method and LinFit) are affected by their convergence, which is data dependent. The iterative IKI method solves a linear least squares system at each iteration, and thus requires more time to reach the desired result. The LinFit method fits a piecewise linear function to data points during its local iteration, which we implemented using linear programming. Using a different fitting criterion may improve the speed of the method. Method ConFit performs a neighboring knot adjustment step after each newly inserted knot, which can be time consuming. Method DOM inserts one knot and solves a linear system at each iteration, hence it is the slowest of all considered methods. We implemented the method as described in the paper, but the timing may be improved by adding all the knots for all segments above the error threshold in an iteration. At each iteration, the linear system changes by only a few columns, which could also be leveraged by using an iterative solver to reduce the execution time further. The run time of AdpCrv method depends heavily on the number of local adjustment iterations, which is, in turn, data dependent. The method is likely to perform faster with smoother data as fewer adjustment iterations are needed. This can be seen in the comparison of AdpCrv method between 1D and data parameterized by arc length, which result in lower curvature. This can explain the faster run time in the parametric case than in the 1D case.

3.5 Conclusion and Discussion

We introduced a novel knot placement optimization method that analyzes high-order derivatives of the input data, and generates a knot placement such that the

resulting least squares fit has low error. We demonstrated our method's effectiveness by comparing it to a number of state-of-the-art knot placement methods, and showed that our method can achieve comparable or higher approximation accuracy using fewer knots for a range of 1D and parametric datasets. Our method is also computationally inexpensive, with run time scaling linearly with the dataset size. In our experiments, our method's execution times were on par with the fastest methods we evaluated.

A challenge for our method is noise in the data, due to the need to calculate higher-order derivatives, which tend to amplify the noise. Methods for computing derivatives of noisy data, e. g., based on an intermediate reconstruction with a smoothing B-spline, will need to be assessed in this context to extend this method for data containing noise.

The regression analysis to determine the number of knots for a given error tolerance is a rough estimate for datasets with a certain convergence rate. Further study can be done on analyzing the smoothness of the dataset to determine its potential convergence rate given a selected B-spline order.

We expand our 1D approach to approximating multidimensional data in the next chapter, including the challenges for derivative computation of multidimensional data and the calculation of the feature function for each dimension.

4. FEATURE-GUIDED KNOT PLACEMENT FOR MULTIVARIATE B-SPLINE APPROXIMATION

In this chapter, we present an extension to the multidimensional setting of the 1D B-spline data fitting solution introduced in the previous chapter. Specifically, we seek a multidimensional knot lattice that enables accurate reconstruction results by the associated tensor product B-splines. Achieving a high accuracy multivariate B-spline approximation faces a similar knot selection problem as its 1D counterpart, but with a vastly larger search space. Existing knot placement methods are restrictive, inefficient, or produce far from optimal approximations. In this chapter, we expand our previous knot placement method for 1D datasets by proposing feature calculation and knot assignment for high dimensional datasets. Similar to our 1D approach, our method makes a derivative-based feature measure that characterizes the amount and spatial distribution of features in the input data. Knots are then selected in such a way as to evenly distribute the feature contents across the domain. We discuss the selection of a near-optimal number of knots in each dimension, a challenge that only arises in a high-dimensional setting. Our method inherits the simplicity and speed of its 1D counterpart and outperforms existing high-dimensional knot placement methods in terms of accuracy and number of knots used, which we demonstrate with a variety of 2D and 3D datasets.

4.1 Introduction

Interpolation and approximation problems for high dimensional data (surfaces, volumes, hypervolumes) are important in areas such as computer-aided geometric design, reverse engineering, modeling, data analysis, and data compression. Of the different kinds of multidimensional B-splines for data approximation (such as classical

B-splines, triangular B-splines, T-splines, box splines, simplex splines, etc.), tensor product B-splines have the advantage of an implicit geometric grid that has simple and efficient construction, minimal data structure storage, and fast interpolation.

Compared with the one-dimensional version of the problem, high dimensional data approximation with tensor product B-spline now requires the selection of the B-spline order and a knot vector for each dimension of the input data. With those determined, the control coefficients of the B-spline reconstruction is then solved in the least-squares sense. Similar to the one-dimensional approximation problem, the choice of knot vectors still has a large impact on the resulting approximation. While a higher density of knots yields a locally increasing approximation accuracy, deciding how to best distribute a set number of knots across each dimension is a challenging problem. Considering the non-linearity of the problem and the exponential increase in solution space with the increased dimensionality, an exhaustive search is intractable, and in the absence of a known optimal mathematical solution, finding an efficient heuristic becomes important. Existing knot placement methods for high-dimensional datasets rely on iterative refinement and various heuristic approaches to generate knot vectors for approximations. However, the results are either far from optimal, or their computational demand increases exponentially with the increase in dimension, which quickly becomes intractable. Some heuristics also impose restrictions on the input data format, dimension, or the order of the B-spline used.

In this work, we extend our heuristic 1D knot placement approach from the previous chapter to the high dimensional problem set, while maintaining efficiency and accuracy. We propose derivative estimation methods for structured and unstructured datasets in high dimensions, and calculate a similar *feature function* as the previous chapter but for each dimension separately. Each dimension's feature function measures the amount and distribution of directional details for that particular dimension. A high value in the feature function indicates that a higher knot density is needed in that dimension in order to capture the details and satisfy the prescribed approximation error tolerance. Our method, unlike other heuristic approaches, differ-

entiate anisotropic details and results in effective knot placement that increases the approximation accuracy while reducing the total control points used.

Our method works for input data of any dimension and for any B-spline order. It is fast and produces high accuracy approximation for a wide range of input data, provided they are smooth and sampled densely enough for accurate estimation of high-order derivatives. To the best of our knowledge, our method is the first direct knot placement method for data of dimensions higher than 1 (curves), i. e., 2D surfaces, 3D volumes, 4D hypervolumes, etc. that takes into account the features of the input data. Existing knot placement methods either do not take into account data features, or are iterative, and as a result, are more computationally expensive.

The remainder of this chapter starts with a presentation of related work in Section 4.2. We present our method in Section 4.3 and compare it with prior works in Section 4.4. Conclusions are drawn and avenues for future work are discussed in Section 4.5.

4.2 Related Work

Knot optimization problem for high dimensional B-spline data fitting is comparatively less explored than the corresponding 1D problem, but still a well-studied topic, with many approaches proposed in the literature.

Early work assumed the input data are given as a series of polylines or on a structured grid, which is a topologically rectangular grid. A direct extension of the NKTP method [33] from the previous chapter can then be easily extended to higher dimensions, where the knots are chosen as the average of consecutive parameters [6] or as the average of representative values or parameter point groups [33]. These methods only consider the point sampling density, and not the properties of the dataset, potentially resulting in inadequate knot distributions in regions with high variations in values.

Treating the parameters, knots, and control points as unknowns gives more flexibility to the problem, but makes it highly non-linear and more complicated. Xie *et al.* [37] proposed combining an optimization method and an iterative method for finding knots, weights, and parameterization of a non-uniform rational B-splines (NURBS) approximation to measured points in 3D space. Their method can be time consuming due to the large number of unknowns resulting in a large solution space.

Others use artificial intelligence to tackle the challenge. Yoshimoto *et al.* [19] introduced the first genetic algorithm that optimizes for the numbers and locations of the knots for curve and surface approximation. Later, more sophisticated multi-objective genetic algorithms were developed for improved approximation results [38, 39]. In those works, the parameters of the genetic algorithms are chosen empirically and are very problem-dependent. The resulting approximation often has redundant knots, and the procedure can be computationally expensive.

Alternative to search-based approaches, heuristic approaches use features of the input dataset to guide the placement of knots. Our proposed method in the previous chapter (Yeh *et al.* [40]) uses high-order derivatives of the input data as a measure of data features and is limited to 1D and parametric curves. For 2D data in structured grid format, Park [41] proposed an approach that uses the curvature to select dominant columns in the dataset to generate knots for B-spline surface approximation. Their method iteratively adds knots in each direction until an error threshold is met. Similarly, but not limited to a structured grid format, Zhang *et al.* [42] combined relocation and knot insertion in B-spline surface fitting. They proposed using the magnitude of the principal curvatures of the mesh as geometric features, and optimizing the knot vectors to divide the cumulative geometric measure evenly across all knot spans. Ravari and Taghirad [43] used group testing to sequentially select salient points in the input curve or surface dataset, which were used to determine the knot vector for B-spline approximation.

Peterka *et al.* [5] proposed using tensor product B-splines to represent scientific data for smooth reconstruction and analytic methods requiring smooth derivatives.

They determined the knot vectors by iteratively adding knots to the knot spans with the highest error, reducing the global approximation error until a predefined threshold is reached. They demonstrated their method on scientific data of up to 4 dimensions, using various B-spline orders. Their method solves a linear system at each grid refinement iteration, and is limited to datasets in a structured grid format.

A series of efficient data fitting methods called progressive-iterative approximation first applied to B-spline approximation by Lin *et al.* [44], worked by iteratively adjusting the control points to fit the input curve or surface data without solving a linear system. Later, the method was expanded to allow for fewer control points than input points, and also implicitly optimized the knot vectors by adding new knots at the location of the highest error and adjusting the local parameterization in the process [45, 46].

4.3 Methodology

In Chapter 3, when approximating a 1D or parametric curve with a B-spline of order p, the p^{th} derivatives of the curve are used to calculate feature values that measure the complexity of the input curve. The feature values then guide the placement of knots needed to accurately capture the curve. The method presented in this section extends the B-spline curve fitting method in the previous chapter to beyond univariate curves. We extend the feature calculation to data with arbitrary dimensions by calculating the partial derivatives of the high-dimensional input data, and using these derivatives to create a feature function for each dimension to guide the knot placement for that dimension. Our work has three main steps. First, high-order partial derivatives are estimated for the input data. These derivatives are estimated on a structured grid. Second, based on the estimated derivatives, a separate feature function is calculated for each dimension of the parameter space. Third, knots are generated based on these feature functions, such that each knot span contains an

equal amount of integrated features. The resulting knots are then used to generate the least squares B-spline approximation.

4.3.1 Derivative Calculation

We first describe the derivative calculation scheme for input datasets that are given in a tensor product grid format. Then we propose a method for derivative estimation for unstructured grids and scattered datasets. Note that if a dataset has known derivatives, those derivatives can be used directly in the next steps of the knot placement algorithm.

Structured Grids

Without loss of generality, assume the input dataset corresponds to a 2-dimensional grid, that is, $D = 2$, with a given set of $m_1 \times m_2$ input points with parameters Ω

$$\Omega = \{u_i\} \times \{v_j\}, \quad i = 1, \ldots, m_1,\ j = 1, \ldots, m_2 \tag{4.1}$$

with $u_i, v_j \in \mathbb{R}$, and each point corresponds to a value in Q

$$Q = \left\{\boldsymbol{q}_{i,j} : \boldsymbol{q}_{i,j} \in \mathbb{R}^G\right\}, \quad i = 1, \ldots, m_1,\ j = 1, \ldots, m_2 \tag{4.2}$$

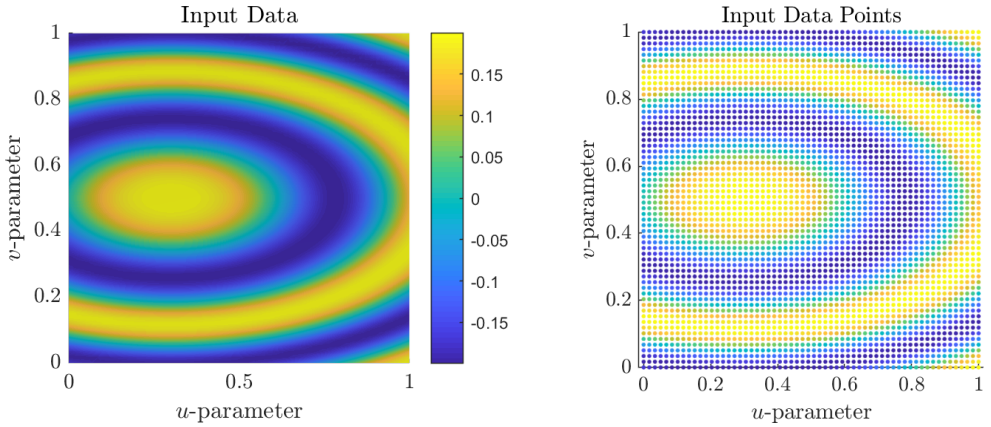
For the purpose of approximating an input dataset with a B-spline of order p, we want to estimate $\partial^p Q/\partial u^p$ and $\partial^p Q/\partial v^p$, the p^{th} partial derivatives in the u- and v-directions, respectively. Let

$$Q_{u^p} = \left\{\boldsymbol{q}^{(p)}_{i,j} \in \mathbb{R}^G\right\}, \quad i = w+1, \ldots, m_1 - w,\ j = 1, \ldots, m_2 \tag{4.3}$$

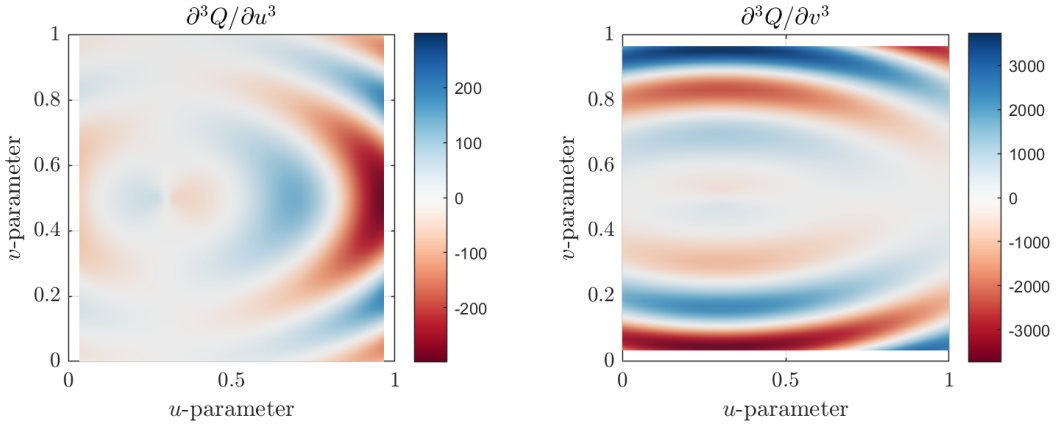
be the set that approximates $\partial^p Q/\partial u^p$ at parameters

$$\Omega_{u^p} = \{u_i\} \times \{v_j\}, \quad i = w+1, \ldots, m_1 - w,\ j = 1, \ldots, m_2 \tag{4.4}$$

where $w = \lfloor (p+1)/2 \rfloor$. Note that the u-direction domain is shrunk by $2w$ points when calculating the derivatives. We find that shrinking the domain suited the later



(a) An example dataset sampled on a 60×60 uniform grid.



(b) The third derivatives in the u- (left) and v-direction (right).

Fig. 4.1.: An example of derivative estimation for a 2D dataset in a structured grid format.

knot placement method (see Section 4.3.2) better than using forward/backward finite difference method at the boundaries.

Each point $\boldsymbol{q}_{i,j}^{(p)}$ is calculated using a finite difference stencil [47]. We use central finite difference method for arbitrarily spaced points with second-degree accuracy.

The finite difference stencil contains $N = 2w+1$ points. For the point at parameter (u_i, v_j), $w+1 \leq i \leq m_1 - w, 1 \leq j \leq m_2$, the stencil points $\{s_k = u_{i-w+k-1} - u_i\}_{k=1}^{N}$ are the parameter differences from u_i to the $2w$ surrounding input points in the u-

direction. The central finite difference coefficients c_k can be solved with the $N \times N$ Vandermonde system

$$\begin{bmatrix} 1 & \cdots & 1 \\ s_1 & \cdots & s_N \\ s_1^2 & \cdots & s_N^2 \\ \vdots & \ddots & \vdots \\ s_1^{N-1} & \cdots & s_N^{N-1} \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} = p! \begin{bmatrix} \delta_{0,p} \\ \vdots \\ \delta_{i,p} \\ \vdots \\ \delta_{N-1,p} \end{bmatrix} \tag{4.5}$$

where the $\delta_{a,b}$ are the Kronecker delta function. Uniformly spaced dataset will have the same set of coefficients for any (u_i, v_j). The derivative values then can be found with

$$Q_{u^p} = \left\{ \boldsymbol{q}_{i,j}^{(p)} = \sum_{k=1}^{N} c_k \boldsymbol{q}_{i-w+k-1,j} \right\}, \quad i = w+1, \ldots, m_1 - w,\ j = 1, \ldots, m_2 \tag{4.6}$$

A similar estimation is repeated for Q_{v^p}. For higher dimensions ($D > 2$), the same procedure applies.

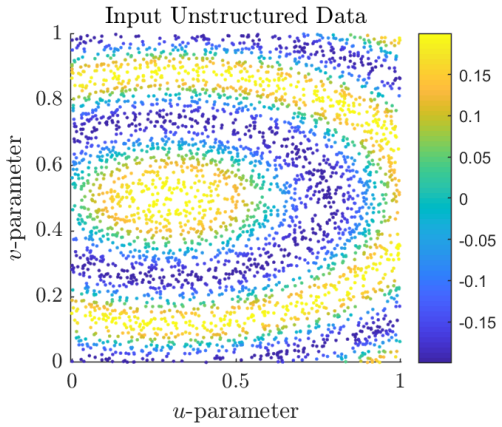
An example derivative calculation result is shown in Figure 4.1. The input data (Figure 4.1a) is sampled on a uniform 60×60 grid, and the third derivatives calculated using the finite difference method are shown in Figure 4.1b.

This approach estimates derivatives for data in a structured grid format. Unstructured data, however, require a different method to estimate derivatives.

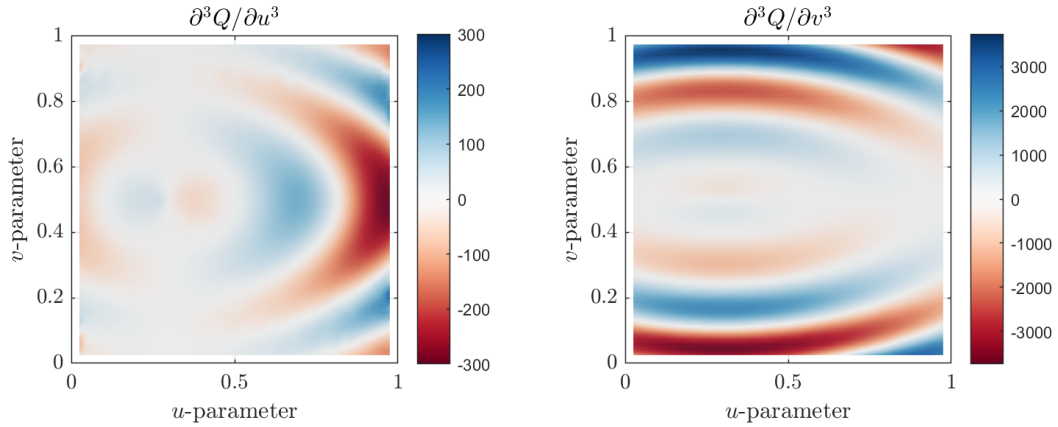
Unstructured Grids

For unstructured datasets, the derivatives are estimated on a regular grid in the parametric domain by using a weighted moving least squares polynomial fit [48]. In our derivative estimation method, a smooth function is locally fitted at every input point; derivatives are evaluated on each fitted function, and results from multiple fitted functions are blended to arrive at the final derivative estimates.

Below, we again assume 2D with no loss of generality. Let $Q = \{\boldsymbol{q}_i\}_{i=1}^m$ at parameters $\Omega = \{\boldsymbol{u}_i = (u_i, v_i)\}_{i=1}^m$ be the input unstructured points, where m is the



(a) An example unstructured dataset with 3600 points randomly sampled in the parameter domain.



(b) The third derivatives in the u- (left) and v-direction (right) on a 60×60 regular grid.

Fig. 4.2.: An example of derivative estimation for a 2D unstructured dataset.

total number of unstructured points. Let Ψ be a smooth function constructed with polynomials of order $\hat{p}$ derived from the Taylor series expansion about $\boldsymbol{u}_i$

$$\Psi(u,v) = \sum_{d=0}^{\hat{p}-1} \sum_{j,k=0}^{j+k=d} \mathbf{c}_{jk} \frac{(u-u_i)^j (v-v_i)^k}{j!k!} \tag{4.7}$$

where $\mathbf{c}_{jk} \in \mathbb{R}^G$ are polynomial fitting coefficients. We picked $\hat{p}$ to be $p+2$ in our work, where p is the approximating B-spline order. Let B_i be the set of M nearest

neighboring points from $\boldsymbol{u}_i$ in parametric space. The function Ψ is fitted to minimize the weighted 2-norm difference between Ψ and the neighbors of $\boldsymbol{q}_i$

$$\min_{\mathbf{c}_{jk}} \sum_{l \in B_i} w_{il} \left\| \Psi(u_l, v_l) - \boldsymbol{q}_l \right\|_2 \tag{4.8}$$

where $w_{il} = 1/\sqrt{\|\boldsymbol{u}_i - \boldsymbol{u}_l\|^2 + \epsilon}$ is the weight per point such that farther points are given less weight in the fit. We set ϵ to be 0.01 times the average distance from the i^{th} point to its neighboring points to avoid the numerical issue of division by a number close to zero.

The coefficients $\mathbf{c}_{jk}$ are solved with a $M \times N$ generalized Vandermonde matrix, where M is the size of B_i, or the number of neighboring points (including the point i), and $N = \hat{p}(\hat{p}+1)/2$ is the number of coefficients. We set M to be $3N$, such that there are 3 times more constraints than unknowns. We found empirically that this choice achieves a good balance between fitting accuracy and performance for our datasets.

Let Ψ_i be the fitted smooth surface centered at $\boldsymbol{u}_i$. The value at a query parameter $\boldsymbol{u}_q$ that is near $\boldsymbol{u}_i$ can be estimated by evaluating $\Psi_i(\boldsymbol{u}_q)$.

To estimate the p^{th} partial derivatives $\boldsymbol{q}_q^{(p)}$ with respect to u at a query location (u_q, v_q), we find the input points closest to the query point, denoted as B_q, and combine the derivatives of the smooth surface evaluated at the query location in relation to the origin of each fitting

$$\boldsymbol{q}_q^{(p)} = \frac{1}{\alpha_q} \sum_{i \in B_q} w_{qi} \frac{\partial^p \Psi_i}{\partial u^p} \left[(u_q, v_q) \right] \qquad \alpha_q = \sum_{i \in B_q} w_{qi} \tag{4.9}$$

using the same weight calculation as previously used. Similarly procedures follow for partial derivatives in the v direction.

The p^{th} partial derivatives of the input dataset are evaluated on a regular grid. We set the grid resolution to be $\lceil \sqrt{m} \rceil \times \lceil \sqrt{m} \rceil$, where m is the number of vertices of the input unstructured data. While this method of estimation works for unstructured datasets with quasi-uniform point sampling, for dataset with a high sampling variation, it may not be suitable due to aliasing problems. This problem can be alleviated by using a finer resolution grid or a grid with local refinement for portions of the input dataset with high point density or high value variations.

We choose neighboring points as points with the least distance in the parametric space. Certain input dataset can benefit from other choice of neighbors, such as using the connectivity information of the unstructured grid to determine the neighbor points [49].

Figure 4.2a shows an unstructured grid sampled from the same function as in Figure 4.1a with uniformly random samples. Figure 4.2b shows the derivatives estimated using the locally smooth fitting method described above. There are visible artifacts in the derivatives that are not found in the finite difference method (Figure 4.1b). The regular grid on which we calculate the derivatives does not cover the entire parameter domain, instead it is shrunk slightly to avoid fluctuations that often occurs on the parameter borders due to the lack of fitting constraints at the border. The amount of shrinkage is calculated with $p(u_{\max} - u_{\min})/(2\sqrt{m})$, which is similar to the border shrinkage of a structured grid with comparable number of points.

While this method also works for calculating derivatives in a structured layout, it is slower than the finite difference method and is not as accurate or stable due to the arbitrary sampling of the input data.

4.3.2 Feature Calculation and Knot Placement

Using the derivative estimation from the previous subsection, we will first compute a feature function for each dimension, then treat each dimension as a 1D problem and use the knot placement method as described in Section 3.3.

Without loss of generality, we present our feature function for the 2D case. The feature calculation applies to higher dimensions as well. We take the p^{th} derivative Q_{u^p}, formatted in a grid, at parameter locations Ω_{u^p}. For demonstration, we use the parameter indices resulting from the finite difference derivative calculation for structured grid, but the unstructured grid derivative parameter can be used in the same manner without loss of generality. To calculate the feature curve for the first parameter dimension (the u-direction), we take the maximum of the components of

each $\boldsymbol{q}_{i,j}^{(p)}$ (if $G > 1$) with the L-∞ norm, then the maximum of that result is taken across the other dimensions (in the 2D case, just the v-direction) to produce the max-derivative set F_u

$$F_u = \left\{ f_i = \max_j \left\| \boldsymbol{q}_{i,j}^{(p)} \right\|_\infty \right\}_{i=1,\ldots,m_1-p} \tag{4.10}$$

with the parameter

$$\widetilde{\Omega}_u = \left\{ u_i^{(p)} \right\}_{i=1,\ldots,m_1-p} \tag{4.11}$$

which are the u-parameters of the 2D derivatives. An example of this reduction from a 2D derivative surface to a 1D max-derivative curve is shown in Figure 4.3, where the maximum value is taken in the v direction to produce a 1D piecewise linear function across the u-parameter.

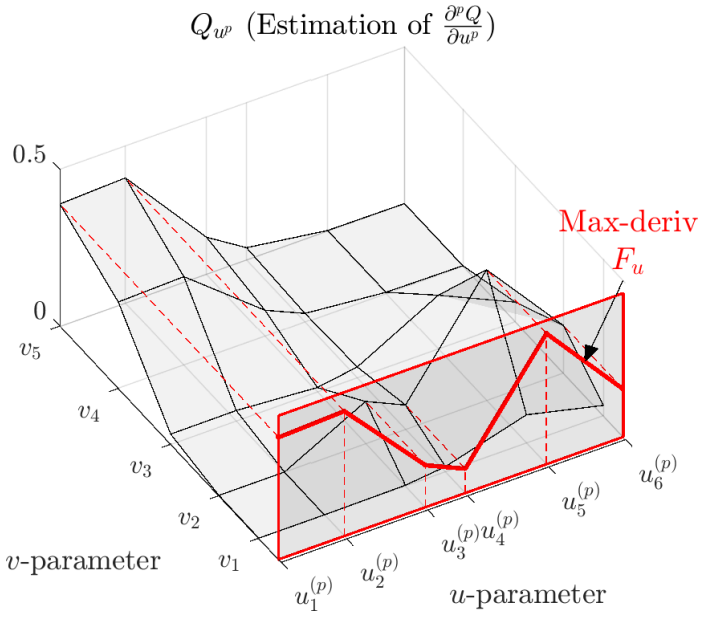


Fig. 4.3.: Demonstration of reducing a 2D tensor product of derivatives to a max-derivative curve. The gray surface is the derivative surface (with 1 component, where the value is shown by the height in the z-axis), and the red line is the F_u curve, by taking the maximum of the derivatives along the v-parameter.

With a derivative estimation calculated for each dimension, the rest of the knot placement method for each dimension follows Section 3.3, which we summarize below. A scaling function is applied to F_u by taking the p^{th} root of f_i,

$$\widehat{F}_u = \left\{ \widehat{f}_i = \sqrt[p]{f_i} \right\}_{i=1,\ldots,m_1-p} \tag{4.12}$$

This scaling is found to place the knots such that approximation error is evenly distributed, which results in the highest error in each knot segment to reduce at the same rate when more knots are used. This is explored in more detail in Section 4.3.4.

Next, the trapezoidal Riemann sum is calculated for each point. Let $\phi_u(u)$, $u \in [u_1, u_{m_1}]$, be the u-dimension *feature function*, defined to be the piecewise function that linearly interpolates the points $(u_1, 0)$, $(u_{m_1}, 0)$, and $(u_i^{(k)}, \widehat{f}_i)$ with $i = 1, \ldots, m_1 - p$. Then the *cumulative feature function* is defined as

$$\Phi_u(s) = \int_{u_1}^{s} \phi_u(u)\, \mathrm{d}u. \tag{4.13}$$

We want to place the knots such that each knot span contains the same amount of features. With knot clamping, the first and last knots are repeated p times. The knot vector $\boldsymbol{\xi}$ containing r number of unique knots is defined as

$$\boldsymbol{\xi} = \{\underbrace{\xi_1, \ldots, \xi_1}_{p}, \xi_2, \ldots, \xi_{r-1}, \underbrace{\xi_r, \ldots, \xi_r}_{p}\} \tag{4.14}$$

where the range of the knots is the same as the range of the input data parameter, that is, $\xi_1 = u_1$ and $\xi_r = u_{m_1}$. The locations of the r unique knots $\{\xi_1, ..., \xi_r\}$ are set such that each knot span contains the same amount of integrated features, that is, each ξ_i is chosen such that

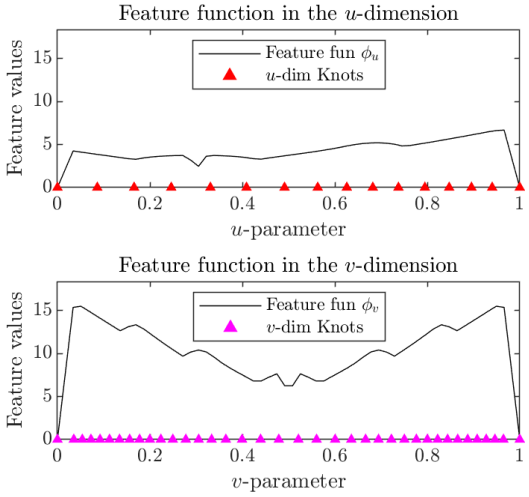
$$\Phi_u(\xi_{i+1}) - \Phi_u(\xi_i) = \frac{\Phi_u^{\max}}{r-1} \tag{4.15}$$

where $\Phi_u^{\max} = \Phi(u_{m_1})$ is the integrated value of $\phi_u(u)$, the u-direction feature function. The value $\Phi_u^{\max}$ can be considered as the total measure of u-direction feature, and this amount is equally distributed across all knot spans.

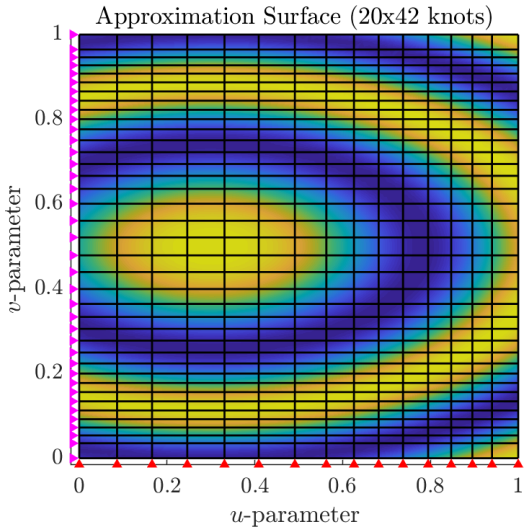
This step gives us the knot vector for the u-dimension. The same steps are repeated to calculate Φ_v and the knot vector for the v-direction. This process is the

same for higher dimension approximation. The resulting knot vectors are then used to solve for the B-spline approximation in the least squares sense.

Figure 4.4a shows the feature function and the resulting knot placement for the u and v directions for the dataset shown in Figure 4.1a. Figure 4.4b shows the resulting approximation and the knot vectors.



(a) Plots of the feature functions and the resulting knot placements for the u and v directions using the derivatives in Figure 4.1b.



(b) The resulting approximation using the knots produced above.

Fig. 4.4.: Example of the feature function calculation and the resulting knot placement.

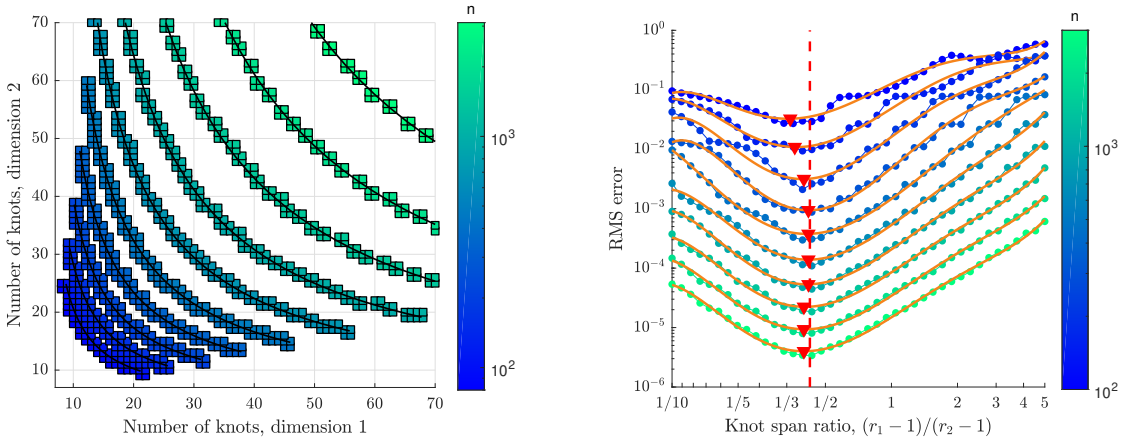
4.3.3 Determining Number of Knots

The knot selection method described in the previous subsection assumes that the number of knots in each dimension is known. In general, the choice of the number of knots is guided by the desired error tolerance: The lower the target error is, the more control points needed, resulting in more knots used. However, in the approximation

of high dimensional data, determining the number of knots for each dimension is not an intuitive process.

We first consider the problem of finding the number of knots in each dimension given a target total number of control points. Later we will consider the problem of determining the total number of control points given a target error.

Knots Per Dimension



(a) The combinations of number of knots used as parameters for our knot placement method. Color denotes number of control points (n). Black lines show isolines for number of control points.

(b) Plot of RMS error vs. knot ratio for various number of control points (n). Red triangles estimate the optimal knot ratio for each n value. Red dash line marks our selection for the optimal knot ratio ($\Phi_1^{\max}/\Phi_2^{\max}$).

Fig. 4.5.: Finding the optimal knot ratio for the dataset in Figure 4.1a.

Intuitively, a dimension with more complex detail than the other dimensions will need more knots for a good approximation. We use the integral of the feature function as a representation of the amount of total detail for a dimension, and use it to determine the number of knots to place. Specifically, the number of knots is chosen

such that the ratio of knot spans is the same as the ratio of integrated feature values between dimensions. A knot vector with r unique knots has $r-1$ knot spans.

Using $\Phi_d^{\max}$, the integrated feature function in dimension d, the desired knot span ratio between any two dimensions d_i and d_j is set to be

$$(r_{d_i}-1)/(r_{d_j}-1) = \Phi_{d_i}^{\max}/\Phi_{d_j}^{\max} \tag{4.16}$$

We will refer to this ratio of the number of knot spans across dimensions simply as *knot ratio* below for simplicity.

The relation between n, the total number of control points, and r_d, the number of unique knots in the d^{th} dimension, is

$$n = \prod_{d=1}^{D} (r_d + p - 1) \tag{4.17}$$

where p is the order of the B-spline, and D is the dimension of the dataset. Then given a target n, each r_d is solved such that Equations (4.17) and (4.16) are satisfied. For example, in the 3D case ($D = 3$), given a target total number of control points n, r_1, r_2, and r_3 are solved with the three equations below,

$$(r_1 + p - 1)(r_2 + p - 1)(r_3 + p - 1) = n \tag{4.18}$$

$$(r_1 - 1)/(r_2 - 1) = \Phi_1^{\max}/\Phi_2^{\max} \tag{4.19}$$

$$(r_1 - 1)/(r_3 - 1) = \Phi_1^{\max}/\Phi_3^{\max} \tag{4.20}$$

which can be solved numerically or in closed-form when possible.

Figure 4.5 shows how well this estimation performs for a higher resolution version of the 2D dataset in Figure 4.1a. First, a target number of control points n is chosen, and our knot placement is generated using a combination of r_1 and r_2 that results in n number of control points. The actual number of control point may not be exactly n; instead, we take the r_1 and r_2 that result in the closest n, find the resulting RMS error with those r_1 and r_2 values, then find the estimated RMS error for the selected n via linear interpolation. Figure 4.5a shows the combinations of r_1 and r_2 evaluated for 10 target n values. Ten black isolines shows the target n values (ranging from 50 to

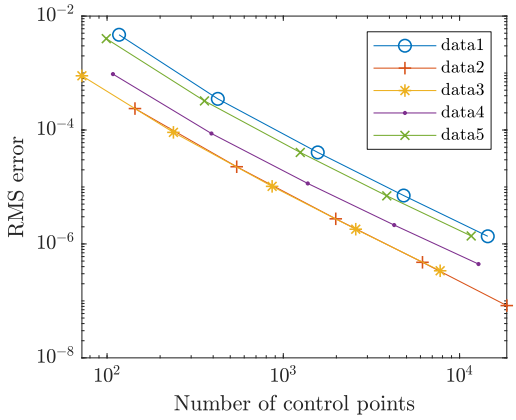
3000) with the axis indicating the number of knots in the first and second dimensions. In the figure, each colored point corresponds to an approximation solution generated using our knot placement method. Figure 4.5b shows the approximation RMS error for each sample on the isolines. (Again, note that samples on the isoline are linearly interpolated from 4 combinations of r_d.) To find the minimum across the n-isolines, a degree-5 polynomial is fitted over the points, shown with orange solid lines. The minimum of each fitted polynomial is indicated with a downward red triangle. These triangles mark the optimal knot ratio for a given n. Every point on the n-isolines has the same total number of control points, but due to the different knot ratio, result in different RMS error. Finding the optimal knot ratio gives the best approximation for a fixed number of control points. The vertical red dashed line shows the integrated feature ratio ($\Phi_1^{\max}/\Phi_2^{\max}$), which is our choice of knot ratio to use. As can be seen in Figure 4.5b, our selection is a reasonable estimate of the optimal ratio. The selected ratio is 0.43, whereas the optimal ratio from the search goes from 0.57 to 0.45 as n increases, converging to our selected ratio at higher number of control points.

This selection method for the number of knots allocates more knots to dimensions with more complex features. The number of knots in Figure 4.4 is picked with this method.

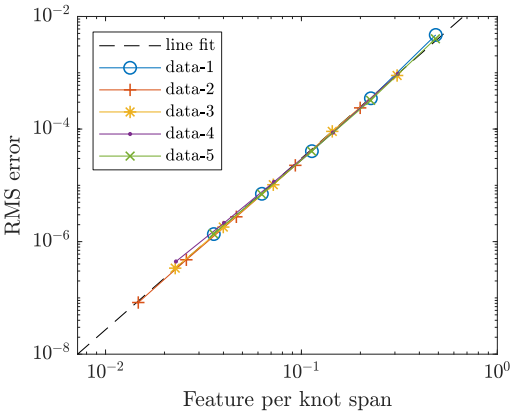
Total Number of Control Points

With a method to determine the number of knots for each dimension given a target total number of control points, we now consider the problem of determining the total number of control points given a certain target error.

Typically, the fewest number of control points to achieve a given target reconstruction error is desired. A more complex dataset would require more control points to reach the same error than a simpler dataset. Beyond its role to determine the number of knots in each dimension, the integral feature value $\Phi^{\max}$ can guide selecting the number of control points as well.



(a) Reconstruction error of the synthetic datasets across a range of numbers of control points.

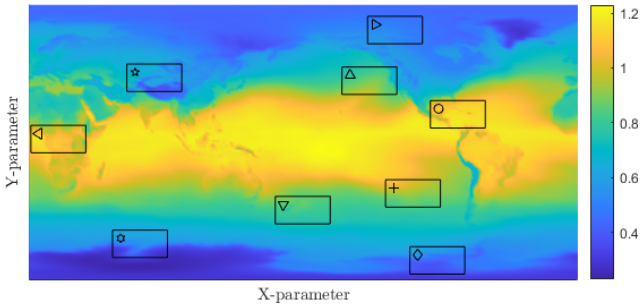


(b) The RMS error is a linear relation to the feature to number of knot spans ratio.

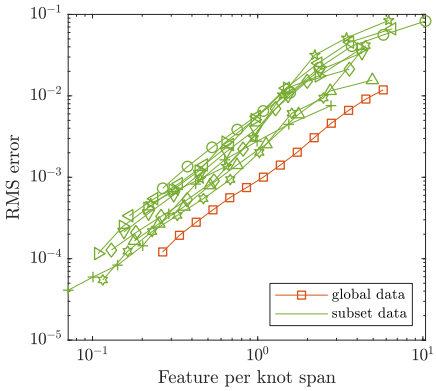
Fig. 4.6.: Linear regression for five synthetic datasets.

Figure 4.6 demonstrates the relation between $\Phi^{\max}$ and the reconstruction error. Five 2D synthetic datasets of different sizes and complexity, resulting in different $\Phi^{\max}$ values, were generated. We reconstructed these datasets with order-3 B-splines of varying number of control points using the method described above. The resulting RMS error of the approximation is shown in Figure 4.6a for the datasets. Figure 4.6b shows that the integrated feature value per knot span ($\Phi_d^{\max}/(r_d - 1)$) has a linear relation to the resulting RMS error for these five datasets. This suggests that given a user-specified error target and the integral feature value of the dataset, one can determine the number of control points needed through linear regression.

This regression, however, is data-dependent. The data shown in Figure 4.6 are smooth synthetic datasets with similar properties. Other datasets with different features, dimensions, and sampling will result in a different linear trend. For example, Figure 4.7a shows a 3,600×1,800 climate dataset. To estimate the trend of the dataset without repeatedly solving for the whole size of the dataset, we randomly selected nine sample blocks that are $1/10^{\text{th}}$ of the width and height of the dataset. Figure 4.7b



(a) Image dataset with nine random subsamples. The icons on the subsamples map to the icons in Figure 4.7b.



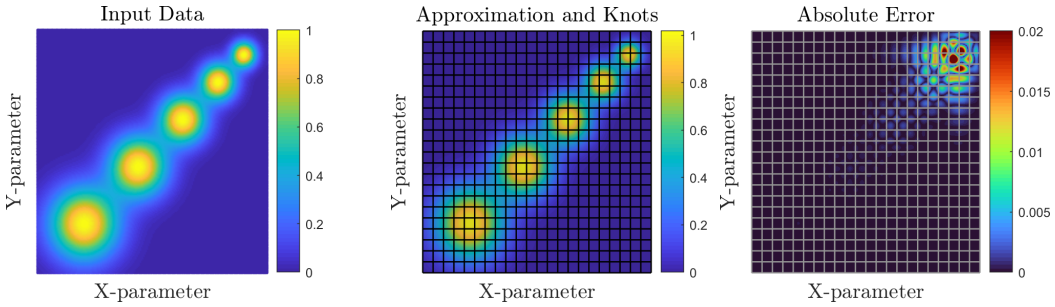
(b) The RMS error versus feature per knot span ratio.

Fig. 4.7.: Simulation dataset exhibit similar feature-knot trend for different subsamples.

shows the same feature-knot ratio vs. RMS error for both the complete dataset and the nine subsamples. The relation no longer lies on a single straight line. Instead, each block trend has a different offset, but share roughly the same slope. However, the climate dataset and its subsamples do share roughly the same slope. This suggests that we can estimate the slope of the regression quickly using a subsample of the input dataset, have an estimate of the y-intercept with one approximation of the full dataset, and thereby find a good approximation for the number of control points for a reconstruction that is close to the target error.

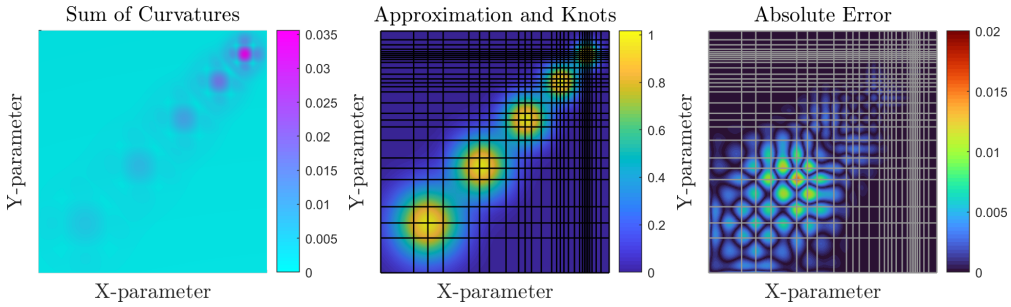
4.3.4 Rationale of Our Method

Two main aspects characterize our method: the design of the feature function, and separability of the feature functions over different dimensions of the data. The following two synthetic examples help to demonstrate the rationale behind these decisions.

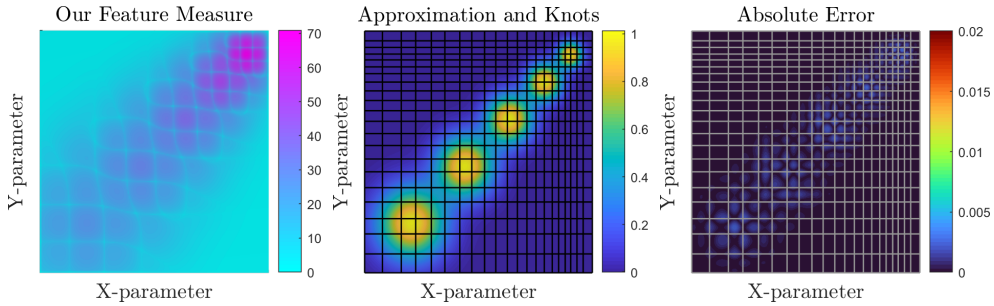


(a) Input Data

(b) Approximation and error using uniform knot placement.



(c) Curvature measure (left), approximation (middle) and error (right) using knot placement by evenly distribute curvature measures across knot spans.



(d) Feature measure (left), approximation (middle) and error (right) using our knot placement.

Fig. 4.8.: Comparison of approximation error distribution of uniform knot placement and our knot placement method.

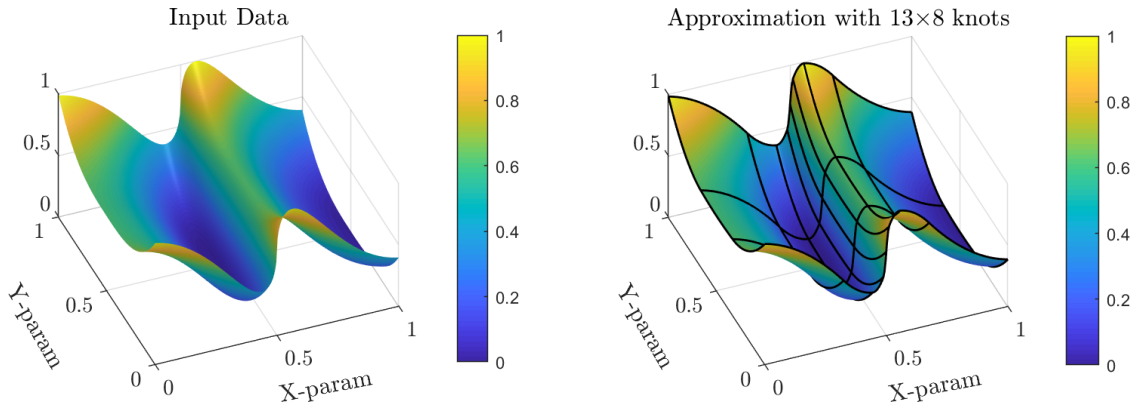
Figure 4.8a shows an input image dataset of five Gaussian functions along the diagonal with gradually changing width but with the same amplitude. Uniform knot

placement is first used to approximate the data. Figure 4.8b shows the bicubic B-spline approximation and the resulting error. Uniform knot placement does not take into account the features of the data, and as a result, the maximum with the smallest radius on the top right corner has the highest error.

A commonly used heuristic is curvature for guiding the placement of knots. Figure 4.8c shows the sum of absolute curvature measure along the X and Y-axis, and the approximation result from placing knots such that the maximum curvature measure is split equally between knot spans along a dimension. With curvature as guidance, the knot placement is more concentrated in the area of interest, but the knot density is too high, resulting in too many knots placed for the top right corner and too few in the lower left corner. Furthermore, prior methods using curvature as features to guide knot placement [41, 42] consider only cubic B-spline approximations. Using curvatures as feature is not suitable for reconstructions with B-splines of other orders.

In contrast, our knot placement method calculates a feature measure that is accurate in terms of quantifying the desired relative knot density for any given dataset for any order of B-splines. Figure 4.8d shows our feature measure of the same dataset in both dimensions, and the resulting B-spline approximation and error using the same number of knots. As can be seen, our feature evenly splits the error across knot spans, resulting in roughly constant error across each Gaussian.

We separate the derivative calculation into each dimensional direction so that fluctuations in one direction do not affect the knot placement in the other dimensions. Figure 4.9 illustrates this idea. Figure 4.9a shows input data that has high variations in the X-direction, but is relatively smooth in the Y-direction. As shown in Figure 4.9b, our method calculates a high feature measure in the middle of the X-parameter, and thereby places more knots at the location of the fluctuation. On the other hand, few knots are used in the Y-direction because there is little variation.



(a) Input data with concentrated X-direction variation, but little variation in the Y-direction.

(b) Order-3 B-spline approximation using our method. Black lines mark the knots.

Fig. 4.9.: High fluctuation in the X-direction has little effect on the Y-direction knot placement.

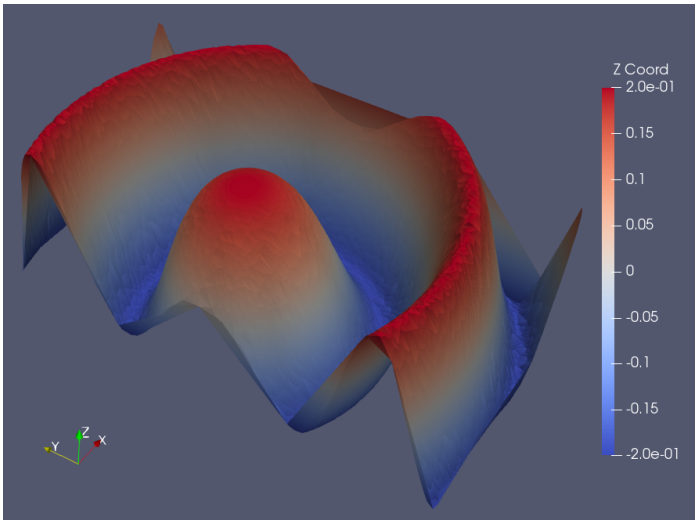
4.4 Experimental Results

The performance of our knot placement algorithm is compared against two prior state-of-the-art works: Zhang *et al.* [42], which works for triangular surface meshes, and Peterka *et al.* [5], which works for structured grid data in any number of dimensions. We use various synthetic, measured, and simulated datasets for our evaluations.

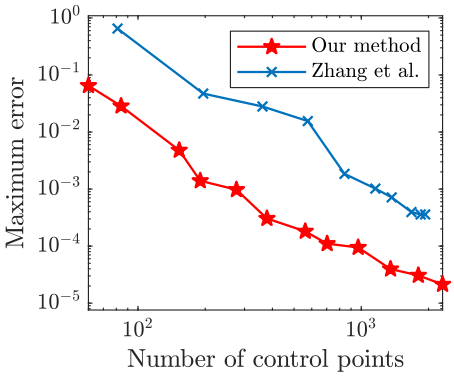
4.4.1 Triangular Surface Meshes

We implemented the method by Zhang *et al.* [42] in MATLAB. Their method uses the sum of the principal curvature of the mesh as a feature to guide the knot placement. In each iteration, at most 5 knots are added in knot spans with the highest approximation error until a target tolerance is reached.

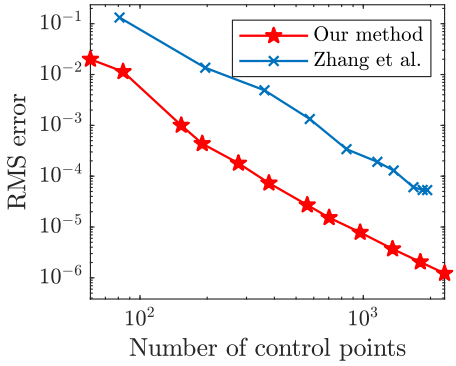
Note that the method by Peterka *et al.* is for structured grids only, and is thus omitted in comparison of triangular surface mesh reconstruction.



(a) Synthetic surface data with 10k points in an unstructured grid.



(b) Maximum error

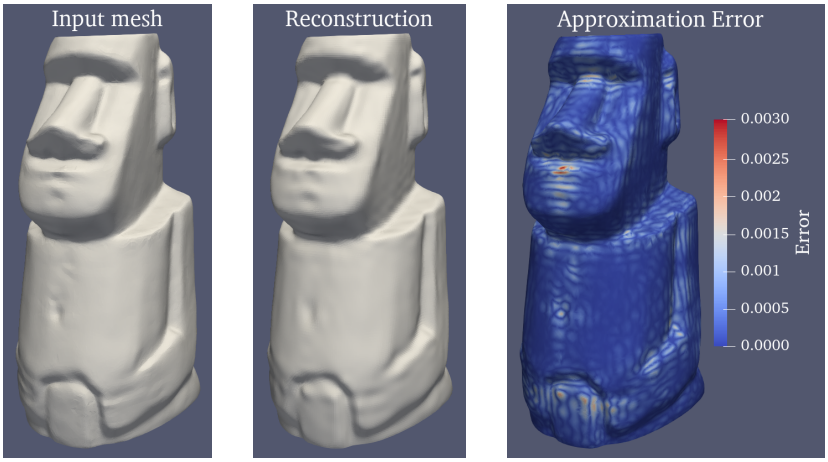


(c) Root mean squared error

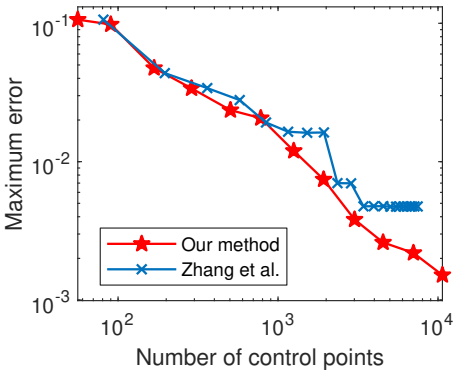
Fig. 4.10.: Comparing approximation error for a synthetic surface data.

Figure 4.10a shows the first dataset, which is a synthetically generated 2D wave with 10k points randomly distributed in the parameter domain. The dataset is sampled from the same function used to generate the data in Figures 4.1a and 4.2a. Figures 4.10b and 4.10c show the maximum and RMS errors, respectively, comparing approximation error of our method with the method by Zhang *et al.* with a varying number of control points. For the method by Zhang *et al.*, each point in the error plots corresponds to an iteration of the method until it terminates. For our method,

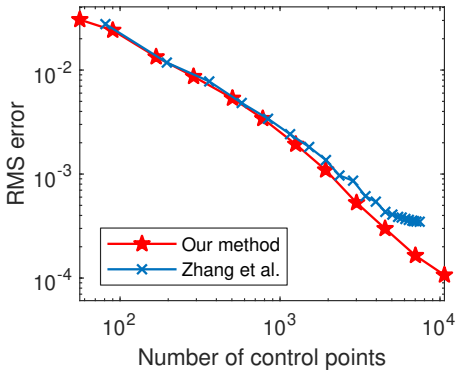
each point corresponds to an independent run. Our method was able to achieve a lower error for all numbers of control points we tested. This is due to the uneven nature of the data in the X and Y dimensions, with the X-dimension needing less control points compared with the Y-dimension. The feature used by Zhang *et al.* does not take into account the anisotropy.



(a) The original Moai dataset (left), the reconstruction by our method with 50×57 knots (middle), and the pointwise approximation error using our method (right).



(b) Maximum error



(c) Root mean squared error

Fig. 4.11.: Comparing approximation result for the Moai dataset.

The second dataset shown in Figure 4.11a is a surface mesh of a Moai model with 61k points and 120k triangles. Figures 4.11b and 4.11c show the approximation error

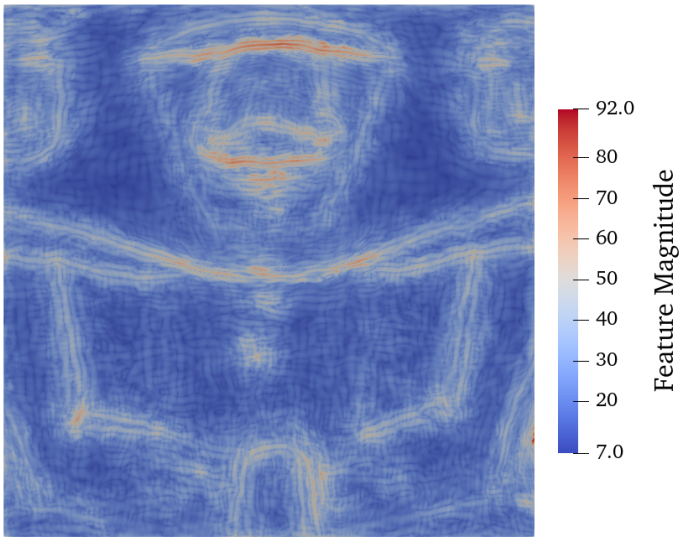


Fig. 4.12.: The Moai dataset in parameter space, colored by feature magnitude. Black lines marked 50x57 knot lines placed by our knot placement method.

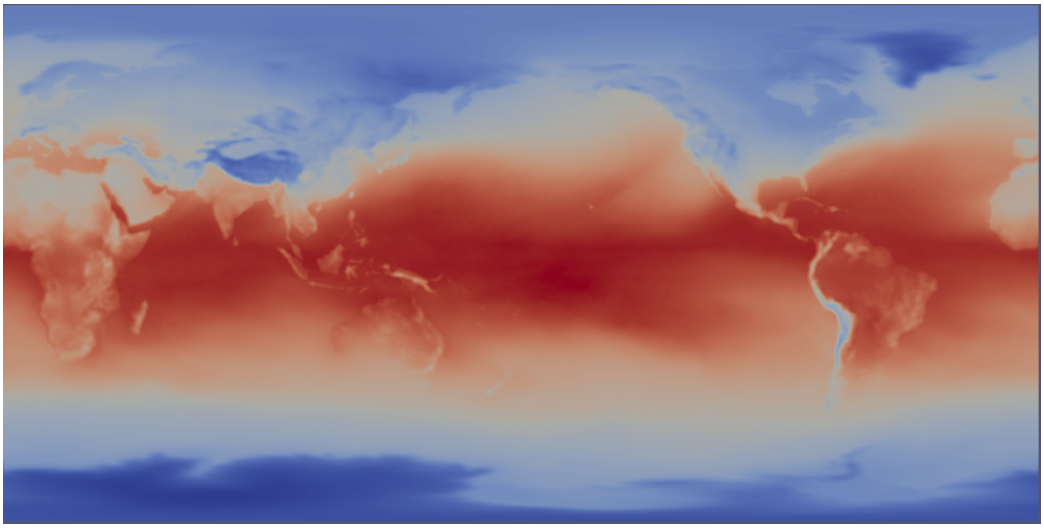
of our method and the method by Zhang *et al.*. The RMS error of both methods are similar, while the maximum error for the method by Zhang *et al.* plateaus in several places. Figure 4.12 shows the Moai model in parameter space, colored by the feature value calculated by our method. The black lines mark the 50×57 knot vectors generated using our method.

4.4.2 Multidimensional Simulation Data

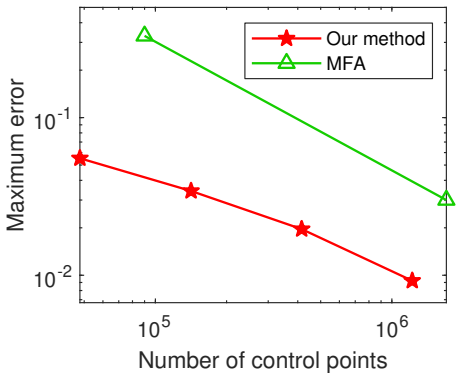
The second method we compare with is a multivariate functional approximation (MFA) method by Peterka *et al.* [5]. Their method is used for scientific data on structured grid format in arbitrary dimensions for any degree of B-spline.

Note that the method by Zhang *et al.* is for geometric surface meshes, and thus is not used in this subsection to compare against scientific data that does not have a meaningful curvature measure.

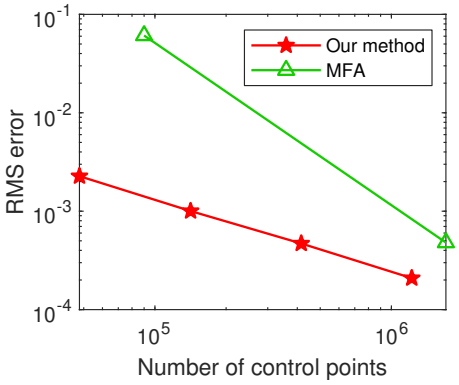
Figure 4.13a shows a 1,800×3,600 image of climate data modeling the FLDSC (Clearsky downwelling long wave flux at surface) variable of the Community Atmo-



(a) Input image data of size 3600×1800.



(b) Maximum error

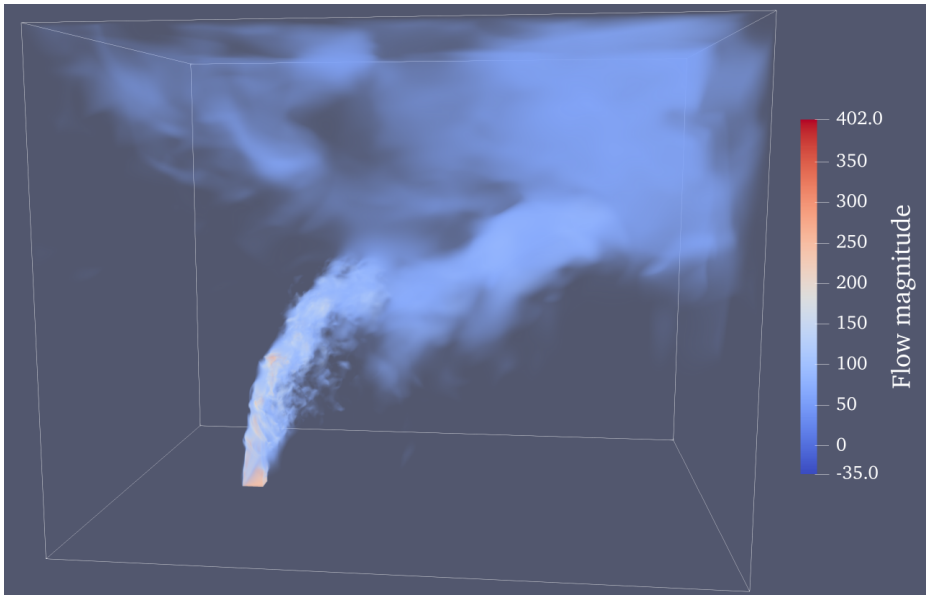


(c) Root mean squared error

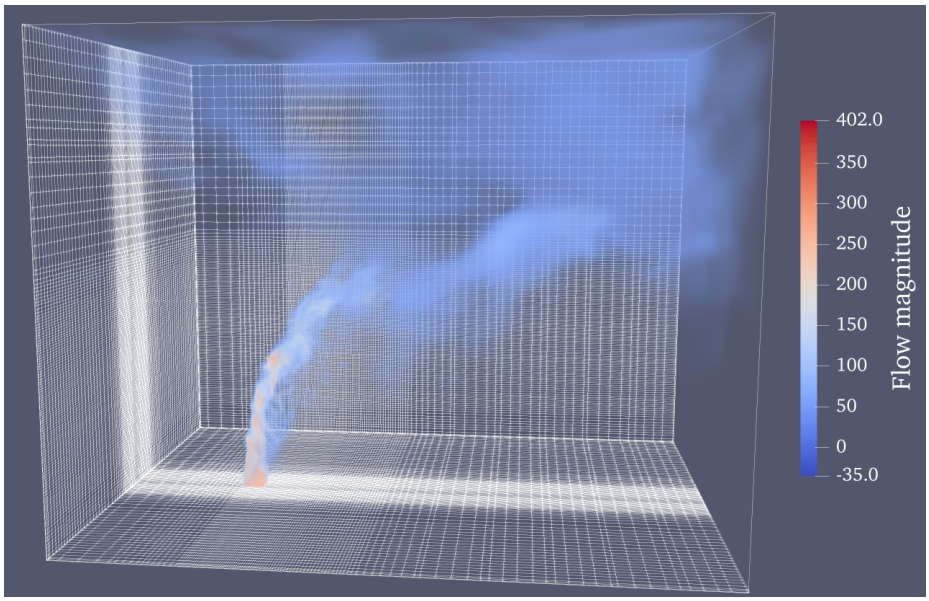
Fig. 4.13.: Compare approximation result of the climate data.

sphere Model (CAM) developed at the National Center for Atmospheric Research (NCAR) [50]. Figures 4.13b and 4.13c show the plots comparing the approximation error of our method with MFA. Our method achieves lower error compared with the MFA method. This could be due to the isotropic property of the knot insertion in MFA.

The last dataset is a structured 3D dataset generated by an S3D simulation of fuel jet combustion in the presence of an external cross flow [35]. The dataset is a

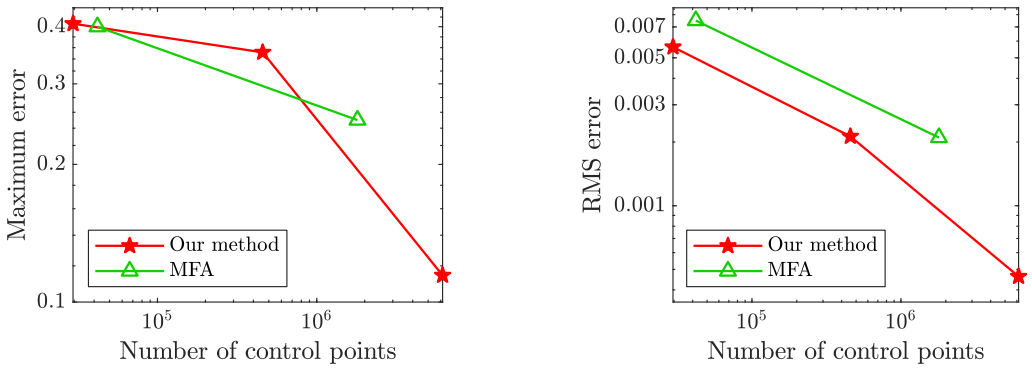


(a) Volume rendering of the input S3D dataset with $704 \times 540 \times 550$ resolution.



(b) Volume rendering of our reconstruction of the S3D dataset using $122 \times 129 \times 115$ knots. Knot placement is marked by the white lines.

Fig. 4.14.: S3D data



(a) Maximum error

(b) Root mean squared error

Fig. 4.15.: Approximation error for the S3D data

regular image of size $704 \times 540 \times 550$. Figure 4.14a shows a volume rendering of the input data. Figure 4.14b shows our high resolution reconstruction of the S3D dataset, with the knots used drawn as white lines on the border of the volume. Figures 4.15a and 4.15b show the maximum and RMS error comparison between MFA and our method. Both methods achieve similar maximum error, while our method reaches lower RMS error for similar number of control points.

4.4.3 Asymptotic Complexity

The three main parts of our method involve calculating the high-order derivatives, calculating the feature function in each dimension, and then finding the knot placements.

If the dataset is on a grid, the first step of derivative calculation using central finite difference takes $\mathcal{O}(m)$ time, where m is the size of the input dataset. In the case of unstructured input dataset, the run time for derivative calculation can be more complicated to estimate. In our method of local fitting for derivative calculation, we assume that we evaluate the derivatives in as many locations as there are input points, and each local surface fitting uses a constant number of nearest points. Thus the time complexity is $\mathcal{O}(cm)$ where c is the time it takes to locally fit the surface and

evaluate the derivatives. The second step, calculating the feature function, has $\mathcal{O}(m)$ time bound, where m is the size of the input data, or for the case of unstructured grid, the size of the derivative data. The third step of placing knots with the feature function involves linear interpolating the inverse cumulative function, and has run time linear in the size of each feature functions, typically at the scale of D^{th} root of the input data size.

In total, our method operates with $\mathcal{O}(m)$ run time, or linear complexity in the dataset size, assuming the number of knots has already been predetermined.

Both the method by Zhang *et al.* and MFA are iterative, where each iteration involves solving a linear least squares system and adding knots at the locations of the highest error until a certain threshold is reached. Our method places the knots with just one iteration, with comparable or lower error.

4.5 Conclusion and Discussion

We expanded our knot placement method from Chapter 3 on multivariate datasets using the high-order derivatives of the input data to generate a feature measure for effective knot placement such that the resulting least-squares fit has a low error. We showed the intuition behind the feature measure of our method and demonstrated our method's effectiveness by comparing it with state-of-the-art knot placement methods for various formats of high dimensional data. We showed that our method can achieve comparable or higher approximation accuracy using fewer knots for a range of 2D and higher dimensional datasets of structured and unstructured formats. Our method is also computationally inexpensive compared with existing methods, with run time scaling linearly with the size of the dataset, and for the case of unstructured data, the size of its derivative grid as well.

To the best of our knowledge, our method is the first direct knot placement method for high dimensional data that takes into account the anisotropic features of the input data.

A challenge for our method is the high-order derivative estimation for unstructured, noisy, undersampled, or non-smooth data. Datasets that have lower-order discontinuities, resulting in spikes in the numerical calculation of the derivatives, can cause undesirable results using our method. Similarly for noisy data, derivative calculation using finite differences will amplify the noise, resulting in unusable derivatives. While we used a smooth fitting of the unstructured dataset for the purpose of derivative calculation, doing so robustly remains a challenge for data with uneven sampling density. Alternative methods for more robust high-order derivative computation will be investigated in future work.

Datasets with high variation can result in a knot placement that is too dense in regions without input data points. This would result in a rank deficient linear system and likely undesirable fluctuations in the resulting least-squares reconstruction. For the 1D scenario, we described in Section 3.3 a method to prevent such a case from happening. For multidimensional datasets, the solution is less clear. One straightforward solution to this problem is to limit the knot placement density in each dimension such that resulting knot spans always have some input data points in it, which is employed by the method by Zhang *et al.* [42]. This is a workaround, but definitely not a desirable solution for datasets with varying point sampling that needs the nonuniformity of the knot placement to reach the desirable error threshold, because the point sampling (or the lack of) in one local region could limit the knot density for another region that has more detail and requires denser knots to capture. For unstructured input datasets, it becomes a more complicated problem that requires further research. We will tackle this challenge in the next chapter.

5. B-SPLINE APPROXIMATION FOR HIGHLY NONUNIFORM SCATTERED DATA

In the multidimensional B-spline data reconstruction method introduced in the previous chapter, feature functions are estimated for unstructured grids via a uniform estimation of the derivatives, and an approximation is found with a direct least squares fit. However, in cases where the point distribution of the input dataset is highly nonuniform, which is common for simulation datasets obtained by adaptive refinement, certain complications arise with this approach. This chapter improves upon the unstructured data approximation pipeline for cases of unstructured datasets with arbitrary point distribution. Specifically, a more efficient derivative evaluation scheme is proposed that utilizes a subdivision tree-based (quadtree in 2D, octree in 3D, etc.) evaluation resolution to generate the feature function needed for knot placement that is more efficient than the uniform grid suggested in the previous chapter. Furthermore, when the resulting knot vector produces an ill-conditioned or rank-deficient system, which is a typical scenario when the resolution of the reconstruction mesh locally exceeds that of the input points, we propose a variable regularization scheme that ensures a full rank system and yields smooth results while maintaining details of interest in the reconstruction. We show that our variable regularization scheme outperforms the commonly used uniform regularization techniques in terms of the reconstruction quality.

5.1 Introduction

When it comes to high dimensional data representation, unstructured grids are a popular choice, owning to the flexibility they provide with unrestricted point sampling, cell shape varieties, and arbitrary cell connectivity, allowing for local refinement

and arbitrary boundary geometry. However, this flexibility comes at the price of an increase storage cost, and inefficient random access and interpolation, which are often performance bottlenecks in the processing of unstructured grids. Often supplementary data structures are built to aid in efficiency of random access operations (e. g., [51]). Specific numerical analysis such as high order derivatives estimation also proves a challenge for unstructured grids.

In comparison, a rectilinear grid data model makes these post-processing tasks straightforward. Indeed, a rectilinear grid is highly storage-efficient and provides fast random access and interpolation operations without additional data structure due to its implicit cell location and connectivity. Using high order B-splines as the reconstruction basis, high order derivative evaluation also becomes straightforward with a rectilinear grid. Hence, an accurate rectilinear reconstruction of an input unstructured point set offers a compelling alternative data representation for a variety of data analysis tasks.

In many cases, unstructured grids are constructed with highly nonuniform point density. This happens often when higher accuracy is needed in localized regions in scientific simulations, geometric modeling, and other data acquisition methods. While Chapter 4 covers the problem of quickly determining a suitable knot vector for an accurate fit of tensor-product B-splines on unstructured grids, when the point distribution of the input data is significantly uneven in density, a unique set of challenges arise in the reconstruction process. In this chapter, we address two of the problems that arise in this context.

Firstly, due to the uneven distribution of the data, the method of evaluating the derivatives over the input data needs to be reconsidered. In the previous chapter, a uniform grid is used to sample the derivatives of the input data, with the resolution of the grid determined by the number of the input data. However, when the point distribution is highly nonuniform, using a uniform grid sampling can result in an aliasing problem where information is not captured at regions where there are more input points than grid points. Choosing a grid whose resolution is as fine as the

densest region of the input data could prevent the aliasing problem. However, a grid that is fine enough can incur a high computational cost unnecessarily if the input point sampling is highly nonuniform. A method is needed that adaptively samples the input and seamlessly connects to the subsequent tasks in the knot placement pipeline.

Secondly, by the nature of tensor product construction, the uneven distribution of features in the input data can create regions of dense knot placements in regions with few input points. In many cases where there are regions with more knots than input points, resulting in more unknown control coefficients to solve for than the local number of constraints, the system becomes ill-conditioned or rank deficient. Solution to such systems typically have oscillation artifacts in the resulting reconstruction. A common solution to this problem is to use a regularization term that adds smoothness constraints to the reconstruction, thereby conditioning the linear system. A challenge for such a method lies in choosing the regularization parameter that balances accuracy and smoothness. However, it is often impossible to find a unique parameter value that strikes that balance across all fitting constraints.

To avoid sampling too densely and wasting computational resources in the generation of feature functions, the uniform grid used previously is replaced by a subdivision tree (quadtree/octree), which is built based on the input data point distribution. The tree is then used to adaptively evaluate the derivatives at different refinement levels. To achieve a high accuracy least squares approximation of a nonuniformly distributed data, a variable regularization is introduced that enforces smoothing on the resulting approximation based on the input point distribution. The proposed variable regularization ensures the fitting problem to be well-conditioned while balancing the smoothness and accuracy. The results obtained with this approach capture the details in the input dataset accurately while preventing the under-constraint regions from producing oscillatory visual artifacts.

This chapter is formatted as follows. Section 5.2 discuss related work on structured grid reconstruction of unstructured data. Section 5.3 presents the more effi-

cient quadtree-based derivative evaluation for generating feature functions in our knot placement method for nonuniformly distributed data. Section 5.4 propose the variable regularization scheme as an alternative to the commonly used uniform regularization method to account for the ill-conditioned system caused by the nonuniformity of the input data points. Results are shown in Section 5.5, followed by Section 5.6 where we conclude this chapter and discuss possible extensions to this work.

5.2 Related Work

The challenge of reconstructing scattered data with irregular sample density using a structured grid is studied extensively in the literature.

A naive way to avoid the rank deficiency problem caused by a nonuniformly distributed input is to opt not to allow for knot spans that are too small such that there are knot spans with no input point, as proposed by Zhang *et al.* [42]. Yet this simplistic approach places a undesirable limit on how fine the reconstruction grid can be, thereby adversely affecting the reconstruction accuracy.

To allow for reconstruction on a finer grid, a number of prior works consider a hierarchical construction on uniform grids. Lee *et al.* proposed using hierarchical refinement B-splines (HB-splines) [52] to approximate irregularly spaced points with a coarse-to-fine hierarchy of overlapping control lattices to generate a sequence of bicubic B-spline functions whose sum approaches the desired interpolation function. Zhang *et al.* [53] improved the memory usage of HB-splines by limiting the refinement in subsets of the domain that needs the refinement. The resulting surfaces from these techniques tend to zero when there is no constraint, which is not a favorable outcome for some applications. Bertram *et al.* [54] presents a fast method that adaptively approximates large data sets with highly non-uniform sampling densities with hierarchical B-splines for terrain visualization. This is accomplished through hierarchical layers of successively more refined grid guided by quadtrees, and fusing the resulting surface components by knot removal.

Regularization is a standard approach to address the ill-posedness of a linear system, and is well known in approximation theory and statistics. In the context of B-spline data fitting, smoothness of fit, also called roughness minimization, is often chosen as additional constraint to achieve good visual results [55, 56]. Though roughness minimization in reconstruction has mostly been done for images, and largely unexplored for rectilinear grids.

A drawback in this regularization is that penalizing the square of the roughness (in a least squares setting) causes over-smoothing of sharp features in the image. This is undesirable when features should be preserved by the reconstruction. On the other hand, L-1 regularization, which minimizes the absolute value of derivatives, allows a few large values of derivative magnitude. This allows sharp edges but result in spike-like artifacts that appear to be remnant of sample density distribution [57]. It was reported by Francis *et al.* that reconstructions using second derivatives result in more structurally similar results [58].

Vazquez *et al.* [59] proposed a method for image reconstruction from nonuniform samples using B-splines by imposing regularization on the energy of squared second derivative of the reconstructed image in spline spaces. Arigovindan *et al.* [60] propose image reconstruction on irregularly sampled points using a smoothness constraint that minimizes the sum of all possible partial derivatives of a given order, and furthermore developed a fast multigrid algorithm to solve the system. Their method is limited to uniform grids and requires a hand-picked smoothing parameter. Subsequent work by Vuçini *et al.* extends the work to 3D volumes [61] and proposed using different regularization constants for each spatial direction to deal with anisotropic characteristics [62].

Another class of regularization strategies involves partial differential equations (PDEs) that are based on anisotropic diffusion. These approaches can fine-tune the local reconstruction smoothness with specific diffusion tensors and is often used in edge detection, edge-preserving image de-noising, and sparse image interpolation.

Bourquard and Unser [63] proposed a novel method called *anisotropic interpolation* for reconstructing images from sparse, spatial point measurement. Their reconstruction consists of a series of quadratic regularized reconstructions. Each quadratic regularization is constructed using first-order derivatives such that it is directionally adaptive by making use of the information from the previous reconstruction in the series. However, their regularization method is based on first-order derivatives, and is not suitable for sharp details.

Francis *et al.* [58] propose reconstructing 2D images from noisy and sparsely sampled points through an efficient multi-resolution based weighted regularization method. They solve a series of reconstructions at different resolution levels. The regularization at each level is constructed using a probabilistic model based on the local image structure from the previous reconstruction in the series. Francis *et al.* [57] improved upon that work by using maximum entropy principle and directionally adaptive second-order derivatives for determining the probability model for regularization designed to further reduce distortions.

The overwhelming majority of the prior works on scattered point approximation with nonuniform sampling distribution are focused on reconstructing regular image on a uniform grid.

T-splines, first introduced by Sederberg *et al.* [64], allow for adaptive local refinement of B-splines, providing advantages over tensor-product B-splines for approximating nonuniformly distributed scattered data. Several works proposed methods for optimizing a T-spline fitting for unstructured data [65,66]. However, the iterative nature of the fitting process and the non-tensor-product construction of T-splines, which complicates the basis constructions, make these methods computationally expensive.

ElRushaidat *et al.* [67] proposed a multilinear reconstruction of scattered points on a rectilinear grid with nonuniform grid spacing using a two-level regularization parameter. The choice of the regularization parameter is determined per grid point using the number of scattered points in a grid cell and the highest value constraint from the scattered point.

5.3 Adaptive Derivative Evaluation

In the previous chapter, we presented a method of knot placement for multivariate unstructured dataset using the partial derivatives of the input data estimated on a regular grid whose grid resolution is determined by the number of input points. The evaluated derivatives on the regular grid are reduced to a 1D function per dimension that represents the directional feature of that particular dimension (See Figure 4.3). This feature function is then used to place knots for the particular dimension.

For scattered data whose point distribution is highly nonuniform, estimating the derivatives on a regular grid with uniform resolution can be problematic. The uniformity of the grid can miss detail at locations with point density higher than the grid resolution, and evaluating the derivatives on a fine grid at locations with few input points can be a waste of computing resources.

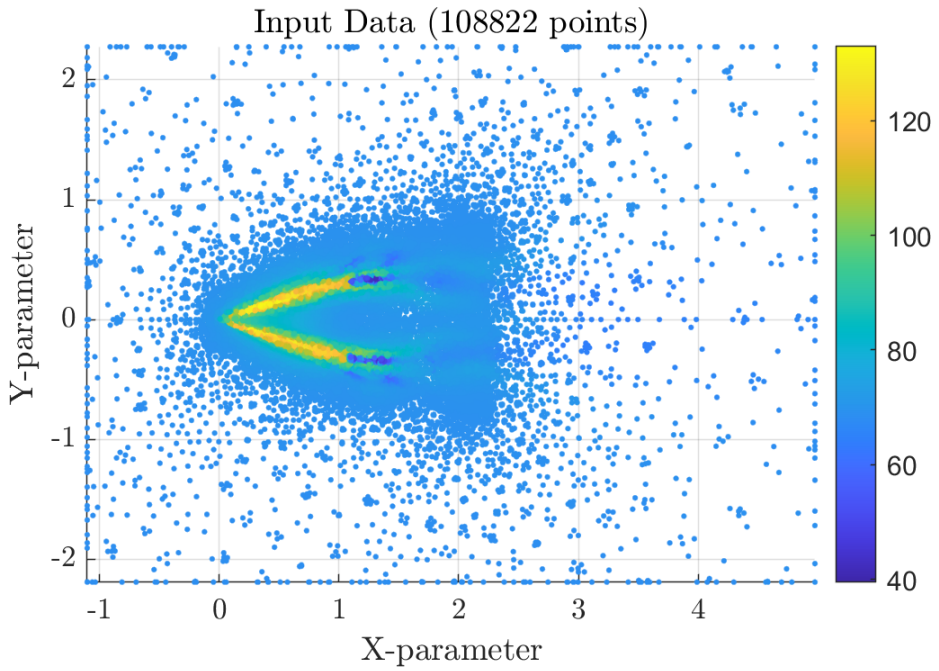


Fig. 5.1.: Input data with 109k points of a 2D slice of the Edelta wing simulation, points colored by velocity magnitude.

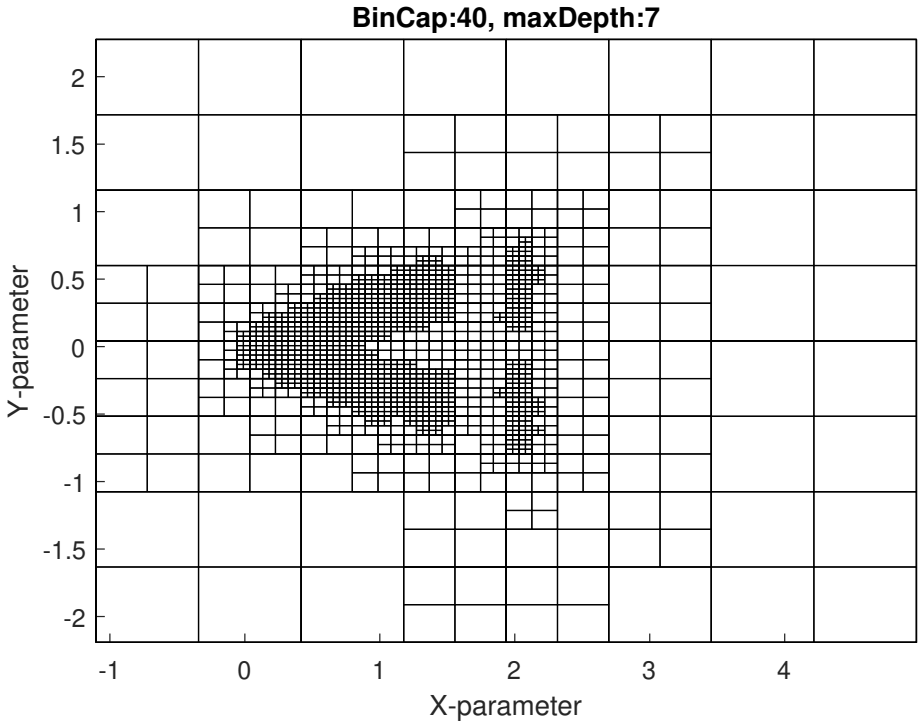


Fig. 5.2.: Quadtree constructed based on the point distribution of the 2D Edelta wing.

To overcome this problem, we propose an adaptive resolution evaluation method. The description below is limit to a 2D domain, however the method can be extended to higher dimension trivially. Using the input point distribution, we subdivide the 2D domain adaptively with a quadtree, the quadtree refinement then represents the resolution at which derivatives should be evaluated on. The quadtree is constructed by subdividing the domain into four for a 2D domain, or eight for a 3D domain, if the number of points inside a tree bin is higher than a predetermined threshold. This threshold is data-dependent, and is determined by the expected smoothness of the dataset in relation to the point distribution. Higher thresholds result in a less refined tree nodes with more points per leaf node, and lower threshold result in a more refined tree with more leaf nodes. Figure 5.2 shows the quadtree built on the input data in

Figure 5.1 with the point threshold set to 40. For visual clarify, the quadtree depth is limited to 7. The actual quadtree built has a deeper depth.

The quadtree provides a good representation of the point distribution of the input data. With the assumption that point distribution is correlated with the complexity of the data, it is reasonable to assume that higher density of points results in the same number of samples in a smaller space. Once the quadtree is built, the estimated derivatives are evaluated at the vertices of the quadtree. The method of derivative evaluation is the same as described in the previous chapter (Section 4.3), where a local least squares fit is done on the input points based on Taylor polynomials, and the derivative of the polynomial fit is evaluated.

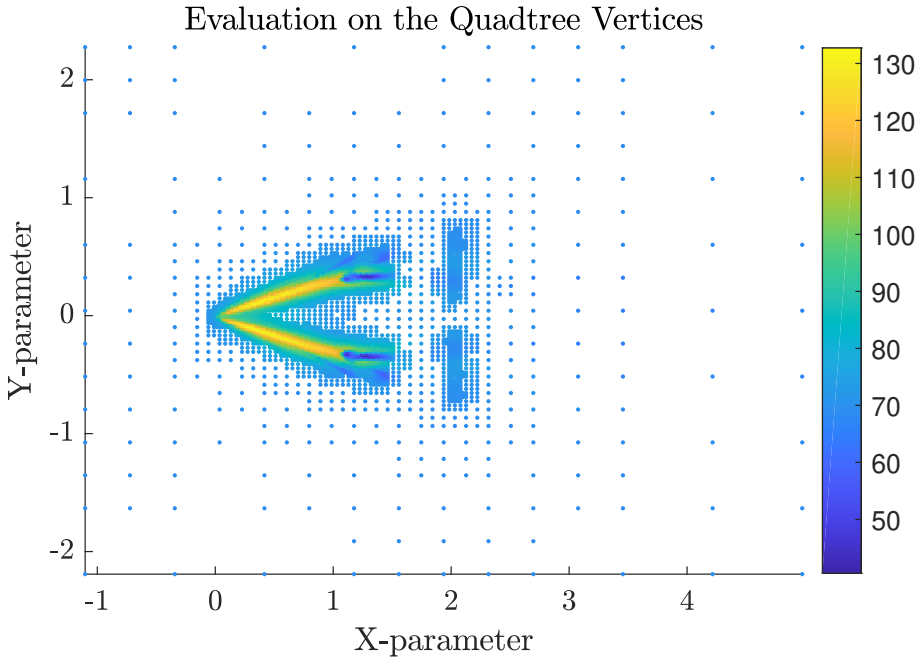


Fig. 5.3.: Evaluation of the velocity magnitude at the quadtree vertices using the quadtree in Figure 5.2.

Figure 5.3 shows the locations of the quadtree vertices in Figure 5.2. Figure 5.4 shows the result of evaluating the third partial derivatives in the Y-direction of the 2D Edelta wing dataset using the full resolution of the quadtree. The colors are filled-in

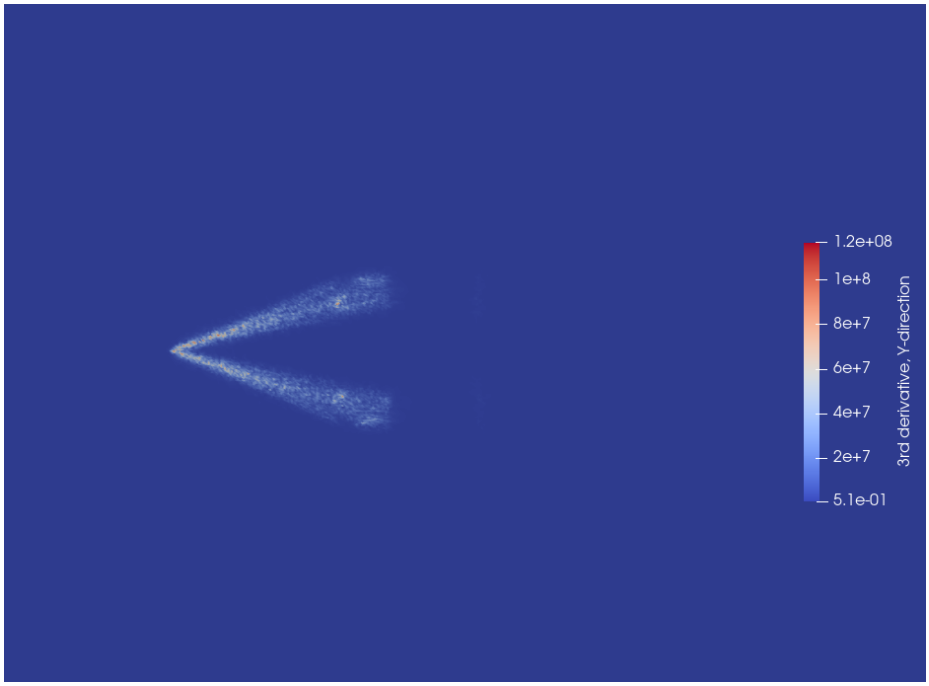
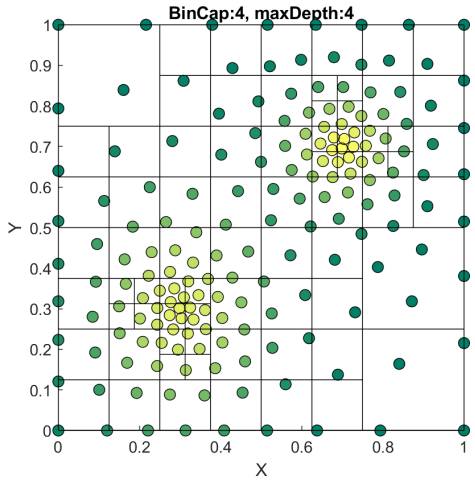


Fig. 5.4.: Evaluation of the third derivatives in the Y-direction of the 2D Edelta wing.

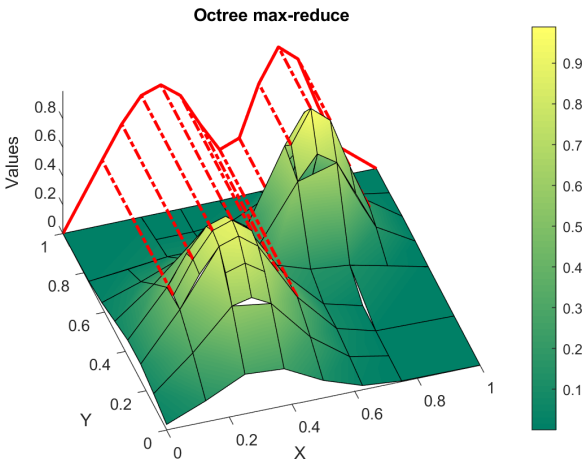
compared to Figure 5.3 for easier visual. Using the quadtree focuses the evaluation at locations where more sampling may be needed to capture the data features.

Once the partial derivatives are evaluated at the vertices of the quadtree nodes, the next step is to build the feature function for each dimension. To do so, we use linear interpolation along a quadtree edge. Consider a tree node to be a rectangle (or a hexahedron in 3D), we sample the edges of the rectangle (resp. hexahedron) linearly. The maximum of the values along the edges are then taken in each direction of the partial derivatives. The maximum value is equivalent to the max derivative set F_u in Equation (4.10). The rest of the knot placement proceed as described in the Section 4.3 using the max-derivative set F_u and its corresponding parameter.

Figure 5.5 shows an example of a simple dataset with two peaks with nonuniformly point density. A quadtree is build over the dataset, and the maximum value of the evaluated value at each node of the quadtree is taken along the X-direction. Note that while the example shown here is evaluating the quadtree estimating the data



(a) Example of a quadtree generated on points with 2 Gaussian bumps.



(b) Reducing the quadtree values to 1 dimension by the max values (red lines).

Fig. 5.5.: Example of 2D maximum-reduction using a quadtree.

values directly and taking the per-direction-maximum of the values, this quadtree maximum method is for evaluating the high order derivatives of the input dataset.

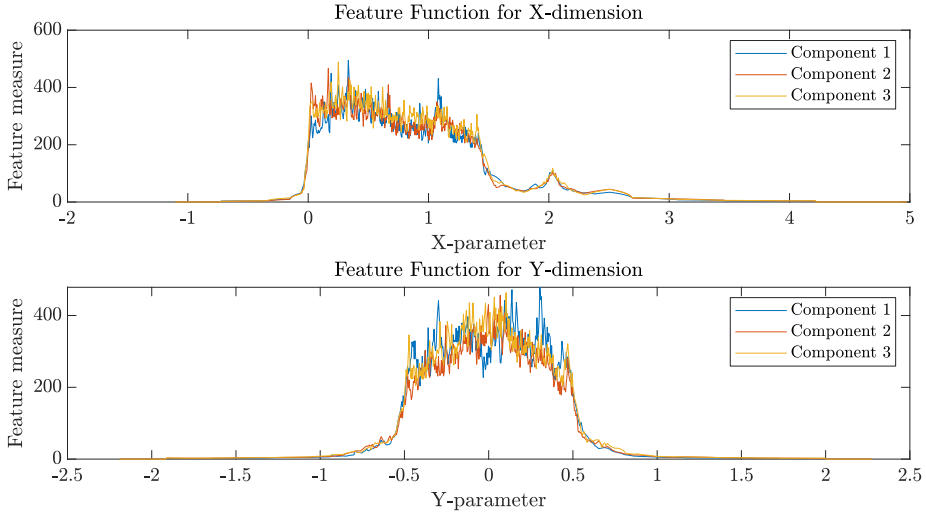


Fig. 5.6.: Feature functions for the velocity variable in the X- and Y-parameter directions, constructed using the third derivatives estimation. The three colors are for each component of the velocity.

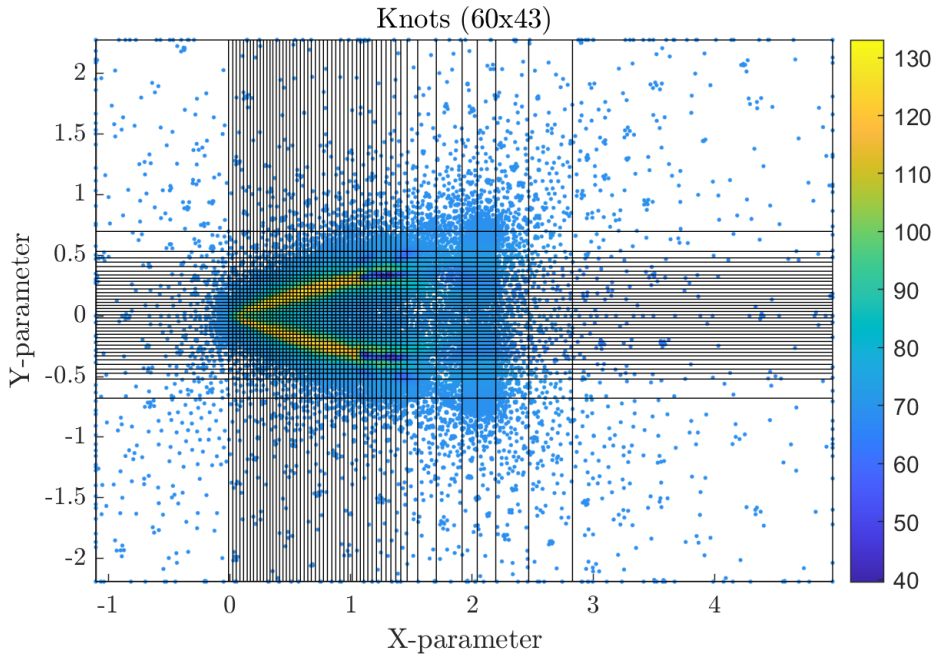


Fig. 5.7.: Knot vectors placed using the feature functions in Figure 5.6.

Using this quadtree evaluation method, 7.7k evaluations were done with the 2D Edelta wing dataset with a quadtree of depth 10, which would have resulted in a grid of 1 million points to evaluate at the resolution of the finest quadtree bin.

Figure 5.6 shows the feature functions for the X and Y-dimension of the Edelta wing data calculated by finding the maximum value of the quadtree edges using linear interpolation. Figure 5.7 shows the resulting knot placement with this feature functions.

5.4 Adaptive Regularization

With the B-spline degree and knot vector determined, the missing control coefficients can be solved with a linear least square system in Equation (2.8). However, when the input data contain nonuniform point distribution, the matrix **A** is likely to be rank-deficient at parameter locations where there are more control coefficients

than scattered points, producing a system with more degrees of freedom than number of constraints locally.

A rank-deficient system has an infinite number of solutions (or in this case, infinite number of least squares solutions). A typical way to choose a unique solution is by finding the solution that minimizes the norm of the solution vector in the infinite solution space. We use `lsqminnorm` function in `MATLAB` for rank-deficient cases. However, the resulting solution will most likely contain unwanted artifacts. Figure 5.8 shows the reconstruction of the B-spline approximation of the minimum-norm solution of the 2D slice of the Edelta wing using the knot vectors from Figure 5.7. Locations with more unknown control points than input scattered points result in oscillatory artifacts in the reconstruction, most notably at the border where long skinny cells are formed by closely packed knots on one dimension but not the other.

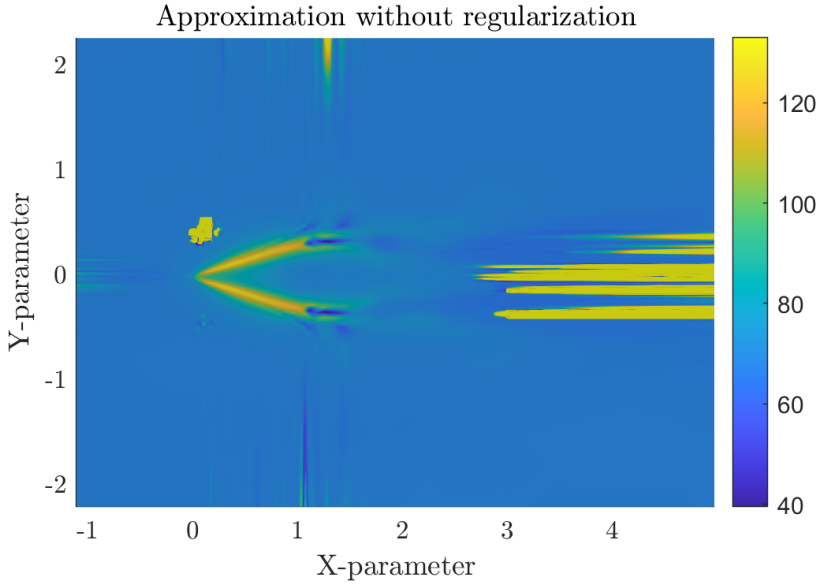
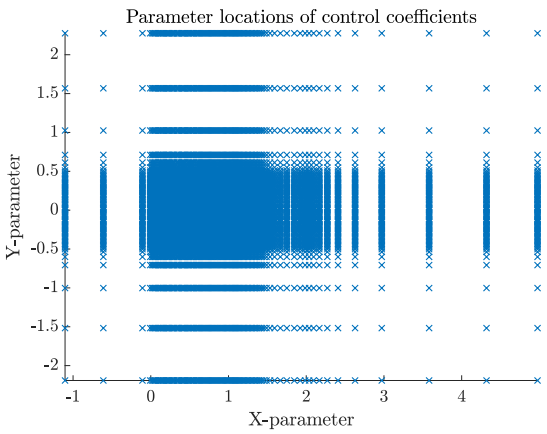
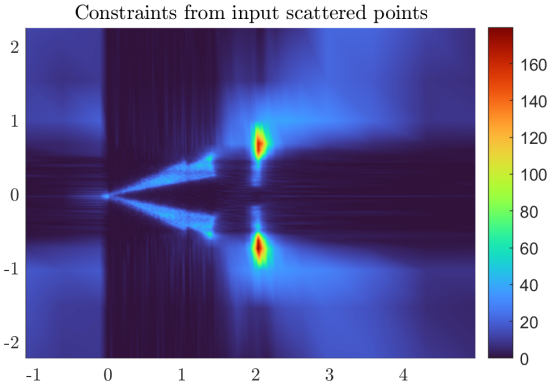


Fig. 5.8.: Least squares reconstruction of a rank-deficient system results in oscillatory artifacts at locations with not enough constraints.

A naive way of measuring the amount of constraints per control point is by looking at the absolute sum of the column of the linear system $\mathbf{A}$ in Equation (2.9). Matrix $\mathbf{A}$ contains the B-spline basis functions evaluated at each input point parameter location.



(a) $\boldsymbol{L}$, the parameter locations representing each control point.



(b) s_i, the column sum of $\mathbf{A}$ that indicates the amount of constraints from the input scattered points for each control point.

Fig. 5.9.: Visualizing amount of constraints per control point.

Each column of $\mathbf{A}$ corresponds to the constraints on a control point, and each row corresponds to an input scattered point's constraints on the control point. A control point that has few scattered points in its basis support, or with the scattered points almost outside of its basis support (such that the corresponding basis value in the matrix is close to zero), would have no or very little constraints.

To visualize the constraints on each unknown control point, we define the following. We pick a parameter location to represent each unknown control point in the system. Let $\boldsymbol{L}_i$ be the representative parameter location of the i^{th} control point, defined as the parameter location $\boldsymbol{u}$ with the maximum value of a basis function $N_{i,p}$.

$$\boldsymbol{L}_i = \underset{\boldsymbol{u}}{\operatorname{argmax}}\, N_{i,p}(\boldsymbol{u}) \tag{5.1}$$

Figure 5.9a shows the parameter location $\boldsymbol{L}_i$ for each control coefficients associated with the knots in Figure 5.7.

Let s_i be the column sum of $\mathbf{A}$, estimating the amount of constraints from the input points for a control point $\boldsymbol{c}_i$.

$$s_i = \sum_{j=1}^{m} N_{i,p}(u_j) \tag{5.2}$$

Figure 5.9b colors each point of L_i by s_i, showing the amount of constraints each control point has from the input points. As the amount of constraints is directly related to the point distribution, a dark blue area in Figure 5.9b indicates control points with low constraints that may result in undesirable, numerically unstable solution. Indeed, correlation can be seen between the low values (blue area) and region in the reconstruction in Figure 5.8 that produces highly oscillation result. However, note that the column sum s_i is by no means a direct reflection of the condition of the matrix $\mathbf{A}$. The matrix $\mathbf{A}$ can still be rank-deficient or ill-conditioned even if every control point has a large number of constraints. That can happen when locally input points in a region are overlapped or spans a lower dimensional space than the dataset (e. g., colinear points in a 2D space or coplanar points in a 3D space).

As explained in previous section, adding a regularization term to the system is a common technique to mitigate these types of ill-posed problems. The regularization effective trades closeness of fit against smoothness on the solution (Ch 6.1.5 [68]). This is achieved through modifying Equation (2.9) by adding the extra smoothing constraints

$$\underset{\boldsymbol{c}}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| q_i - C(u_i) \right\|_2^2 + \lambda^2 S(C) \right) \tag{5.3}$$

where S is a penalty function on the roughness of the B-spline reconstruction C, and λ controls the relative contribution of the smoothness constraints to the overall error measure. This makes the optimization problem well-posed and have a unique solution.

While there are many choices for the added constraint, in the context of B-spline data fitting, the most commonly used function is to measure roughness by the sum of the second-order partial derivatives of the resulting B-spline reconstruction [55, 56]. This gives

$$S(C) = \int \cdots \int \sum_{d}^{D} \left\| \frac{\partial^2 C}{\partial u_d^2} \right\|_2^2 \mathrm{d}u_1 \ldots \mathrm{d}u_D \tag{5.4}$$

which integrates across the hypervolume the sum of the second partial derivatives in all axial directions.

In 2D, this would be

$$S(C) = \int \int \left(\left\| \frac{\partial^2 C(\boldsymbol{u})}{\partial u^2} \right\|_2^2 + 2 \left\| \frac{\partial^2 C(\boldsymbol{u})}{\partial u \partial v} \right\|_2^2 + \left\| \frac{\partial^2 C(\boldsymbol{u})}{\partial v^2} \right\|_2^2 \right) \mathrm{d}u \, \mathrm{d}v \tag{5.5}$$

with $\boldsymbol{u} = (u, v)$.

We discretize the integral by taking the sum of samples of the second derivatives at the n control coefficient parameter locations $\boldsymbol{L}$.

$$S(C) = \sum_{i=1}^{n} \sum_{d}^{D} \left\| \frac{\partial^2}{\partial u_d^2} C(L_i) \right\|_2^2 \tag{5.6}$$

Approximating data with B-spline of order higher than 2 allows for analytical derivatives of the approximating B-spline. Recall from Section 2 that the derivatives of an order-p B-spline curve is another B-spline of order $p-1$, calculated using subset of the knot vectors and the original control points with Equations (2.5) and (2.6). For higher order B-spline in a tensor product construction, the derivatives are calculated similarly. As each high dimensional basis functions are a product of the 1D B-spline basis in each dimension, the partial derivatives in one direction is calculated by differentiating the basis along that dimension. One can thereby construct the interpolation equation of the second derivatives of the B-splines at any given parameter locations. We thus construct the regularization by appending the linear system in Equation (2.8) with the zero-second-derivatives constraints.

$$\begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{S} \end{bmatrix} \boldsymbol{x} \approx \begin{bmatrix} \boldsymbol{b} \\ \mathbf{0} \end{bmatrix} \tag{5.7}$$

$\mathbf{A}$, $\boldsymbol{x}$, and $\boldsymbol{b}$ are in the original equation representing respectively the B-spline basis of the input data locations, the unknown control coefficients, and the input data values. Matrix $\mathbf{S}$ is the basis of the second derivatives of C evaluated at parameter locations $\boldsymbol{L}$. This linear system solves Equation (5.3).

The choice of λ, the regularization parameter, works as a trade-off factor between closeness of fit and smoothness. A high λ value results in smooth reconstruction that may not be accurate, whereas a small λ value results in a more accurate reconstruction but with potential numerical issues and artifacts.

The bottom two reconstructions in Figure 5.12 show what happens with low and high λ. The low $\lambda = 1$ result in high accuracy reconstruction at the locations with high number of points, but fluctuations and artifacts at locations with few points. The right reconstruction with $\lambda =$ 1e3 has a smooth fit without the artifacts, but also noticeable loss of detail in the fit.

Inspired by the bi-level smoothing scheme by ElRushaidat *et al.* [67], we propose choosing λ independently for each control point. The λ would be a function of the amount of constraints the control point has from the input scattered points. A control coefficient with few constraints will be assigned a higher λ value, whereas control points already with sufficient constraints will have a low or zero λ value.

Let s_i be the constraints from input data for a control point $\boldsymbol{c}_i$

$$s_i = \sum_{j=1}^{m} N_{i,p}(u_j) \tag{5.8}$$

which is the sum of the basis function at the parameter location of the input data. This is equivalent to the sum of the i-column of the matrix $\mathbf{A}$.

We choose a tolerance for the lowest constraint value allowed as s^*. The amount of constraint we want to add is

$$s_i^+ = max\,(0, s^* - s_i) \tag{5.9}$$

We then calculate each λ_i value per control point $\boldsymbol{c}_i$ with

$$\lambda_i = s_i^+ / \sum_j |S_{i,j}| \tag{5.10}$$

where $S_{i,j}$ is the (i, j) entry in the smoothing matrix $\mathbf{S}$. Here, λ_i is calculated as the ratio between the constraints we want to add to the sum of the column of matrix $\mathbf{S}$ associated to the particular control point.

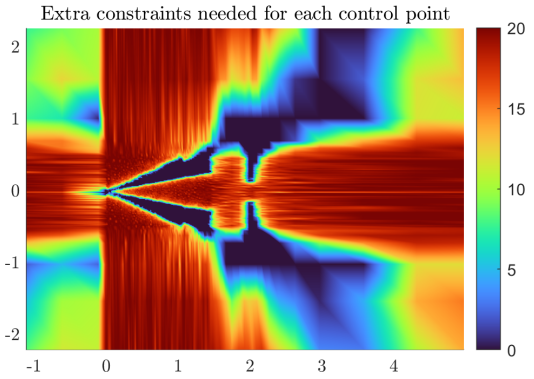
This gives us the updated interpolation equation from Equation (5.11) to use individual λ values

$$\underset{\boldsymbol{c}}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| q_i - C(u_i) \right\|_2^2 + \sum_{i=1}^{n} \lambda_i^2 S(C(\boldsymbol{L}_i)) \right) \tag{5.11}$$

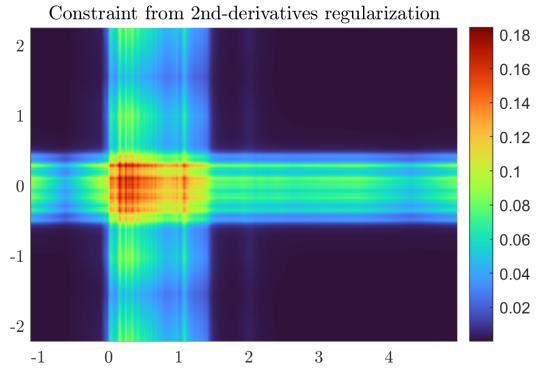
or scaling each row of the constraint matrix

$$\begin{bmatrix} \mathbf{A} \\ \boldsymbol{\lambda}\mathbf{S} \end{bmatrix} \boldsymbol{x} \approx \begin{bmatrix} \boldsymbol{b} \\ \mathbf{0} \end{bmatrix} \tag{5.12}$$

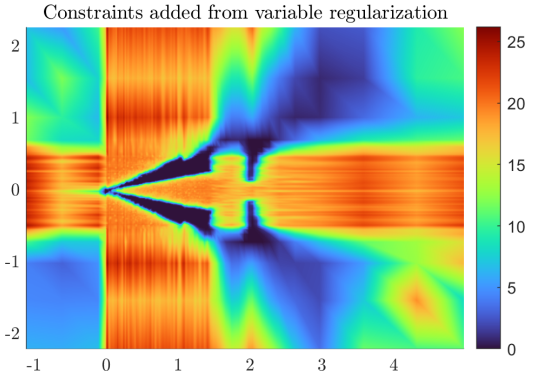
where $\boldsymbol{\lambda}$ is the diagonal matrix with $\{\lambda_1 \cdots \lambda_n\}$ on the diagonal that contains the variable regularization parameters.



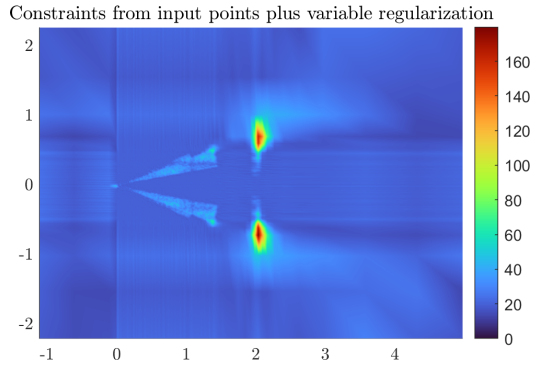
(a) s_i^+, the amount of constraint we aim to add, with target constraint value of $s^* = 20$.



(b) Column sum of the $\mathbf{S}$ matrix, measuring constraints added from the regularization.



(c) Column sum of the matrix $\boldsymbol{\lambda}\mathbf{S}$



(d) Column sum of the final regularized matrix using variable $\boldsymbol{\lambda}$.

Fig. 5.10.: Calculate $\boldsymbol{\lambda}$ based on constraints.

Figure 5.10 shows the various constraint calculations for the 2D Edelta dataset. Figure 5.10a visualize the s_i^+ values of each control points using a constraint threshold of $s^* = 20$. Higher values indicate that more constraints should be added. Fig-

ure 5.10b shows the constraints from minimizing the sum of the second-order partial derivatives at the parameter location $\boldsymbol{L}$. Due to the nonuniform knot placement, the constraints from the second derivatives gives higher constraints for the shorter knot spans. This results in a horizontal and vertical segments where the values are higher due to the finer knot spacing. After calculating the λ_i for each control points, Figure 5.10c shows the column sum of $\boldsymbol{\lambda}\mathbf{S}$, which are constraints from the lower regularization part of the matrix. It is equivalent to the amount of constraints added to the linear system to regularize it, and should closely match Figure 5.10a. Figure 5.10d show the column sum of the system in Equation (5.12), which shows the total constraints from both the input scattered points and from the regularization. While the points with higher values are mostly unchanged, there are no longer dark blue regions that represent a lack of constraints.

This method of variable regularization applies heavier constraints to control coefficients with few constraints from the scattered input data, but few constraints to already heavily constraint control points. It effectively condition the part of the linear system where necessary without influencing the approximation result for more heavily constraint regions.

5.5 Results

We present the approximation results of two 2D data taken from physics flow simulations, with highly varying input point locations as needed by the simulations.

To normalize the effect of the different sizes of the dataset, we scale the dataset domain such that the smallest element in the dataset has the same edge length. While this may not be the most accurate method of normalization, it suit the demonstration purpose of this paper.

The first dataset is a 2D slice of the Edelta dataset, a Navier-Stokes simulation of wind flow around a delta-shaped wing at low speed and high angle of attack.

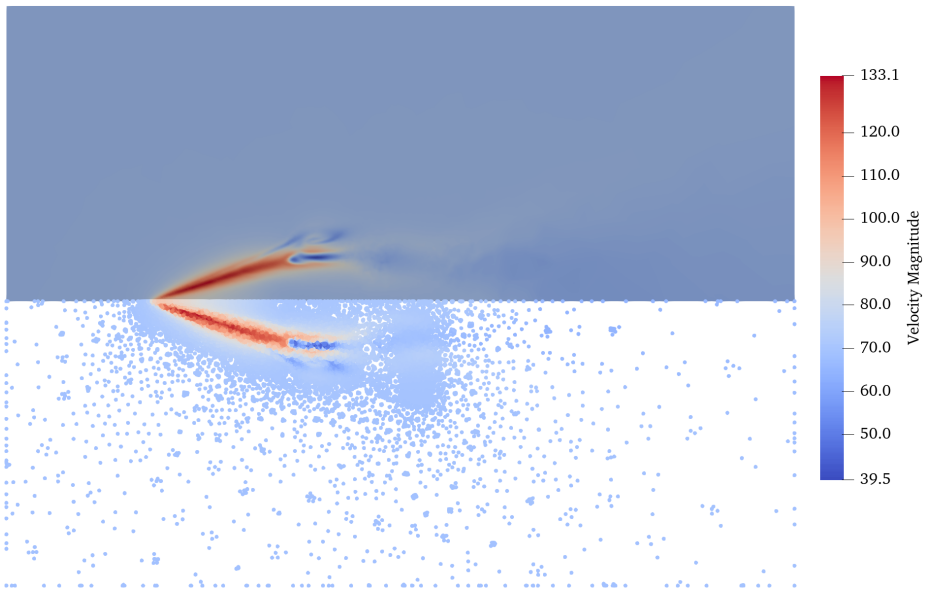


Fig. 5.11.: Edelta wing dataset with 109k points, colored by velocity magnitude. Lower half of the figure shows the point distribution.

Figure 5.11 show the slice of the delta wing colored by velocity magnitude. The top half shows the triangulated surface, and the bottom half shows the point distribution.

The $s*$ value, like the λ value in uniform regularization, is data-dependent, and needs to be chosen for each dataset.

Figures 5.12 and 5.13 compare the input unstructured dataset with reconstructions using different values of λ on the same knot vectors and order-3 B-splines. We compare our proposed constraint-guided variable-λ regularization with two single-λ regularization of low and high values (1 and $1e3$ respectively). Figure 5.12 shows the whole 2D slice of the data, whereas Figure 5.13 shows the close-up view of one of the vortices above the wing.

A low λ value results in oscillatory artifacts in the region of small knot spacing, as visible in Figures 5.12 especially on the top and right side. Small oscillatory artifacts is also visible in the close-up view in Figure 5.13 of the low-smoothing reconstruction to the right side of the vortices formed as well. On the other hand, a high λ value produces an over-smoothed reconstruction, as can be seen in both figures, but

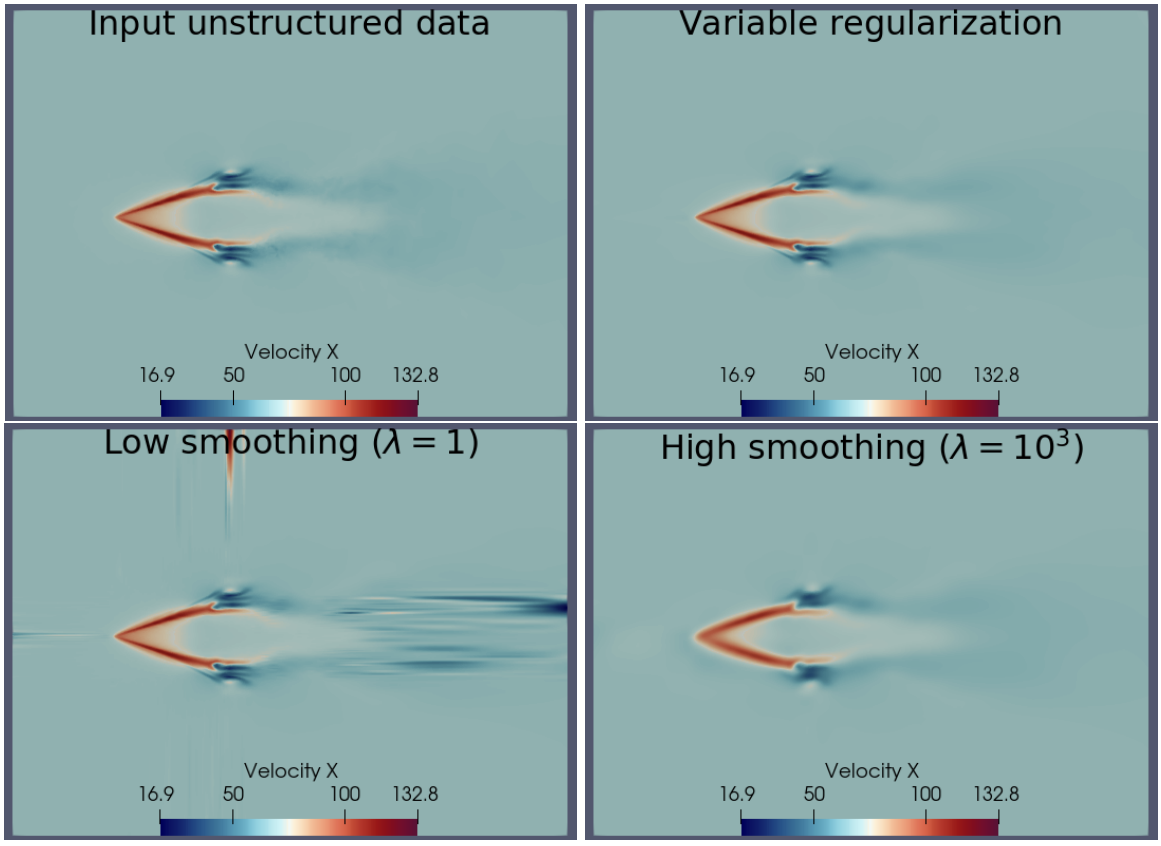


Fig. 5.12.: Comparing the slice of Edelta reconstruction using various regularization tuning parameter.

particularly noticeable in Figure 5.13 where the details of the vortices are blurred out. With our proposed constraint-weighted smoothing, the resulting reconstruction accurately capture the structure and topology of the vortices, similar to with the low smoothing reconstruction.

With uniform λ value of 1 and $1e3$, the condition number of the $\mathbf{A}$ matrix is 524 and $1.5e4$ respectively. With our variable λ, the condition number is 176, a better conditioned matrix than either of the uniform λ.

The second dataset we consider is a 2D slice of the ICE train dataset, a simulation of a high-speed train traveling at a velocity of 250 km/h with wind blowing from the side at an angle of 30 degrees. Figure 5.14 shows the point distribution of the Z-

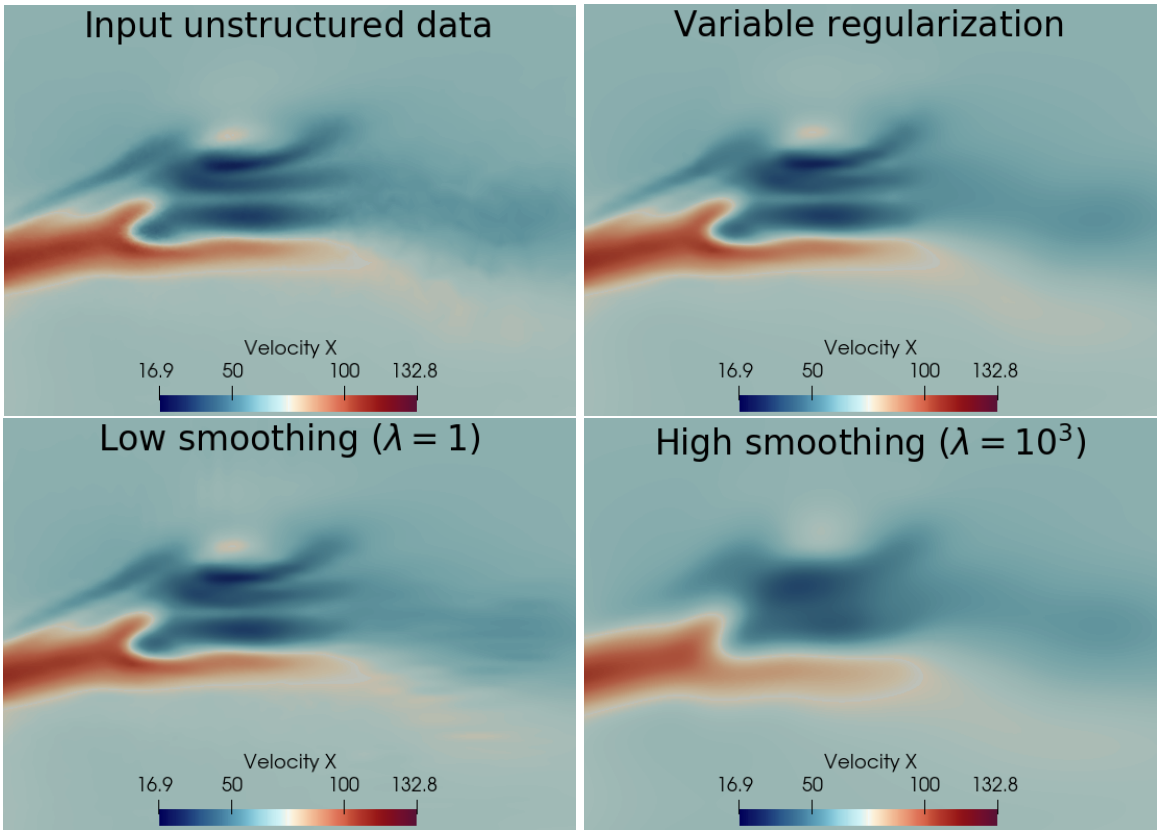


Fig. 5.13.: Zoom-in comparison of Figure 5.12.

component of the velocity. The input points are highly refined around the train body in the middle, and more sparse away from the body of the train. Figure 5.15 shows the knot vectors used for this dataset.

Figure 5.16 and 5.17 compare the input unstructured data with reconstructions of our proposed variable smoothing with a low and high smoothing for the ICE train dataset. Figure 5.16 shows the whole slice of the dataset, whereas Figure 5.17 zooms in on the middle part of the train where two compartments connect. Similar to the result of the Edelta, a low smoothing value results in oscillatory artifacts at locations with few input points but dense knot vectors. This can be seen in Figure 5.16 at the top right corner for the low smoothing result, and also in the close-up result in Figure 5.17 on top of the connection of the two compartments. On the other hand, high smoothing results in a severely blurred out reconstruction, making it difficult to

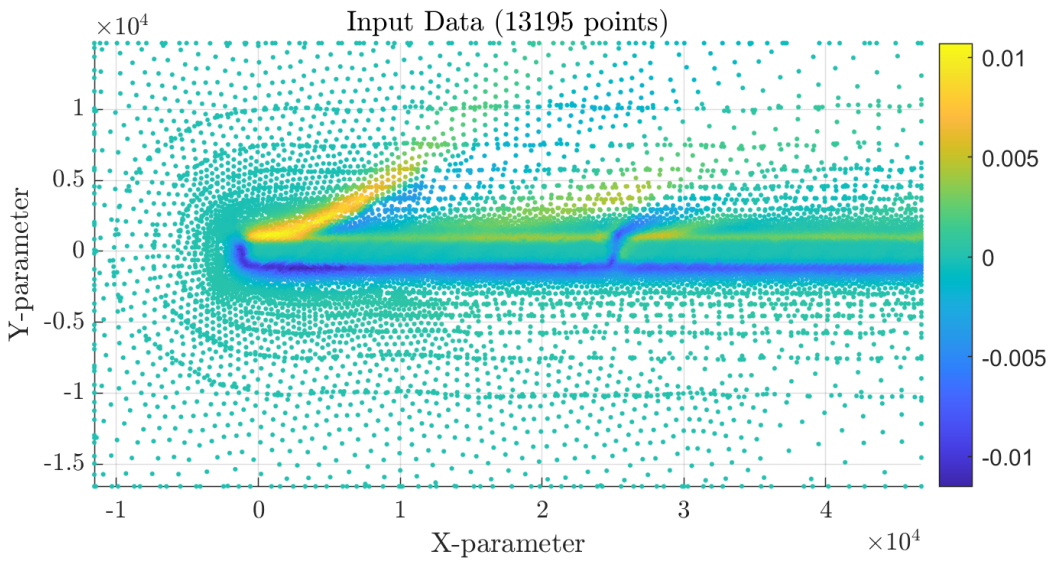


Fig. 5.14.: The ICE train input data points with 13k points, colored by the Z-velocity.

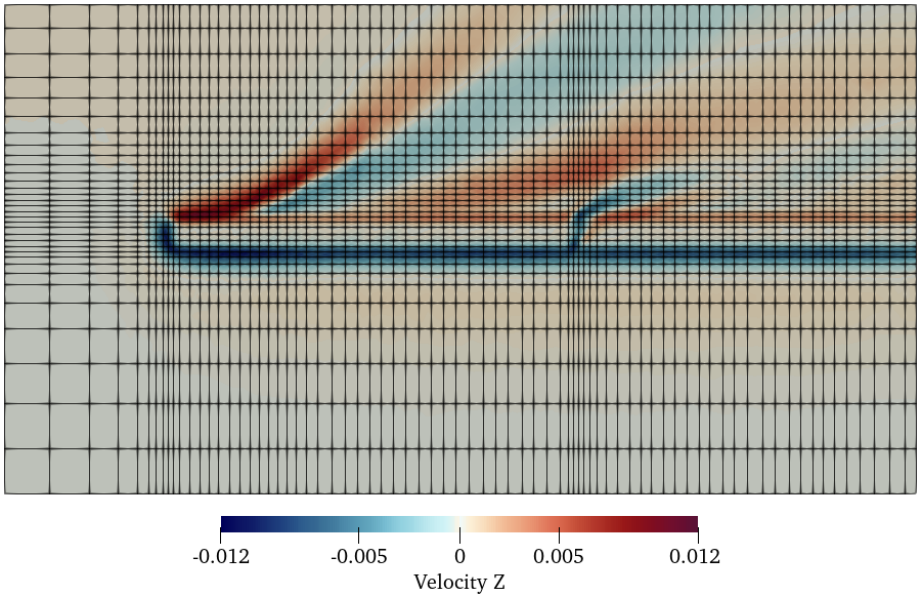


Fig. 5.15.: Knot vectors (80×38) used for the ICE train dataset.

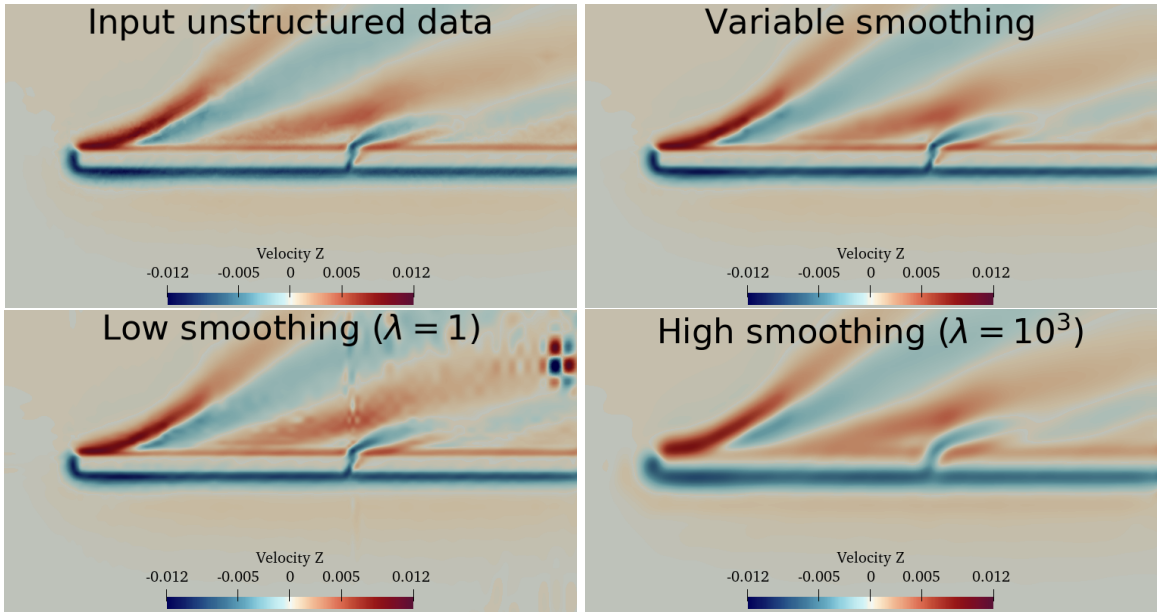


Fig. 5.16.: Comparing the slice of ICE train reconstruction using various regularization λ parameters.

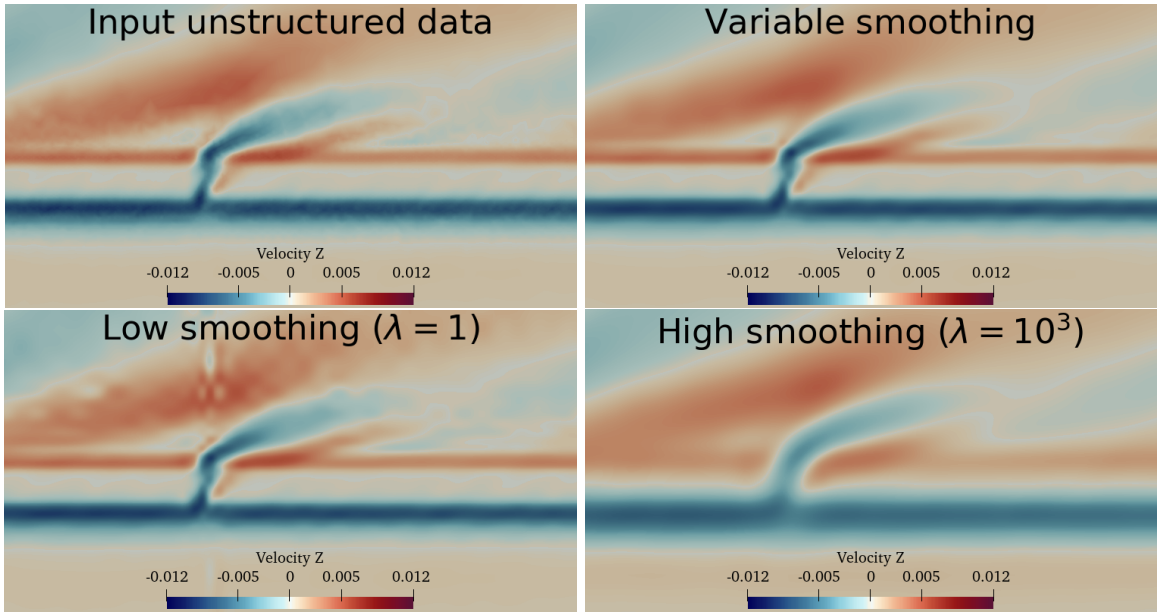


Fig. 5.17.: Comparing a close-up view of the ICE train reconstruction.

identify visual features in both figures. Our variable smoothing reconstruction retains the level of detail present in the low-λ reconstruction, while retaining smoothness in regions where the lack of constraints would otherwise result in visual artifacts.

For the ICE train dataset, the condition number of the $\mathbf{A}$ matrix for uniform λ value of 1 and $1e3$ is 41 and $3.3e3$ respectively. With our variable λ, the condition number is 16. Our variable regularization approach again generated a better conditioned matrix than either of the uniform λs.

5.6 Conclusions

In this chapter, we presented improvements to the previous chapters' tensor product B-spline reconstruction to address cases of highly nonuniformly distributed input data.

We proposed an adaptive derivative evaluation method based on an adaptive subdivision tree (quadtree/octree in 2D/3D, respectively) construction on the input data to efficiently and accurately calculate the feature functions needed for knot placement which takes into account the input data point distribution. This improvement prevents aliasing artifacts in the derivative estimation and reduces computation cost without sacrificing accuracy.

Furthermore, in the cases where nonuniformly sampled input data result in an ill-posed linear system, we propose modifications to the widely-used regularization by roughness minimization with a per-unknown weighted regularization based on the surrounding point density. Not only is the resulting system well-conditioned, the corresponding reconstruction solution also retains details of the input data without over-smoothing, which often occurs with the standard uniform regularization. We show with two 2D simulation datasets that our regularization approach produces higher reconstruction quality and is free of visual artifacts compared to uniform regularization.

Several avenues exist for future work. Our current construction of the regularization is isotropic, as the sum of the second derivatives is considered instead of as separate components. Future work includes taking ideas from anisotropic diffusion to have a regularization that takes into account the local anisotropy of the input data. In addition, criteria other than input point distribution can be considered for a more effective choice of regularization as well. Characteristics of the dataset such as principal curvature or gradient can be used for that purpose.

6. CONCLUSION AND FUTURE WORK

In this dissertation, we consider the problem of B-spline-based data approximation from the point of view of knot optimization. We first consider the problem of accurate approximation of one-dimensional cases of time-series and parametric curves with B-splines. We proposed a method of knot placement that results in highly accurate approximation with fewer control points compared to existing methods. This is achieved by adjusting the density of knots to match the values of a feature function that we derive from the high order derivatives of the data. We discuss important practical considerations such as capping the knot density to prevent a rank deficient system, and how to select the number of knots given a target error tolerance. We show the superior approximation accuracy and efficient performance of our proposed approach compared to state-of-the-art knot placement methods.

In the following chapter, we expanded our method to work with higher dimensional datasets using tensor product B-splines. We proposed derivative calculation methods for structured and unstructured datasets to determine the feature measure, then derived a feature function per dimension to distribute knots. We discussed the choice of the relative number of knots for each dimension given the amount of feature measure in each dimension, and some guidance to determine the number of control points, which was then used to derive the number of knots for each dimension, through linear regression. We showed that our knot placement algorithm produces approximation results with higher accuracy than state-of-the-arts multidimensional knot optimization methods for structured and unstructured datasets alike.

Issues arise in the important special case of highly nonuniform point distributions, where the previously proposed uniform grid derivative estimation can become inefficient. We proposed the use of an adaptive subdivision tree (quadtree in 2D, octree in 3D, and so on) to estimate the derivatives, and taking advantage of the tree vertex

alignment to reduce the high spatial dimension derivatives to a 1D feature function for subsequent knot placement steps. Furthermore, when highly nonuniform point distribution results in an ill-conditioned or even rank-deficient system, we used a common regularization method that adds smoothness as a constraint. To overcome the limitation of the standard uniform regularization, we proposed a variable λ-parameter that adjusts the smoothing weight based on the norm of each column of the linear system, which is itself a function of the relative position of surrounding input points. We show that this variable regularization allows for details to be captured while maintaining smoothness of the reconstruction, which is an elusive goal to achieve with a uniform regularization parameter.

Interesting avenues for future work include improving the quality, stability, and speed of the derivative calculation. While our proposed knot placement method has proven to be effective for accurate reconstruction, the quality of our knot placement method depends heavily on the quality of the derivative calculation. Challenges arise when the input data is noisy, contains discontinuities, or are not densely sampled, making a moving least squares fit difficult or unstable.

Other challenges arise when applying our knot placement method on a non-tensor product construction of B-splines, such as hierarchical refinement spline (e. g., THB-splines), T-splines, triangular splines, and others. Expanding the knot placement methods for nonuniform rational B-splines (NURBS) is another intriguing direction for further research.

REFERENCES

REFERENCES

[1] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[2] S. Yoshizawa, A. Belyaev, H.-P. Seidel, F. Giannini, and A. Pasko, "A fast and simple stretch-minimizing mesh parameterization," *Shape Modeling International 2004 (SMI 2004), IEEE, 200-208 (2004)*, 01 2004.

[3] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*. Natick, MA, USA: A. K. Peters, Ltd., 1993.

[4] T. Várady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models — an introduction," *Computer-Aided Design*, vol. 29, no. 4, pp. 255 – 268, 1997.

[5] T. Peterka, N. Youssef S. G., I. Grindeanu, V. S. Mahadevan, R. Yeh, and X. Tricoche, "Foundations of multivariate functional approximation for scientific data," in *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2018, pp. 61–71.

[6] L. Piegl and W. Tiller, *The NURBS Book (2nd Ed.)*. Berlin, Heidelberg: Springer-Verlag, 1997.

[7] W. Ma and J. Kruth, "Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces," *Computer-Aided Design*, vol. 27, no. 9, pp. 663 – 675, 1995.

[8] D. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*, vol. 15, no. 2, pp. 328–343, 1978.

[9] F. Liang, J. Zhao, S. Ji, C. Fan, and B. Zhang, "A novel knot selection method for the error-bounded b-spline curve fitting of sampling points in the measuring process," *Measurement Science and Technology*, vol. 28, no. 6, p. 065015, may 2017.

[10] V. T. Dung and T. Tjahjowidodo, "A direct method to solve optimal knots of b-spline curves: An application for non-uniform b-spline curves fitting," *PLOS ONE*, vol. 12, no. 3, pp. 1–24, 03 2017.

[11] T. Lyche and K. Mørken, "Knot removal for parametric b-spline curves and surfaces," *Computer Aided Geometric Design*, vol. 4, no. 3, pp. 217 – 230, 1987.

[12] T. Lyche, "A data-reduction strategy for splines with applications to the approximation of functions and data," *Ima Journal of Numerical Analysis*, vol. 8, pp. 185–208, 04 1988.

[13] P. D. Loach and A. J. Wathen, "On the Best Least Squares Approximation of Continuous Functions using Linear Splines with Free Knots," *IMA Journal of Numerical Analysis*, vol. 11, no. 3, pp. 393–409, 07 1991.

[14] G. Beliakov, "Least squares splines with free knots: global optimization approach," *Applied Mathematics and Computation*, vol. 149, no. 3, pp. 783 – 798, 2004.

[15] H. Kang, F. Chen, Y. Li, J. Deng, and Z. Yang, "Knot calculation for spline fitting via sparse optimization," *Computer-Aided Design*, vol. 58, pp. 179 – 188, 2015.

[16] W. V. Loock, G. Pipeleers, J. D. Schutter, and J. Swevers, "A convex optimization approach to curve fitting with b-splines," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2290 – 2295, 2011.

[17] P. Laube, M. O. Franz, and G. Umlauf, "Learnt knot placement in b-spline curve approximation using support vector machines," *Computer Aided Geometric Design*, vol. 62, pp. 104 – 116, 2018.

[18] Y. Yuan, N. Chen, and S. Zhou, "Adaptive b-spline knot selection using multi-resolution basis set," *IIE Transactions*, vol. 45, no. 12, pp. 1263–1277, 2013.

[19] F. Yoshimoto, M. Moriyama, and T. Harada, "Automatic knot placement by a genetic algorithm for data fitting with a spline," in *Proceedings Shape Modeling International '99. International Conference on Shape Modeling and Applications*, March 1999, pp. 162–169.

[20] M. Sarfraz and S. A. Raza, "Capturing outline of fonts using genetic algorithm and splines," in *Proceedings Fifth International Conference on Information Visualisation*, July 2001, pp. 738–743.

[21] O. Valenzuela, B. Delgado-Marquez, and M. Pasadas, "Evolutionary computation for optimal knots allocation in smoothing splines," *Applied Mathematical Modelling*, vol. 37, no. 8, pp. 5851 – 5863, 2013.

[22] E. Ülker, "B-spline curve approximation using pareto envelope-based selection algorithm – pesa," *International Journal of Computer and Communication Engineering*, vol. 2, pp. 60–63, 01 2013.

[23] V. Tongur and E. Ülker, "B-spline curve knot estimation by using niched pareto genetic algorithm (npga)," in *Intelligent and Evolutionary Systems*, K. Lavangnananda, S. Phon-Amnuaisuk, W. Engchuan, and J. H. Chan, Eds. Cham: Springer International Publishing, 2016, pp. 305–316.

[24] A. Galvez and A. Iglesias, "Firefly algorithm for explicit b-spline curve fitting to data points," *Mathematical Problems in Engineering*, vol. 2013, 11 2013.

[25] A. Gálvez, A. Iglesias, A. Avila, C. Otero, R. Arias, and C. Manchado, "Elitist clonal selection algorithm for optimal choice of free knots in b-spline data fitting," *Applied Soft Computing*, vol. 26, pp. 90 – 106, 2015.

[26] H. Park and J.-H. Lee, "B-spline curve fitting based on adaptive curve refinement using dominant points," *Computer-Aided Design*, vol. 39, no. 6, pp. 439 – 451, 2007.

[27] W. Li, S. Xu, G. Zhao, and L. P. Goh, "Adaptive knot placement in b-spline curve approximation," *Computer-Aided Design*, vol. 37, no. 8, pp. 791 – 797, 2005.

[28] E. Aguilar, H. Elizalde, D. Cárdenas, O. Probst, P. Marzocca, and R. A. Ramirez-Mendoza, "An adaptive curvature-guided approach for the knot-placement problem in fitted splines," *Journal of Computing and Information Science in Engineering*, vol. 18, September 2018.

[29] A. Razdan, "Knot placement for b-spline curve approximation," 06 2000.

[30] T. Tjahjowidodo, V. Dung, and M. Han, "A fast non-uniform knots placement method for b-spline fitting," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2015, pp. 1490–1495.

[31] C. Conti, R. Morandi, C. Rabut, and A. Sestini, "Cubic spline data reduction choosing the knots from a third derivative criterion," *Numerical Algorithms*, vol. 28, no. 1, pp. 45–61, Dec 2001.

[32] Xuming He, Lixin Shen, and Zuowei Shen, "A data-adaptive knot selection scheme for fitting splines," *IEEE Signal Processing Letters*, vol. 8, no. 5, pp. 137–139, May 2001.

[33] L. Piegl and W. Tiller, "Surface approximation to scanned data," *The Visual Computer*, vol. 16, no. 7, pp. 386–395, Nov 2000.

[34] D. Lenz, O. Marin, V. Mahadevan, R. Yeh, and T. Peterka, "Fourier-informed knot placement schemes for b-spline approximation," 2020.

[35] J. Chen, A. Choudhary, B. Supinski, M. DeVries, E. Hawkes, S. Klasky, W. Liao, k. l. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo, "Terascale direct numerical simulations of turbulent combustion using s3d," *Computational Science & Discovery*, vol. 2, p. 015001, 01 2009.

[36] H.-T. Yau, M.-T. Lin, and M.-S. Tsai, "Real-time nurbs interpolation using fpga for high speed motion control," *Computer-Aided Design*, vol. 38, no. 10, pp. 1123 – 1133, 2006.

[37] W.-C. Xie, X.-F. Zou, J.-D. Yang, and J.-B. Yang, "Iteration and optimization scheme for the reconstruction of 3d surfaces based on non-uniform rational b-splines," *Computer-Aided Design*, vol. 44, no. 11, pp. 1127 – 1140, 2012.

[38] P. González, H. Idais, M. Pasadas, and M. Yasin, "Evolutionary computation for optimal knots allocation in smoothing splines of one or two variables," *International Journal of Computational Intelligence Systems*, vol. 11, pp. 1294 – 1306, 09 2018.

[39] H. Idais, M. Yasin, M. Pasadas, and P. González, "Optimal knots allocation in the cubic and bicubic spline interpolation problems," *Mathematics and Computers in Simulation*, vol. 164, pp. 131 – 145, 2019, the 7th International Conference on Approximation Methods and Numerical Modelling in Environment and Natural Resources, held in Oujda, Morocco, May 17-20, 2017.

[40] R. Yeh, Y. S. Nashed, T. Peterka, and X. Tricoche, "Fast automatic knot placement method for accurate b-spline curve fitting," *Computer-Aided Design*, vol. 128, p. 102905, 2020.

[41] H. Park, "B-spline surface fitting based on adaptive knot placement using dominant columns," *Computer-Aided Design*, vol. 43, pp. 258–264, 03 2011.

[42] Y. Zhang, J. Cao, Z. Chen, X. Li, and X.-M. Zeng, "B-spline surface fitting with knot position optimization," *Computers & Graphics*, vol. 58, pp. 73 – 83, 2016, shape Modeling International 2016.

[43] A. Ravari and H. Taghirad, "Reconstruction of b-spline curves and surfaces by adaptive group testing," *Computer-Aided Design*, vol. 74, 01 2016.

[44] H. Lin, G. Wang, and C. Dong, "Constructing iterative non-uniform b-spline curve and surface to fit data points," *Science in China Series F: Information Sciences*, vol. 47, pp. 315–331, 06 2004.

[45] H. Lin and Z. Zhang, "An extended iterative format for the progressive-iteration approximation," *Computers & Graphics*, vol. 35, no. 5, pp. 967 – 975, 2011.

[46] C. Deng and H. Lin, "Progressive and iterative approximation for least squares b-spline curve and surface fitting," *Computer-Aided Design*, vol. 47, pp. 32 – 44, 2014.

[47] C. Taylor, "Finite difference coefficients calculator." [Online]. Available: http://web.media.mit.edu/~crtaylor/calculator.html

[48] A. Nealen, "An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation," 01 2004.

[49] X. Jiao and H. Zha, "Consistent computation of first- and second-order differential quantities for surface meshes," *Proceedings of the 2008 ACM symposium on Solid and physical modeling - SPM '08*, 2008.

[50] A. J. Conley, R. Garcia, D. Kinnison, J.-F. Lamarque, D. Marsh, M. Mills, A. K. Smith, S. Tilmes, F. Vitt, H. Morrison *et al.*, "Description of the ncar community atmosphere model (cam 5.0)," 2012.

[51] C. Garth and K. I. Joy, "Fast, memory-efficient cell location in unstructured grids for visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1541–1550, 2010.

[52] S. Lee, G. Wolberg, and S. Y. Shin, "Scattered data interpolation with multilevel b-splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 228–244, 1997.

[53] Weiqiang Zhang, Zesheng Tang, and Jie Li, "Adaptive hierarchical b-spline surface approximation of large-scale scattered data," in *Proceedings Pacific Graphics '98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No.98EX208)*, 1998, pp. 8–16.

[54] M. Bertram, X. Tricoche, and H. Hagen, "Adaptive smooth scattered data approximation for large-scale terrain visualization," in *VisSym*, 2003.

[55] G. Wahba, *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611970128

[56] J. Duchon, *Splines minimizing rotation-invariant semi-norms in Sobolev spaces*, 11 2006, vol. 57, pp. 85–100.

[57] B. Francis, M. Mathew, and M. Arigovindan, "Image reconstruction from undersampled confocal microscopy data using multiresolution based maximum entropy regularization," *Journal of Instrumentation*, vol. 14, no. 07, pp. P07 015–P07 015, jul 2019.

[58] ——, "Multiresolution-based weighted regularization for denoised image interpolation from scattered samples with application to confocal microscopy," *J. Opt. Soc. Am. A*, vol. 35, no. 10, pp. 1749–1759, Oct 2018.

[59] C. Vazquez, E. Dubois, and J. Konrad, "Reconstruction of irregularly-sampled images by regularization in spline spaces," in *Proceedings. International Conference on Image Processing*, vol. 3, 2002, pp. III–III.

[60] M. Arigovindan, M. Suhling, P. Hunziker, and M. Unser, "Variational image reconstruction from arbitrarily spaced samples: A fast multiresolution spline solution," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 450–460, 2005.

[61] E. Vuçini, T. Möller, and M. E. Gröller, "Efficient reconstruction from non-uniform point sets," *The Visual Computer*, vol. 24, no. 7-9, pp. 555–563, 2008.

[62] ——, "On visualization and reconstruction from non-uniform point sets using b-splines," in *Computer Graphics Forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 1007–1014.

[63] A. Bourquard and M. Unser, "Anisotropic interpolation of sparse generalized image samples," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 459–472, 2013.

[64] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and t-nurccs," *ACM Trans. Graph.*, vol. 22, no. 3, p. 477–484, Jul. 2003.

[65] Yimin Wang and Jianmin Zheng, "Adaptive t-spline surface approximation of triangular meshes," in *2007 6th International Conference on Information, Communications Signal Processing*, 2007, pp. 1–5.

[66] Y. Wang and J. Zheng, "Curvature-guided adaptive t-spline surface fitting," *Computer-Aided Design*, vol. 45, no. 8, pp. 1095 – 1107, 2013.

[67] D. El-Rushaidat, R. Yeh, and X. Tricoche, "Accurate parallel reconstruction of unstructured datasets on rectilinear grids," *Unpublished manuscript*, 2021.

[68] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2012, vol. 3.

VITA

VITA

Raine Yeh received her Ph.D. in computer science from Purdue University working under Prof. Xavier Tricoche in the Computer Graphics and Visualization Lab. Raine Yeh's dissertation work was primarily concerned with signal, mesh, and volumetric data fitting and reconstruction in structured B-splines space. She received her B.S in computer science from the University of Illinois at Urbana-Champaign in 2011 and her M.S. in computer science from Purdue University in 2016.