Statistical Analysis of Industrial Processes using fast Nonparametric Regression Techniques

Master Thesis Submitted to

Prof. Dr. Wolfgang K. Härdle

Prof. Dr. Weining Wang

Ladislaus von Bortkiewicz Chair of Statistics

C.A.S.E.- Centre for Applied Statistics and Economics

Humboldt-Universität zu Berlin

by

Simon Diehl

(541439)

in partial fulfillment of the requirements

for the degree of

Master of Statistics

Berlin, January 30, 2014

Abstract

In this thesis we show how to use nonparametric regression techniques to develop monitoring systems for industrial machinery. Since data sets are often very big and monitoring has to be done online we present two different approaches to speed up computational time of the nonparametric methods. Both approaches are compared regarding run-time and approximation error and their properties are illustrated in a simulation study. We apply our methodology to real data from an industrial supply engineering machine.

Keywords: Nonparametric Regression, Fast Fourier Transform, Fast Gauss Transform, Binning, Energy Consumption, Industrial Machinery

Contents

1 Introduction

Energy costs are a major proportion of the total costs in industrial production processes. Due to the growing scarcity of resources, energy efficient production becomes more and more important. Against this background, in 2011, the International Organization for Standardization (ISO) developed a specification (ISO 50001) for companies to implement an energy management system to improve energy efficiency in their production process.
Although, in Germany it is not compulsory to establish such an energy management system, companies having energy intensive production can benefit from complying with the specification since they are eligible to claim for a reduction of the EEG reallocation charge. In order to fulfill the requirements of ISO 50001, companies must provide information on the improvements made in terms of energy efficiency. This raises the question how to measure energy efficiency in a production process and how to quantify potential- and realized savings. Since almost every sort of machinery is influenced by its environment, the energy consumption varies with a change in the environmental conditions. Therefore one need to control for these environmental conditions in order to achieve three objectives.

1. Estimating potential savings in energy demand due to investment in new machinery
2. Check whether the desired efficiency enhancement is realized after an investment
3. Early detection of malfunctioning that causes higher energy demand than "usual"

The first aim boils down to a comparison of expected values of two different machineries given the same conditions. We would like to make statements of the form "How much energy would the new machinery have consumed under the same conditions in the same time?". Summing up this difference over time allows us to estimate the benefit of an investment in the new machinery. Whereas the first and the second objective are interesting in the long-run, the third one is more important in the short-run. Depending on the sort of machinery, it is likely that the energy demand rises as a consequence of a broken component or a blocked filter. Early notice of such an event is therefore of great importance. To distinguish an unusual raise in energy demand from usual ups and downs, we need to have a threshold to compare with. The focus of this thesis is mainly on the statistical tools needed to develop an early detection warning system as described by objective three.

Industrial production processes and the corresponding supply engineering have been studied extensively from the beginning of the industrialization by engineers and physicists. Nowadays we have a broad theoretical knowledge about the underlying functional principles of different types of processes. In thermodynamics, for instance, Carnot's cycle describes an optimal process to convert thermal energy into work and vice versa. In practice, however, heating and cooling engines do not reach the efficiency of a Carnot's cycle due to frictional losses, heat losses, leakages and inefficient control system engineering. The deviation of the actual energy demand to the theoretical optimum is widely used as a benchmark for the efficiency of an engine (Callen, 2006). Similar benchmarks are defined in Electrical Engineering (Sarma, 2001) and Mechanics (Ugural, 2003). In this work we take a different approach to make statements about the performance of engines. Instead of using an optimal process as a benchmark we compare actual energy input to the input needed in the past in "similar" situations by the same machine. This is advantageous since a theoretical optimum might be hard to calculate for complex engines or even systems of engines. Furthermore, one cannot ensure that a theoretical process model fits to

a real process. To put it differently: Our process in consideration might be influenced by factors that are not covered by theory. We try to circumvent this problem by including explanatory factors while making as little assumptions about the functional principles as possible. A threshold for "usual" energy demand is computed using kernel regression techniques. The application of data-driven techniques is essential in our setting because of the diversity of engines for which we want to establish a warning system. On the downside, however, we trade model flexibility for computational complexity. This is a serious issue since we potentially have a large number of massive data sets to deal with simultaneously. We propose two different approaches that save computing time by introducing an approximation error.

In section two we introduce the kernel regression techniques to estimate the conditional mean and conditional quantiles. In section three we present the two classes of algorithms that can be used to circumvent computational constraints in this framework. We also give theoretical results on runtime and approximation error and illustrate their properties in a simulation study. In section five we apply the proposed methods to real data from supply engineer machineries. Section six concludes.

2 Methodology

As an intuitive benchmark to compare the actual energy demand with, we use the conditional expectation of energy demand given the external circumstances, $\mathrm{E}[Y|X]$. We suppose that the actual energy consumption is close to the expected value if the machinery runs regularly. If the demand is significantly higher than expected this might indicate increased wear on machinery or some other malfunctioning. To estimate the conditional expectation we apply kernel regression techniques since they provide the flexibility we need to deal with lots of different types of machinery without explicitly specifying a statistical model in a parametric form.
The conditional mean gives insights into the "average behavior" of an engine. In our situation, however, we need a broader picture to distinguish regular from irregular behavior. Namely, we are interested in the dispersion of energy consumption given the circumstances. Therefore, we additionally estimate conditional quantiles of the energy consumption nonparametrically to determine upper bounds that can be used in a warning system.

2.1 Kernel Mean Regression

Let $y \in \mathbb{R}$ denote the energy demand of an engine, $x \in \mathbb{R}^d$ denote some explanatory variables and $\varepsilon \in \mathbb{R}$ is an unobservable error term. We assume that the input y depends on the explanatory variables x and the error term ε by the equation:

$$y_i = m(x_i) + \sigma(x_i)\varepsilon_i, \quad i = 1, \ldots, n$$

We further assume that the error terms ε are *iid*. For thermodynamic processes, however, this assumption might be questionable because of there inherent inertia. We circumvent the issue of dealing with time series effects explicitly by taking hourly averages, since the inertia of thermodynamic processes is usually of short persistence, in particular shorter than ten minutes. An explicit treatment of time series effects in this setting is beyond the scope of this work. We refer the reader to Härdle et al. (1997) and Fan (2003) for comprehensive introduction to nonparametric time series models.

2.1.1 Local Constant Estimator

A well established estimator of the unknown function $m(\bullet)$ goes back to the pioneering work by Nadaraya (1964) and Watson (1964) and is known as multivariate local constant estimator:

$$\widehat{m}_H^{\mathrm{LC}}(x) = \frac{\sum_{i=1}^{n} \mathcal{K}_H(x - X_i)\, y_i}{\sum_{i=1}^{n} \mathcal{K}_H(x - X_i)} \tag{1}$$

Where $H \in \mathbb{R}^{d \times d}$ is a symmetric and positive definite bandwidth matrix and

$$\mathcal{K}_H(\bullet) = |H|^{-1} \mathcal{K}\left\{H^{-1} \times (\bullet)\right\}$$

denotes a multivariate Kernel function, $\mathcal{K} : \mathbb{R}^d \to \mathbb{R}$. Equation (1) can be rewritten as:

$$\widehat{m}_H^{\mathrm{LC}}(x) = \frac{1}{n}\sum_{i=1}^{n} W_{Hi}(x) Y_i \tag{2}$$

with

$$W_{Hi}(x) = \frac{\mathcal{K}_H(x - X_i)}{n^{-1}\sum_{j=1}^{n} \mathcal{K}_H(x - X_j)}$$

This representation leads to a very intuitive interpretation of the local constant estimator as a weighted average of the dependent variable. A comprehensive overview of the derivation and the properties of the local constant estimator is given by (Li and Racine, 2007, chapter 3) and (Härdle et al., 2004, chapter 4).
In order to compute the estimator given in equation (1) and (2) we need to choose a Kernel function $\mathcal{K}$ and a bandwidth matrix H. (Wand and Jones, 1995, Chapter 2.7) show that the choice of the kernel function is not crucial in terms of efficiency of the estimation and can be based on other criteria like computational efficiency. Therefore, we choose a Gaussian product kernel of the form:

$$\mathcal{K}_H(u) = \prod_{j=1}^{d} \gamma \exp\left(-u_j^2/h_j^2\right) = \gamma \exp\left(-\sum_{j=1}^{d} u_j^2/h_j^2\right)$$

This choice is convenient for computationally reasons as will be shown in section (3). The constant γ is chosen such that $\mathcal{K}_H(u)$ is in fact a kernel function. That is, it must integrate to one:

$$\int_{\mathbb{R}^d} \gamma \exp\left(\frac{-\|u\|^2}{h^2}\right) du = 1$$

For sake of readability, we skip the constant γ hereinafter. For the local constant estimator it cancels out anyway. By using a product kernel we implicitly assume that the bandwidth matrix H is diagonal:

$$H = \begin{pmatrix} h_1 & & 0 \\ & \ddots & \\ 0 & & h_d \end{pmatrix}$$

In contrast to the choice of the kernel function, the choice of the bandwidths is critical in terms of efficiency of the estimator. We choose the vector of bandwidths $h = (h_1, \ldots, h_d)$ such that the leave-one-out cross-validation criterion is minimized:

$$\begin{aligned} \mathrm{CV}(h) &= \frac{1}{n}\sum_{i=1}^{n} \left\{Y_i - \widehat{m}_{h,-i}(X_i)\right\}^2 w(X_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} \left\{Y_i - \widehat{m}_h(X_i)\right\}^2 \left\{1 - \frac{1}{n} W_{h,i}(X_i)\right\}^{-2} w(X_i) \end{aligned}$$

Where $\widehat{m}_{h,-i}$ denotes the estimator of the conditional mean leaving out the $i-$th observation and $w(X_i)$ is some weight function. A detailed treatment of the cross-validation bandwidth selection can be found in (Härdle et al., 2004, Chapter 4.3)

Though the local constant estimator is appealing for it's catchy interpretation, it suffers from poor performance near the boundaries of the support (Wand and Jones, 1995, Chapter 5.5). Fitting a polynomial of order p locally around a point x, instead of a constant, yields estimators with improved performance at the boundaries. A popular choice of such estimators is the so called local linear estimator which we describe in the following.

2.1.2 Local Linear Estimator

If we assume that the unknown function $m(x)$ is a least one times differentiable in every dimension, it can be approximated by a multivariate Taylor Series in the neighborhood of some point x_0 by:

$$m(x) \approx m(x_0) + \nabla m(x_0)^\top (x - x_0)$$

with $\nabla m(x_0)$ being the gradient of m evaluated at x_0. Stone (1977) and Cleveland (1979) show that under these conditions the local linear estimator can be derived by the solution of the weighted least squares minimization problem:

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} \left\{ Y_i - \beta_0 - \beta_1^\top (X_i - x) \right\}^2 \mathcal{K}_h(X_i - x) \tag{3}$$

Minimizing problem (3) w.r.t to β_0 and β_1 gives us two quantities of interest. The first term $\widehat{\beta}_0$ is an estimator of the value of function m at the point x. Namely

$$\widehat{m}_h^{\text{LL}}(x) = \widehat{\beta}_0(x)$$

Whereas $\widehat{\beta}_1$ is an estimator of all partial derivatives of m w.r.t to x. For sake of compactness we do not show the statistical properties of the local linear estimator, e.g. the asymptotic behavior, but direct the reader to the comprehensive works of Wand and Jones (1995) and Fan and Gijbels (1996). To express a general local polynomial estimator we make use of the following notation:

$$S_{h,j}(x) = \sum_{i=1}^{n} \mathcal{K}_h(x - X_i)(X_i - x)^j \tag{4}$$

$$T_{h,j}(x) = \sum_{i=1}^{n} \mathcal{K}_h(x - X_i)(X_i - x)^j Y_i \tag{5}$$

Where j denotes a $d-$dimensional multi-index (Masry, 1996). Using this notation, a general local polynomial estimator of order p can be expressed as:

$$\widehat{\beta}(x) = \begin{pmatrix} S_{h,0}(x) & \dots & S_{h,p}(x) \\ \vdots & \ddots & \vdots \\ S_{h,p}(x) & \dots & S_{h,2p}(x) \end{pmatrix}^{-1} \begin{pmatrix} T_{h,0}(x) \\ \vdots \\ T_{h,p}(x) \end{pmatrix}$$

For $p = 0$ we get the local constant estimator, which can be formulated as:

$$\widehat{m}_h^{\text{LC}}(x) = \frac{T_{h,0}(x)}{S_{h,0}(x)}$$

The representation given by equation (4) and (5) is particular useful since it reveals that local polynomial estimators are in fact based on matrix-vector products. For example, the one-dimensional local constant estimator at different target points $x_j,\ j = 1,\dots,m$, involves the computation of the following matrix-vector products:

$$\begin{pmatrix} T_{h,0}(x_1) & S_{h,0}(x_1) \\ \vdots & \vdots \\ T_{h,0}(x_m) & S_{h,0}(x_m) \end{pmatrix} = \begin{pmatrix} \mathcal{K}_h(x_1 - X_1) & \dots & \mathcal{K}_h(x_1 - X_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}_h(x_m - X_1) & \dots & \mathcal{K}_h(x_m - X_n) \end{pmatrix} \begin{pmatrix} y_1 & 1 \\ \vdots & \vdots \\ y_n & 1 \end{pmatrix}$$

Similar expressions can be formulated for higher order local polynomial estimators. In the following we use a general short hand notation to indicate such estimators:

$$\begin{pmatrix} G(x_1) \\ \vdots \\ G(x_m) \end{pmatrix} = \begin{pmatrix} \mathcal{K}_h(x_1 - X_1) & \dots & \mathcal{K}_h(x_1 - X_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}_h(x_m - X_1) & \dots & \mathcal{K}_h(x_m - X_n) \end{pmatrix} \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix}$$

The choice of ν depends on the estimator we want to calculate. We intentionally let ν be unspecified to point out the general structure of local polynomial regression problems.

So far, we have introduced Kernel regression techniques to learn from the data how an engine behaves "on average" under regular conditions. A deviation from this behavior is read as an indicator for some malfunctioning. The question arises how to distinguish normal fluctuations around the expected value from unusual deviations caused by deterioration or defects. The method of choice to cope with that is conditional quantile regression, which yields a measure of the regular dispersion.

2.2 Kernel Quantile Regression

Conditional Quantile Regression techniques became more and more popular in the recent years since the groundbreaking work of Koenker and Bassett Jr (1978). Working with conditional quantiles is appealing since they provide a broader picture of an unknown distribution of interest. The conditional quantile at the level of τ of a CDF F is given by:

$$q_\tau(x) = \inf\{y : F(y|x) \geq \tau\} = F^{-1}(\tau|x)$$

with $\tau \in (0,1)$. cf. (Li and Racine, 2007, chapter 6.3)
A widely used (global) estimator of conditional quantiles is given by the minimizer of the weighted sum of absolute distances:

$$\min_{\mu} \sum_{i=1}^{n} \rho_\tau (Y_i - \mu) \tag{6}$$

with $\rho_\tau(u) = u[\tau - \mathbb{1}(u \leq 0)]$ is called check function and μ is the conditional quantile.
The nonparametric kernel estimator of a conditional quantile is a locally weighted version of

equation (6). The local constant and the local linear quantile estimates are given by:

$$\min_{\beta_0} \sum_{i=1}^{n} \rho_\tau (Y_i - \beta_0) \mathcal{K}_h(X_i - x) \tag{7}$$

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} \rho_\tau \left\{ Y_i - \beta_0 - \beta_1^\top (X_i - x) \right\} \mathcal{K}_h(X_i - x) \tag{8}$$

respectively.
As in the conditional mean regression, the degree of smoothing in every dimension has to be chosen. To do so, we apply the rule-of-thumb bandwidth introduced by Yu and Jones (1998):

$$h_{\alpha,j} = h_j \left[\frac{\alpha(1-\alpha)}{\phi \left\{ \Phi^{-1}(\alpha) \right\}^2} \right]^{1/5}$$

Where h_j denotes the mean regression bandwidth in dimension j, $\phi(\bullet)$ and $\Phi^{-1}(\bullet)$ are the PDF and the inverse of the CDF of a standard normal distribution, respectively.

3 Fast Algorithms

In section 2.1 we have shown that various kernel estimation problems boil down to the calculation of matrix-vector products of the following form:

$$\begin{pmatrix} G(x_1) \\ \vdots \\ G(x_m) \end{pmatrix} = \underbrace{\begin{pmatrix} \mathcal{K}_h(x_1 - X_1) & \dots & \mathcal{K}_h(x_1 - X_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}_h(x_m - X_1) & \dots & \mathcal{K}_h(x_m - X_n) \end{pmatrix}}_{\Psi} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{9}$$

Where X denote points from a reference period that we use to compare with the actual energy demand of an engine. We call these points "sources". The points from the review period, denoted by x, are called "targets". A naïve (and exact) calculation of this product has runtime $\mathcal{O}(m \times n \times d)$. Hence, even for a moderate number of source points s and target points t kernel techniques become computationally expensive. For large sample sizes, however, a direct evaluation of (9) is impractical for realtime analysis. Considerable speedups in evaluating Matrix-Vector products of this form are attainable if the matrix Ψ reveals some suitable structure. That is, there is some connection between the elements of Ψ that can be exploited. If Ψ does not come with any suitable structure, we can impose it by appropriate preprocessing, at the expense of loosing precision.

In the following we present two classes of algorithms that achieve subquadratic runtime for calculating Matrix-Vector products of this type. First, we show how the Fast Fourier Transform (FFT) can be utilized to evaluate equation (9) in $\mathcal{O}(n \log n)$ time if the source points and the target points are located on regular grids. Since data is usually not located on regular grids, we present binning strategies to impose a regular grid structure. Secondly, we present a class of algorithms, called Fast Multipole Method (FMM). It gains computational speed by element wise expansion of the matrix Ψ and block wise factorization to "encapsulate information" that can be reused to save computations. Although there are numerous variations of the FMM available depending on the shape of the Kernel function $\mathcal{K}(\bullet)$ and the method of expansion we focus on the case of a Gaussian Kernel since it is the most relevant case in our situation. Particular attention is given to the Fast Gauss Transform introduced by Greengard and Strain (1991) and a modified version, the Improved Fast Gauss Transform, developed by Raykar et al. (2005).

Both, the (FMM) and the (FFT), were honored to belong to the "top ten algorithms of the 20th century" (Cipra, 2000). Nevertheless, they also have some shortcomings in particular situations. We address these shortcomings as well as the advantages of both methods in terms of speed and accuracy. We especially focus on the performance in higher dimensions.

3.1 Fast Fourier Transform and Binning Procedures

The application of the Fast Fourier Transform to speed-up nonparametric estimation procedures goes back to the groundbreaking work by Silverman (1982). He suggested a two-step procedure to accelerate univariate Kernel Density estimation. In the first step, the raw data is mapped to a regular grid with M number of grid points ($M \ll n$). This enormously simplifies the data structure and makes the problem almost independent of the original sample size. In the second step, the simplified data structure can be exploited be the means of the Fast Fourier Transform. Härdle (1987) applies this idea to univariate Kernel smoothing. Fan and Marron (1994) give a refined mapping procedure (so called linear binning) that shows better performance in terms of bias than the formerly used constant binning. They also present an application using a local linear estimator. Wand (1994) extends the work of Fan and Marron (1994) to multivariate settings. In the following we present this strategy in greater detail. Therefore we first briefly describe the mapping procedure used. Then we show how our Matrix-Vector multiplication problem is connected to a circular convolution setting and how the FFT can be utilized to speed up calculations. Particular attention is paid to the runtime in higher dimensions. Finally we emphasize the approximation error caused by the binning.

3.1.1 Binning Procedures

Let $x_i, y_i \in \mathbb{R}, i = 1, \ldots, n$ be a sample of exogenous and endogenous variables in a regression framework, respectively. Let further $g_j \in \mathbb{R}, j = 1, \ldots, M$ be equally spaced grid points. Moreover, let $\eta(x_i, g_j)$ denote a function that assigns weight to each combination of data points x_i and grid points g_j. The total weight of a grid point g_j is called "bin count" and defined as:

$$c_j = \sum_{i=1}^{n} \eta(x_i, g_j)$$

In the same way we can summarize the endogenous variable at the grid points as a weighted average of those y_i for which the corresponding x_i put weight to the grid point g_j:

$$\bar{y}_j = c_j^{-1} \sum_{i=1}^{n} \eta(x_i, g_j) y_i$$

A straightforward choice of the weight function $\eta(\bullet)$ is an indicator function that assigns unit weight to grid point g_j if it is nearest to data point x_i and zero weight to all other grid points:

$$\eta(x_i, g_j)^{CB} = \begin{cases} 1 & \text{if } |x_i - g_j| < |x_i - g_k| \; \forall j \neq k \\ 0 & \text{else} \end{cases}$$

This approach is called Constant Binning since the weight from all data points x for which g is the nearest grid point is constant regardless how far these points are away from g . Figure (1) shows a mapping of 20 data points to a grid with 7 grid points using Constant Binning. A line indicates that grid point g_j receives (unit) weight from data point x_i.
In contrast to constant binning the Linear Binning approach takes into account how far a data point is away from a grid point. The smaller the distance between grid point and data point, the higher the weight that is assigned to the grid point. Figure (2) illustrates that idea. Again,

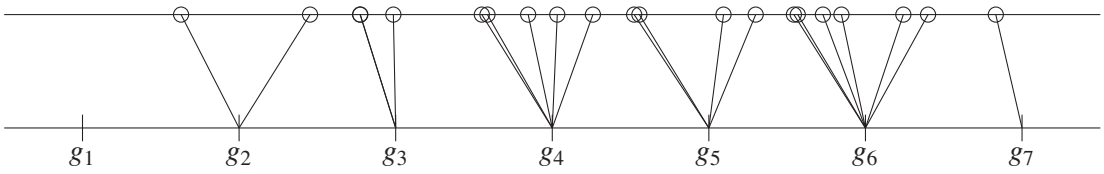

Figure 1: Mapping data points to grid using constant binning

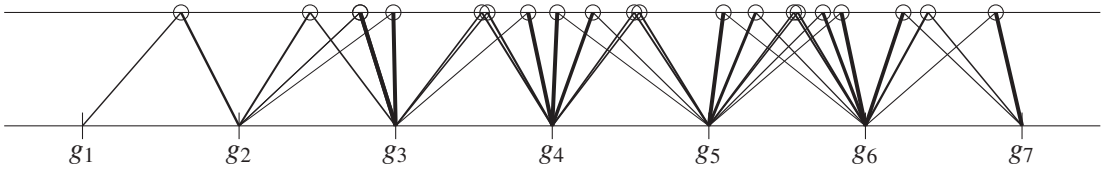

Figure 2: Mapping data points to grid using linear binning

20 data points are mapped to a regular grid with 7 grid points but using linear binning. The presence of a line between a data point and a grid point indicates that we assign weight from the data point to the grid point. The thickness of the line indicates the strength of the assignment. Fan and Marron (1994) suggested the following weight function for the linear binning case:

$$\eta(x_i, g_j)^{LB} = \left(1 - \frac{|x_i - g_j|}{\Delta}\right)_+$$

Where Δ is the distance between two consecutive grid points, that is $\Delta = (g_M - g_1)/(M-1)$. The authors also give an algorithm to calculate the constant and linear weights c_j in $\mathcal{O}(n)$ time. For sake of compactness we skip a detailed description of the procedure and direct the reader to the original paper.

Applying one of the above mentioned binning techniques leads to remarkable computational savings for the evaluation of (9). The number of computations reduces from $\mathcal{O}(n^2)$ to $\mathcal{O}(M^2+n)$ in the binned version. The binned local constant Estimator, for instance, is given by:

$$\widehat{m}(g_j) = \frac{\bar{T}_0(g_j)}{\bar{S}_0(g_j)} \tag{10}$$

with

$$\begin{pmatrix} \bar{T}_0(g_1) & \bar{S}_0(g_1) \\ \vdots & \vdots \\ \bar{T}_0(g_M) & \bar{S}_0(g_M) \end{pmatrix} = \underbrace{\begin{pmatrix} \mathcal{K}_h(g_1 - g_1) & \dots & \mathcal{K}_h(g_1 - g_M) \\ \vdots & \ddots & \vdots \\ \mathcal{K}_h(g_M - g_1) & \dots & \mathcal{K}_h(g_M - g_M) \end{pmatrix}}_{\bar{\Psi}} \times \begin{pmatrix} c_1\bar{y}_1 & c_1 \\ \vdots & \vdots \\ c_M\bar{y}_M & c_M \end{pmatrix}$$

To obtain estimates at points not lying on the grid one can make use of interpolation techniques as described by (Press, 2007, Chapter 3).

Although binning reduces the computational cost quite substantially, even further improvements are possible if we consider that in fact the binned matrix-vector product is a discrete convolution. Conveniently, discrete convolutions can be computed fast using the FFT, which is presented in the following section.

3.1.2 Convolution and the Fast Fourier Transform

Let $\{u_k\}$ and $\{w_k\}$ be two univariate sequences with period n, i.e. $u_k = u_{k+\gamma n},\ \forall \gamma \in \mathbb{N}$. The discrete convolution of these sequences is then defined by:

$$f_k = u * v = \sum_{i=0}^{n-1} w_i u_{k-i}$$

Van Loan (1992) shows that this is in fact equivalent to the matrix-vector product:

$$f = C_n(u)w \tag{11}$$

Where $C_n(u)$ denotes a circulant matrix. A circulant matrix is characterized by the fact that every column is a down-shifted version of its predecessor. To be specific, a circulant matrix has the following form:

$$C_n(u) = \begin{pmatrix} u_0 & u_{n-1} & \dots & u_2 & u_1 \\ u_1 & u_0 & u_{n-1} & & u_2 \\ \vdots & u_1 & u_0 & \ddots & \vdots \\ u_{n-2} & & \ddots & \ddots & u_{n-1} \\ u_{n-1} & u_{n-2} & \dots & u_1 & u_0 \end{pmatrix}$$

As pointed out by (Golub and Van Loan, 2012, Chapter 4.8), a particularly useful characteristic of the matrix $C_n(u)$ is that it can be expressed as a polynomial in the downshift-operator $\mathcal{D}_n$:

$$C_n(u) = u_0 D_n^0 + u_1 \mathcal{D}_n^1 + \ldots + u_{n-1}\mathcal{D}_n^{n-1} = \sum_{k=0}^{n-1} u_k \mathcal{D}_n^k$$

We make use of the Matlab index notation to define the downshift-operator $\mathcal{D}_n$ as follows:

$$\mathcal{D}_n = I_n(:, [2:n, 1])$$

In case that $n = 4$, for example, it is given as:

$$\mathcal{D}_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

An exploitable property of the downshift-operator $\mathcal{D}_n$ is the fact that it can be diagonalized as follows:

$$\mathcal{F}_n \mathcal{D}_n \mathcal{F}_n^{-1} = \Lambda_n,$$

where $\mathcal{F}_n$ is the Discrete Fourier Transform (DFT) matrix with elements:

$$[\mathcal{F}_n]_{pq} = \omega_n^{(p-1)(q-1)} = \exp\{-2\pi(p-1)(q-1)i/n\}, \quad i^2 = -1$$

Λ_n is a diagonal matrix of the following form:

$$\Lambda_n = \begin{pmatrix} 1 & & & \\ & \omega_n & & \\ & & \ddots & \\ & & & \omega_n^{n-1} \end{pmatrix}$$

Since the circulant matrix $C_n(u)$ is a polynomial in the downshift-operator $\mathcal{D}_n$ and $\mathcal{D}_n$ can be diagonalized by $\mathcal{F}_n$ it follows that $\mathcal{F}_n$ also diagonalizes the circulant matrix $C_n(u)$:

$$\mathcal{F}_n \mathcal{C}_n(u) \mathcal{F}_n^{-1} = \sum_{k=0}^{n-1} u_k \mathcal{F}_n \mathcal{D}_n^k \mathcal{F}_n^{-1} = \sum_{k=0}^{n-1} u_k \big(\underbrace{\mathcal{F}_n \mathcal{D}_n \mathcal{F}_n^{-1}}_{\Lambda_n} \big)^k = \sum_{k=0}^{n-1} u_k \Lambda_n^k = diag(\mathcal{F}_n u)$$

In conjunction with equation (11) this leads to the particular useful result:

$$f = u * w = C_u(u) w = \mathcal{F}_n^{-1} \left\{ (\mathcal{F}_n u) \odot (\mathcal{F}_n w) \right\} \tag{12}$$

This is in fact nothing else than the well-known discrete circular convolution theorem. For a detailed proof we refer the reader to Hunt (1971). A concise introduction is given by (Van Loan, 1992, Chapter 4.2)

At a first glance we have not gained any improvement in terms of numerical performance by introducing the DFT factorization. Since the matrix $\mathcal{F}_n$ is of the same size as $C(u)_n$ it is quite the opposite. However, closer examination of the matrix $\mathcal{F}_n$ shows that it reveals a special structure that can be exploited to speed up the calculation of equation (12). This is exactly what is done by a huge class of algorithms called Fast Fourier Transforms (FFT).
To illustrate the idea behind the FFT let $n = 2^3 = 8$. Further, let $x \in \mathbb{C}$ be a vector of length n. The discrete Fourier transform is then given by:

$$\mathcal{F}_8 x = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega 7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix}$$

Since $\omega = \omega_8 = \exp\{-2\pi i/8\}$ is a primitive $n-th$ root of unity the higher exponents simplify. If we group the columns with even and odd indices we can see how the DFT of length 8 is related to a DFT of length 4. Applying this recursively is the core of the Radix-2 Factorization algorithm invented by Cooley and Tukey (1965). Reordering the columns of $\mathcal{F}$ is achieved by

post-multiplying by the permutation matrix $P = I_n(:, [2:2:n, 1:2:n-1])$. For $n = 8$ this is:

$$P = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\text{odd-cols}} \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{even-cols}}$$

This yields:

$$\mathcal{F}_8 x = \mathcal{F}_8 P P^\top x = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega & \omega^3 & \omega^5 & \omega^7 \\ 1 & \omega^4 & 1 & \omega^4 & \omega^2 & \omega^6 & \omega^2 & \omega^6 \\ 1 & \omega^6 & \omega^4 & \omega^2 & \omega^3 & \omega & \omega^7 & \omega^5 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & \omega^2 & \omega^4 & \omega^6 & -\omega & -\omega^3 & -\omega^5 & -\omega^7 \\ 1 & \omega^4 & 1 & \omega^4 & -\omega^2 & -\omega^6 & -\omega^2 & -\omega^6 \\ 1 & \omega^6 & \omega^4 & \omega^2 & -\omega^3 & -\omega & -\omega^7 & -\omega^5 \end{array}\right) \left(\begin{array}{c} x_1 \\ x_3 \\ x_5 \\ x_7 \\ \hline x_2 \\ x_4 \\ x_6 \\ x_8 \end{array}\right)$$

By applying this reordering the centerpiece of the FFT becomes clear, since:

$$\mathcal{F}_8 x = \left(\begin{array}{c|c} \mathcal{F}_4 & \Omega_4 \mathcal{F}_4 \\ \hline \mathcal{F}_4 & -\Omega_4 \mathcal{F}_4 \end{array}\right) P^\top x = \left(\begin{array}{c|c} \mathcal{I}_4 & \Omega_4 \mathcal{I}_4 \\ \hline \mathcal{I}_4 & -\Omega_4 \mathcal{I}_4 \end{array}\right) \left(I_2 \otimes \mathcal{F}_4\right) P^\top x, \quad \Omega_4 = \begin{pmatrix} 1 & & & \\ & \omega & & \\ & & \omega^2 & \\ & & & \omega^3 \end{pmatrix}$$

This result holds in general as:

$$\mathcal{F}_n x = \left(\begin{array}{c|c} \mathcal{I}_m & \Omega_m \mathcal{I}_m \\ \hline \mathcal{I}_m & -\Omega_m \mathcal{I}_m \end{array}\right) \left(I_2 \otimes \mathcal{F}_m\right) P^\top x, \qquad \forall n = 2^k, k \in \mathbb{N}, k \geq 1, m = n/2$$

Now we can apply the same kind of factorization to $\mathcal{F}_m$ and so on until we end at $\mathcal{F}_1$. For the inverse Fourier Transform, $\mathcal{F}_n^{-1} x$, the idea of splitting applies analogously. In this way we reduce the amount of work for the discrete convolution (12) from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$.

Besides this recursive procedure, there are many more FFT algorithms. A non-recursive version of the Radix-2 splitting was invented by Cooley and Tukey (1965). For the special case that $n = 4^k$ or $n = 8^k, k \in \mathbb{N}$ there also exist optimized Radix-4 and Radix-8 factorization, respectively (Duhamel and Hollmann, 1984). Rader (1968) and Kolba and Parks (1977) give algorithms for cases where n is prime. A generalization to $n = p^k, k \in \mathbb{N}, p$ prime was developed by Winograd (1978). A variant of the FFT that allows for arbitrary n is given by Bluestein (1970). A profound overview of the different modifications of the FFT can be found in Duhamel and Vetterli (1990) and Van Loan (1992). Recent work on fast DFT for data that is sparse in the frequency domain has been done by Hassanieh et al. (2012a) and Hassanieh et al. (2012b). In addition to the computational aspects, there is also a huge strand of literature on the theory of the Fourier Transform. A comprehensive introduction is given by Brockwell (1986), Stein and Shakarchi (2003) and Wong (2011), to name only a few.

We have seen that a Matrix-Vector product can be calculated in $\mathcal{O}(n \log n)$ computations using the FFT if the matrix is circulant. To utilize this result we must show that the matrix of kernel weights for the binned data $\widetilde{\Psi}$ can easily be transformed to a circulant matrix. Therefore, we use the fact that the matrix $\widetilde{\Psi}$ has a Toeplitz structure. A $(n \times n)$ matrix $\mathtt{T}$ is called Toeplitz if the entries are constant along each diagonal. For example:

$$\mathtt{T} = \begin{pmatrix} u_0 & u_{-1} & u_{-2} \\ u_1 & u_0 & u_{-1} \\ u_2 & u_1 & u_0 \end{pmatrix}$$

is Toeplitz. Since the grid points g are equally spaced, the distances between them only depend on the difference in their indices: $g_i - g_j = \Delta(i-j), \forall i, j$. The distance between the indices i and j is constant along each diagonal. Hence, the matrix Ψ is indeed Toeplitz. One can derive a Circulant matrix $\mathtt{T}^*$ from the Toeplitz matrix $\mathtt{T}$ by imposing a circularity structure in the following way:

$$\mathtt{T}^* = \begin{pmatrix} u_0 & u_{-1} & u_{-2} & 0 & 0 & 0 & u_2 & u_1 \\ u_1 & u_0 & u_{-1} & u_{-2} & 0 & 0 & 0 & u_2 \\ u_2 & u_1 & u_0 & u_{-1} & u_{-2} & 0 & 0 & 0 \\ 0 & u_2 & u_1 & u_0 & u_{-1} & u_{-2} & 0 & 0 \\ 0 & 0 & u_2 & u_1 & u_0 & u_{-1} & u_{-2} & 0 \\ 0 & 0 & 0 & u_2 & u_1 & u_0 & u_{-1} & u_{-2} \\ u_{-2} & 0 & 0 & 0 & u_2 & u_1 & u_0 & u_{-1} \\ u_{-1} & u_{-2} & 0 & 0 & 0 & u_2 & u_1 & u_0 \end{pmatrix}$$

Keeping in mind that multiplying $\mathtt{T}^*$ with a vector is in fact a circular convolution, the augmentation is needed to avoid wrap-around effects. A general construction procedure is given by (Van Loan, 1992, chapter 4.2.4). For a comprehensive study of the linkages between Toeplitz matrices and Circulant matrices we refer the reader to Gray (2006). The vector that is multiplied by the matrix $\mathtt{T}^*$ is also padded with zeros to match dimensions
Now, we have everything at hand what we need to apply the FFT on the binned version of equation (9). We use the following shorthand notation:

$$\mathcal{K}_h(g_i - g_j) = \mathcal{K}_h\{(i-j)\Delta\} = \mathcal{K}_{|i-j|}$$

to define:

$$\begin{aligned} \psi &= \begin{pmatrix} K_0 & \dots & K_{M-1} & \mathbf{0} & K_{M-1} & \dots & K_1 \end{pmatrix}^\top \\ c^* &= \begin{pmatrix} c^\top & \mathbf{0} \end{pmatrix}^\top \\ c\bar{y}^* &= \begin{pmatrix} c\bar{y}^\top & \mathbf{0} \end{pmatrix}^\top \end{aligned}$$

In this way we can evaluate the local constant estimator in equation (10) as follows:

$$\begin{aligned} \bar{\mathbf{T}}_0 &= \mathcal{F}^{-1}\left\{\mathcal{F}(\psi) \odot \mathcal{F}(c\bar{y}^*)\right\}_{(1:M)} \\ \bar{\mathbf{S}}_0 &= \mathcal{F}^{-1}\left\{\mathcal{F}(\psi) \odot \mathcal{F}(c^*)\right\}_{(1:M)} \end{aligned}$$

The index $(1:M)$ means, that we take only the first M values.
We see that by exploiting the special structure of the matrix $\widetilde{\Psi}$, we further reduce the run-time complexity of the matrix-vector product in equation (9) from $\mathcal{O}(n+M^2)$ to $\mathcal{O}(n+m\log(m))$ where m, $m \geq 2M-1$, is the length of the zero padded vectors $c*$ and $c\bar{y}^*$. To benefit most from the FFT, m should be chosen such that it is the smallest integer greater or equal $2M-1$ that is a power of 2. Namely: $m = 2^k$ with $k = \lceil \log(2M-1)/\log(2) \rceil$.

3.1.3 Moving to higher Dimensions

The extension of this approach to higher dimension is straightforward. A common way to compute the grid counts and the kernel weights in higher dimensions is to take the product of the univariate estimates as suggested by Wand (1994). This leads to a multivariate array of kernel weights $\widetilde{\Psi}$ that has a Toeplitz structure within every slice. For this we can also impose a circularity structure by extending every slice in the same way as we did in the univariate case. As in the univariate case, we can utilize the (multivariate) FFT to speed up calculations. We refer the reader to Chan (1988) and Chan and Olkin (1994) for a detailed overview.
Although extending this approach to higher dimensions is straightforward, the benefit is questionable when working with $d \geq 3$. The problem in higher dimension is that the binning procedure is a fairly rigid type of data clustering. Thus, we implicitly assume that the data is roughly uniformly distributed. In higher dimensions, however, this is rather unlikely. If the data is not uniformly distributed the grid has to be fine enough to keep the approximation error small. This naturally causes problems in terms of memory consumption. To illustrate that, we assume that $d = 3$. If we create a grid with 250 grid points in each dimension, we come up with $250^3 = 15,625,000$ grid points in total where most of them receive zero weight. To avoid wrap-around effects in the convolution step we additionally have to pad the grid arrays with zeros. Doing this, we generate an array with $512^3 = 134,217,728$ entries. Working with double-precision values this array has approximately a size of $1GB$. Since we have to store at least 2 such arrays in memory, this is quite restrictive and exactly the opposite of what we wanted to achieve by using binning. Storing a four-dimensional array with 512^4 entries requires approximately $550GB$ in memory and is prohibitively expensive.
Nevertheless, since most of the entries are zeros, one can make use of data structures for sparse data. Then, only the non-zeros values are stored along with their indices. This reduces the memory consumption dramatically if the share of non-zero values is relatively small. Unfortunately, until now standard Software Packages like Matlab or R do not support multidimensional FFT on sparse data structures.
Moreover, the runtime complexity of the multidimensional FFT grows exponentially with the dimension d. If we use the same number of grid points M, in every dimension the runtime of a $d-$dimensional convolution is:

$$\mathcal{O}(m^d d \log m) \tag{13}$$

with $m = 2^k, k = \lceil \log(2M-1)/\log(2) \rceil$. Hence, even for huge sample sizes and a moderate number of grid points in every dimension, the amount of work to evaluate equation (9) using the FFT approach exceeds the direct evaluation clearly when dealing with higher dimensional data. In a nutshell, one can say that this approach suffers from the sparsity of high dimensional data due to its rigid clustering scheme.

3.1.4 Approximation error

Mapping continuous data onto a regular grid introduces an approximation error. Using a rough grid, we obtain large numerical improvements but we run the risk of a substantial error and vice versa. This raises the question how to trade accuracy for speed. Namely, how to determine the binning rule and the number of grid points.
To answer that question, a pioneering study is given by Hall (1982) who investigated the impact of rounding errors on univariate density and density-derivative estimation. Scott and Sheather (1985) extend this work and give an explicit formula for the Integrated Mean Squared Error (IMSE) of a univariate kernel density estimator as a function of the bin width Δ. Along the same lines, Jones (1989) give Asymptotic results for $n, M \to \infty$. Hall and Wand (1996) examine the influence of the binning rule and the smoothness of the true density on the approximation error of a binned kernel density estimator. They show that linear binning is asymptotically superior to constant binning in terms of the Mean Squared Error (MSE) between a binned and a direct univariate kernel density estimator. Let $\widehat{f}(x)$ and $\widetilde{f}(x)$ be a direct estimator for a univariate density and a binned estimator, respectively. The MSE is then given as:

$$\mathrm{E}\{\widetilde{f}(x) - \widehat{f}(x)\} = \begin{cases} \Delta^2 \alpha_1 \, \mathrm{E}\{\mathcal{K}_h'(x-X)^2\} + \mathcal{O}\big(\Delta^2\big) & \text{Const. Binning} \\ \Delta^4 \left[\alpha_2 \, \mathrm{E}\{\mathcal{K}_h''(x-X)^2\} + \alpha_3 \{\mathrm{E}\mathcal{K}_h''(x-X)\}^2\right] + \mathcal{O}\big(\Delta^4\big) & \text{Lin. Binning} \end{cases} \tag{14}$$

With $\alpha_1 = (12n)^{-1}$, $\alpha_2 = (120n)^{-1}$, $\alpha_3 = (1-n^{-1})/144$ and $\Delta \to 0$. This result also generalizes to higher dimensions and other kernel estimators like local constant and local linear regression estimators (Wand, 1994). Since the runtime complexity for both, constant and linear binning, is $\mathcal{O}(n)$, the latter method is preferable in practice.
Determining the minimum number of grid points that guarantees a given error bound is much more challenging in practice. If the true function of interest is very smooth, only a few grid points suffice to get close estimates. For a very wiggly function however, a very fine grid is needed to keep the approximation error small. Hence, the optimal number of grid points for which the binned estimator satisfies some upper error bound depends on the shape of the function we want to estimate. To our knowledge, there is no straightforward way to select the optimal number of grid points in advance for a certain error bound without any prior information about the data. Doing some pre-analysis to estimate an optimal M could heavily reduce the overall computational benefit.
As a rule-of-thumb, Hall and Wand (1996) suggest to set $100 \leq M \leq 500$ to get good approximations for a wide range of estimation problems. Their results are based on binned density estimation of 15 different types of densities described by Marron and Wand (1992). In univariate local polynomial regression Fan and Marron (1994) recommend to use $400 \leq M$. In higher dimensions, however, this choice might be impractical due to memory restrictions.

3.2 Fast Gauss Transform

Applying the FFT on pre-binned data to make use of the convolution theorem (12) may lead to large computational savings if the distribution of the data is not far from an uniform. For fairly clustered data, however, this approach performs poorly, since a large number of grid points is needed to maintain an acceptable approximation error. This is particularly apparent in higher

dimensions. A better performance might be achieved by binning the data on an irregular grid and make use of a nonuniform FFT algorithm (cf. Dutt and Rokhlin (1993), Greengard and Lee (2004) and Wefers and Vorländer (2012)).
The algorithm we present in the following seizes this idea. The so called Fast Multipole Method is originated in computational physics and was invented to speed up evaluation of all pairwise interactions of particles in a gravitational or Coulomb field (Greengard and Rokhlin, 1987). Instead of simplifying the data structure via binning, it simplifies the kernel function by series expansion around some cluster centers. A vast variety of implementations exist depending on the kernel function and the clustering procedure. Greengard et al. (1998) and Gumerov and Duraiswami (2005) apply this approach to Helmholtz equations. Greengard and Rokhlin (1997) deal with the Laplace equations whereas Coifman et al. (1993) show an application using wave equations. A concise introduction with references to several applications is given by Beatson and Greengard (1997).
In this application we restrict ourselves to the special case of a Gaussian Kernel function leading to the so called Fast Gauss Transform (FGT) invented by Greengard and Strain (1991). Particular attention is paid to a refined modification of the original FGT invented by Yang et al. (2003) and Raykar et al. (2005) that shows superior performance in higher dimensions. Therefore, they call it Improved Fast Gauss Transform (IFGT)

3.2.1 Introductory Example

As a starting point we assume that the Kernel function in use is degenerated. That means it can be written in the following form:

$$\mathcal{K}(x,X) = \sum_{k=1}^{p} \Phi_k(x)\Theta_k(X)$$

Where the two functions $\Phi(x)$ and $\Theta(X)$, depending only on the source points X and the target points x, respectively. The matrix of kernel weights Ψ in equation (9) can then be decomposed as follows:

$$\begin{pmatrix} \mathcal{K}(x_1,X_1) & \dots & \mathcal{K}(x_1,X_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_m,X_n) & \dots & \mathcal{K}(x_m,X_n) \end{pmatrix} = \underbrace{\begin{pmatrix} \Phi_1(x_1) & \dots & \Phi_p(x_1) \\ \vdots & \ddots & \vdots \\ \Phi_1(x_m) & \dots & \Phi_p(x_m) \end{pmatrix}}_{\Phi} \times \underbrace{\begin{pmatrix} \Theta_1(X_1) & \dots & \Theta_1(X_n) \\ \vdots & \ddots & \vdots \\ \Theta_p(X_1) & \dots & \Theta_p(X_n) \end{pmatrix}}_{\Theta} \tag{15}$$

Replacing Ψ in equation (9) by the product $\Phi \times \Theta$ gives:

$$\begin{pmatrix} G(x_1) \\ \vdots \\ G(x_m) \end{pmatrix} = \underbrace{\begin{pmatrix} \Phi_1(x_1) & \dots & \Phi_p(x_1) \\ \vdots & \ddots & \vdots \\ \Phi_1(x_m) & \dots & \Phi_p(x_m) \end{pmatrix} \times \overbrace{\begin{pmatrix} \Theta_1(X_1) & \dots & \Theta_1(X_n) \\ \vdots & \ddots & \vdots \\ \Theta_p(X_1) & \dots & \Theta_p(X_n) \end{pmatrix}}^{(*)} \times \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix}}_{(**)}$$

Evaluating the second part of the RHS, denoted by $(*)$, has runtime complexity of $\mathcal{O}(np)$ and yields a $(p \times 1)$ vector. Taking the product of Φ and this vector has a runtime of $\mathcal{O}(mp)$. Altogether, we can evaluate $(**)$ in $\mathcal{O}(p(n+m))$ time without any approximation error. If $p \ll \min(n,m)$ this is advantageous compared to direct evaluation.

3.2.2 General Case

Standard kernel functions are typically non-degenerated. Hence, this approach is not applicable. Nevertheless, an approximative solution is still feasible. The core of FMM-like algorithms is using a series expansion of the Kernel function that converges relatively fast and truncate it after few terms:

$$\mathcal{K}_h(x,X) \approx \sum_{k=1}^{p} \Phi_k(x,c)\Theta_k(X,c)$$

The point c denotes the expansion center where the information about the source points is to be "encapsulated". This yields:

$$\begin{aligned} G(x) &\approx \sum_{i=1}^{n}\sum_{k=1}^{p} \Phi_k(x,c)\Theta_k(X_i,c)\nu_i \\ &= \sum_{k=1}^{p} \Phi_k(x,c) \sum_{i=1}^{n} \Theta_k(X_i,c)\nu_i \end{aligned}$$

Figure (3) illustrates this approach. The upper panel shows the interactions between the sources (black points) and the targets (red points) in a direct calculation. The amount of work to evaluate the influence of the source points on an additional target point is $\mathcal{O}(n)$. The lower panel shows how the information about the sources points is "summed up" at the expansion center. This summation can be done in a single pass. Evaluating the influence on an additional target point scales $\mathcal{O}(1)$.

3.2.3 Using a Gaussian Kernel

Assume the following definition of a multivariate Gaussian Kernel function:

$$\mathcal{K}(x,X) = \exp\left(-\|x-X\|^2/h^2\right)$$

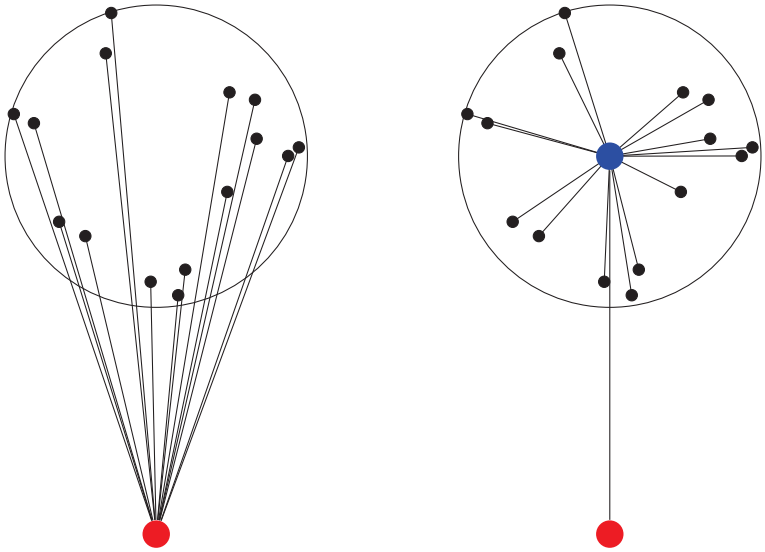

Figure 3: Encapsulate information at expansion center.

We skip the normalizing constant $1/\sqrt{2\pi}$ for notational convenience and shift it to the weight ν. The same bandwidth h applies in all dimensions by pre-transforming the data as:

$$X_{i,j} = h\frac{X^*_{i,j}}{\sqrt{2}h_j},$$

where h_j and $X^*_{i,j}$ denotes the bandwidth and the raw data in dimension j, respectively. The (weighted) sum of Gaussians at target point X is then given by:

$$\begin{aligned} G(X) &= \sum_{i=1}^{n} \exp\left(-\|x-X_i\|^2/h^2\right)\nu_i \\ &= \sum_{i=1}^{n} \exp\left\{-\|(x-c)-(X_i-c)\|^2/h^2\right\}\nu_i \\ &= \underbrace{\exp\left(-\|x-c\|^2/h^2\right)}_{*}\sum_{i=1}^{n}\underbrace{\exp\left(-\|X_i-c\|^2/h^2\right)}_{**}\underbrace{\exp\left\{2(x-c)(X_i-c)/h^2\right\}}_{***}\nu_i \end{aligned}$$

The first term $(*)$ is just a scalar. The second term $(**)$ depends only on the sources and can be precomputed in a single step. The last term $(***)$, however, is the computational bottleneck since the target points and the source points are entangled. To break this entanglement, Raykar et al. (2005) apply Taylor expansion about c:

$$\exp\left\{2(x-c)(X_i-c)/h^2\right\} = \sum_{|\alpha|\geq 0}\frac{2^\alpha}{\alpha!}\left(\frac{x-c}{h}\right)^\alpha\left(\frac{X_i-c}{h}\right)^\alpha$$

Where α denotes a multi-index of length d (see appendix 1). Substituting $(***)$ by the truncated

Taylor series yields:

$$
\begin{aligned}
G(x) &= \exp\left(-\|x-c\|^2/h^2\right)\sum_{i=1}^{n}\exp\left(-\|X_i-c\|^2/h^2\right)\sum_{|\alpha|\geq 0}\frac{2^\alpha}{\alpha!}\left(\frac{x-c}{h}\right)^\alpha\left(\frac{X_i-c}{h}\right)^\alpha \nu_i \\
&\approx \exp\left(-\|x-c\|^2/h^2\right)\sum_{|\alpha|<p}\frac{2^\alpha}{\alpha!}\left(\frac{x-c}{h}\right)^\alpha \underbrace{\sum_{i=1}^{n}\exp\left(-\|X_i-c\|^2/h^2\right)\left(\frac{X_i-c}{h}\right)^\alpha \nu_i}_{\star}
\end{aligned}
$$

By the means of the Cauchy-Schwarz inequality it can be shown that the error $\eta(p,c)$ caused by the truncated evaluation at source point X_i is bounded:

$$
\begin{aligned}
\eta_i(p,c) &= \exp\left(-\|x-X_i\|^2/h^2\right) - (*)\times(**)\sum_{|\alpha|<p}\frac{2^\alpha}{\alpha!}\left(\frac{x-c}{h}\right)^\alpha\left(\frac{X_i-c}{h}\right)^\alpha \\
&\leq \frac{2^p}{p!}\left(\frac{\|X_i-c\|}{h}\right)^p\left(\frac{\|x-c\|}{h}\right)^p \exp\{-(\|X_i-c\|-\|x-c\|)^2/h^2\} \qquad (16)
\end{aligned}
$$

This result is particular useful since it allows to set error bound a priori and choose the values of p and c such that the bound is met.
Now the sum in $(\star)$ is independent of the target points an can be evaluated in advance with $\mathcal{O}(n)$ work. Since there exist $\binom{p+d}{d}$ distinct multi-indices satisfying $|\alpha|<p$ the total runtime complexity is $\mathcal{O}\left(\binom{p+d}{d}n\right)$. Once the sum $(\star)$ is calculated, the evaluation at an additional target point t' needs only $\mathcal{O}\left(\binom{p+d}{d}\right)$ calculations. Hence, the approximative computation of the Matrix-Vector product in equation (9) can be done in $\mathcal{O}\left(\binom{p+d}{d}(n+m)\right)$. Using matrix notation gives a parsimonious representation:

$$
G \approx \begin{pmatrix} \Phi_{1,0} & \dots & \Phi_{1,p-1} \\ \vdots & \ddots & \vdots \\ \Phi_{m,0} & \dots & \Phi_{m,p-1} \end{pmatrix}
\begin{pmatrix} \Theta_{0,1} & \dots & \Theta_{0,n} \\ \vdots & \ddots & \vdots \\ \Theta_{p-1,1} & \dots & \Theta_{p-1,n} \end{pmatrix}
\begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix}
$$

With

$$
\begin{aligned}
\Phi_{j,|\alpha|} &= \exp\left(-\left\|x_j-c\right\|^2/h^2\right)\left(\frac{x_j-c}{h}\right)^\alpha \\
\Theta_{|\alpha|,i} &= \frac{2^\alpha}{\alpha!}\exp\left(-\|X_i-c\|^2/h^2\right)\left(\frac{X_i-c}{h}\right)^\alpha
\end{aligned}
$$

The crucial point here is that the series expansion must be sufficiently short to benefit from the approximation. If some sources are "far away" from the expansion center, a large number of basis functions is needed to keep the approximation error small. To circumvent this issue the data space can be split up into clusters and the expansion is done about the cluster centers for all elements within the cluster.
In their seminal work, Greengard and Strain (1991) used a uniform boxing scheme to subdivide the space. This approach shares the advantage of the binning procedures described in section

(3.1) to be easy to implement. On the downside, however, the number of boxes growths exponentially with the data dimension. An approach using kd-trees for data clustering is given by Lee et al. (2005) and Lee and Gray (2006). In this work we follow the approach taken by Gumerov and Duraiswami (2005) and Morariu et al. (2008) to partition source points into clusters by the means of the so-called "farthest-point clustering".

3.2.4 Data Clustering

Let K be the number of clusters to find in the data and $S \subset \mathbb{R}^d$ be the data space. The objective is to find a partition $\bigcup S_k = S$ such that the maximum of all cluster radii is minimized:

$$\arg \min_{(c_1,\ldots,c_K)} \max_k \max_{X \in S_k} \|X - c_k\|$$

Where c_k denotes the center of the cluster S_k. Bern and Eppstein (1996) show that finding the optimal solution is NP-hard and therefore not feasible if numerical performance is an issue. Nevertheless, Gonzalez (1985) gives a procedure that yield a solution that is not worse than twice the optimum but runs in $\mathcal{O}(nK)$. The algorithm proceeds as follows:
At the initial step a randomly chosen point $s_0 \in S$ is set as the first cluster center. In the next step the distance from every point in S to the first center is calculated. The point that is farthest from the initial center is added to the set of cluster centers. In every iteration the point is added to the set of cluster centers that is farthest from the existing set of cluster centers. That is, this point is further from its closest center point than any other point to its closest center. If the centers are defined, all points are assigned to its nearest center.
A particular useful characteristic of this procedure is that the number of clusters can be increased without altering the existing cluster centers. We will show that this is helpful to determine the runtime-optimal number of clusters. A refined algorithm with runtime $\mathcal{O}(n \log K)$ is given by Feder and Greene (1988).

3.2.5 Expansion about cluster centers

Data points within a cluster are close to its center by the very nature of the clustering algorithm. Hence, a Taylor expansion about a cluster center should converges relatively fast within the cluster. Altogether, the additional amount of work for the clustering procedure is generally overcompensated by savings due to the faster convergence of the Taylor expansion. From equation (16) it is apparent that the error bound gets tighter if the distances $\|X_i - c\|$ decreases. This also means that the series can be truncated at lower order to guarantee the error bound. Applying the Taylor expansion about every cluster center and summing the contribution of all clusters to target point x yields:

$$G(x) \approx \sum_{k=1}^{K} \exp\left(-\|x - c_k\|^2 / h^2\right) \sum_{|\alpha|<p} \frac{2^\alpha}{\alpha!} \left(\frac{x - c_k}{h}\right)^\alpha \sum_{X_i \in S_k} \exp\left(-\|X_i - c_k\|^2 / h^2\right) \left(\frac{X_i - c_k}{h}\right)^\alpha \nu_i$$

Using block matrix notation, the clumsy triple summation is avoided:

$$G \approx \begin{pmatrix} \Phi_1 & \dots & \Phi_K \end{pmatrix}_{(m \times K*p)} \begin{pmatrix} \Theta_1 \\ \vdots \\ \Theta_K \end{pmatrix}_{(K*p \times n)} \omega_{(n \times 1)}$$

with

$$\Phi_k = \begin{pmatrix} \Phi_{1,0,k} & \dots & \Phi_{1,p-1,k} \\ \vdots & \ddots & \vdots \\ \Phi_{m,0,k} & \dots & \Phi_{m,p-1,k} \end{pmatrix}_{(m \times p)}$$

$$\Theta_k = \begin{pmatrix} \Theta_{0,1,k} & \dots & \Theta_{0,n,k} \\ \vdots & \ddots & \vdots \\ \Theta_{p-1,1,k} & \dots & \Theta_{p-1,n,k} \end{pmatrix}_{(p \times n)}$$

and

$$\Phi_{j,|\alpha|,k} = \exp\left\{ - \left\| x_j - c_k \right\|^2 / h^2 \right\} \left(\frac{x_j - c_k}{h} \right)^{\alpha}$$

$$\Theta_{|\alpha|,i,k} = \frac{2^{\alpha}}{\alpha!} \exp\left\{ - \left\| X_i - c_k \right\|^2 / h^2 \right\} \left(\frac{X_i - c_k}{h} \right)^{\alpha} \mathbb{1}(X_i \in S_k)$$

Since the Gaussian function declines relatively fast, the matrices Φ_k contain many values that are almost zero. Hence, we introduce only a small additional error if we set these values equal zero but benefit from the sparsity of Φ_k since tailored methods for sparse matrix multiplication are applicable (cf. Yuster and Zwick (2005), Bell and Garland (2008) and Williams et al. (2009) for furhter readings). Raykar et al. (2005) suggest the following threshold:

$$\Phi_{j,|\alpha|,k} = \begin{cases} \exp\left(- \left\| x_j - c_k \right\|^2 / h^2 \right) \left(\frac{x_j - c_k}{h} \right)^{\alpha}, \left\| x_j - c_k \right\| \leq \max\limits_{X_i \in S_k} \left\| X_i - c_k \right\| + h\sqrt{\log(1/\eta)} \\ 0, \qquad \text{else} \end{cases} \tag{17}$$

Where η denotes the user-defined overall error tolerance. That is, the deviation from the direct calculation is bounded by:

$$\max_{x_j} \left\{ \frac{\left| \widetilde{G}(x_j) - G(x_j) \right|}{\sum\limits_{i=1}^{n} |\nu_i|} \right\} \leq \eta \tag{18}$$

3.2.6 Runtime Analysis

The runtime of the algorithm is influenced at three stages. The clustering procedure can be done in $\mathcal{O}(n \log(k))$ according to Feder and Greene (1988). The amount of work to calculate the "source dependent" part of the Taylor series is $\mathcal{O}(n\binom{p+d-1}{d})$. Evaluation of the "target dependent" part is influenced by the number of clusters for which the cluster center is closer

to the target point then the threshold given in equation (17). We denote this number as n_C. Altogether, the proposed algorithm has runtime of:

$$\mathcal{O}\left(n\log(K)+n\binom{p+d-1}{d}+m\binom{p+d-1}{d}n_C\right) \tag{19}$$

In contrast to the FFT approach presented in section (3.1), the runtime does not grow exponentially with the dimension d since:

$$\lim_{d\to\infty}\binom{p+d-1}{d}=d^p$$

Equation (19) also shows, that the runtime increases with the number of clusters K and the truncation p. Obviously, a large number of clusters allows an early truncation of the Taylor series and vice-versa. Finding the runtime-optimal combination of truncation order and the number of clusters is being addressed in the next section.

3.2.7 Parameter selection

As in all approximative algorithms, we trade speed for precision. For a fixed error bound η we try to find the combination of parameters K and p such that the expected runtime is minimized. A natural upper bound for a "fast algorithm" is the runtime of the direct evaluation. If it is not faster than direct evaluation it makes no sense to apply. Morariu et al. (2008) suggest a procedure for choosing the parameters optimally. The basic idea is as follows: The total deviation of the direct calculation and the approximative has two different sources. The first source is the error caused by the truncation of the Taylor series. The second error contribution comes from disregarding target points x that are "far away" from a cluster center. Hence, the deviation can be decomposed as:

$$\begin{aligned}|\widetilde{G}(x)-G(x)| \leq &\sum_{k=1}^{K}\sum_{X_i\in S_k}|v_i|\mathbb{1}\left\{\left\|x_j-c_k\right\|\leq\max_{X_i\in S_k}\|X_i-c_k\|+h\sqrt{\log(1/\eta)}\right\}\eta_{i,j}\\ &+\sum_{k=1}^{K}\sum_{X_i\in S_k}|v_i|\mathbb{1}\left\{\left\|x_j-c_k\right\|>\max_{X_i\in S_k}\|X_i-c_k\|+h\sqrt{\log(1/\eta)}\right\}\eta\end{aligned}$$

Given a fixed number of clusters, the truncation order is chosen as the smallest value of p such that the sum of all cluster-wise errors is not greater then the total error bound. The cluster set is extended as long as the estimated runtime for the Taylor expansion decreases substantially.

4 Simulation

To underpin the theoretical results about runtime complexity and approximation error of the proposed methods, we conduct a simulation study. For that we draw random samples from a multivariate standard normal distribution

$$X \sim \mathcal{N}(\mathbf{0}, \mathcal{I}_d)$$

and estimate the density using "direct" and "fast" approaches. As a benchmark we use the brute-force calculation of the multivariate kernel density estimator

$$\begin{pmatrix} \widehat{f}(x_1) \\ \vdots \\ \widehat{f}(x_n) \end{pmatrix} = \begin{pmatrix} \exp\left(-\|x_1 - x_1\|^2/h^2\right) & \dots & \exp\left(-\|x_1 - x_n\|^2/h^2\right) \\ \vdots & \ddots & \vdots \\ \exp\left(-\|x_n - x_1\|^2/h^2\right) & \dots & \exp\left(-\|x_n - x_n\|^2/h^2\right) \end{pmatrix} \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix}$$

with

$$\nu_i = \frac{1}{nh\sqrt{2\pi}}$$

It is noted that we do not compare the estimates obtained by the described "fast algorithms" to the true density. This is because the techniques are numerical approximations to the brute-force estimator and not estimators on their own right.
We use the following settings to simulate data:

$$\begin{aligned} n &= \{5000, 6000, \dots, 19000, 20000\} \\ M &= \{10, 20, \dots, 490, 500\} \\ d &= \{1, 2, 3\} \end{aligned}$$

The same grid size M is applied in all dimensions. That is, for $M = 100$ and $d = 3$ we map the raw data onto a grid of size $100 \times 100 \times 100$. For the three dimensional case we restricted the grid size to $M \leq 200$. The degree of smoothing in every dimension is chosen using the Rule-of-Thumb provided by Silverman (1986). All calculations are repeated 100 times to weaken the influence of outliers. All values presented are estimates of the median over these 100 repetitions.

4.1 Differences in the binning rules

We have seen that the linear binning rule is superior to constant binning in the sense that it causes a smaller approximation error than constant binning for a given sample size (cf. equation (14)). As a measure of closeness between the brute-force estimator ($\widehat{f}$) and the binned estimator ($\widetilde{f}$) we use the logarithm of the maximum absolute deviation:

$$\log\left[\max_i \left\{ |\widetilde{f}(x_i) - \widehat{f}(x_i)| \right\}\right]$$

The logarithmic transformation is used just for illustrative purposes.
The left graph in figure (4) shows the approximation error of the constant binning estimator and the linear binning estimator, respectively, for different sample sizes and different grid sizes in a one-dimensional setting. It can be seen, that the error caused by constant binning (upper sur-

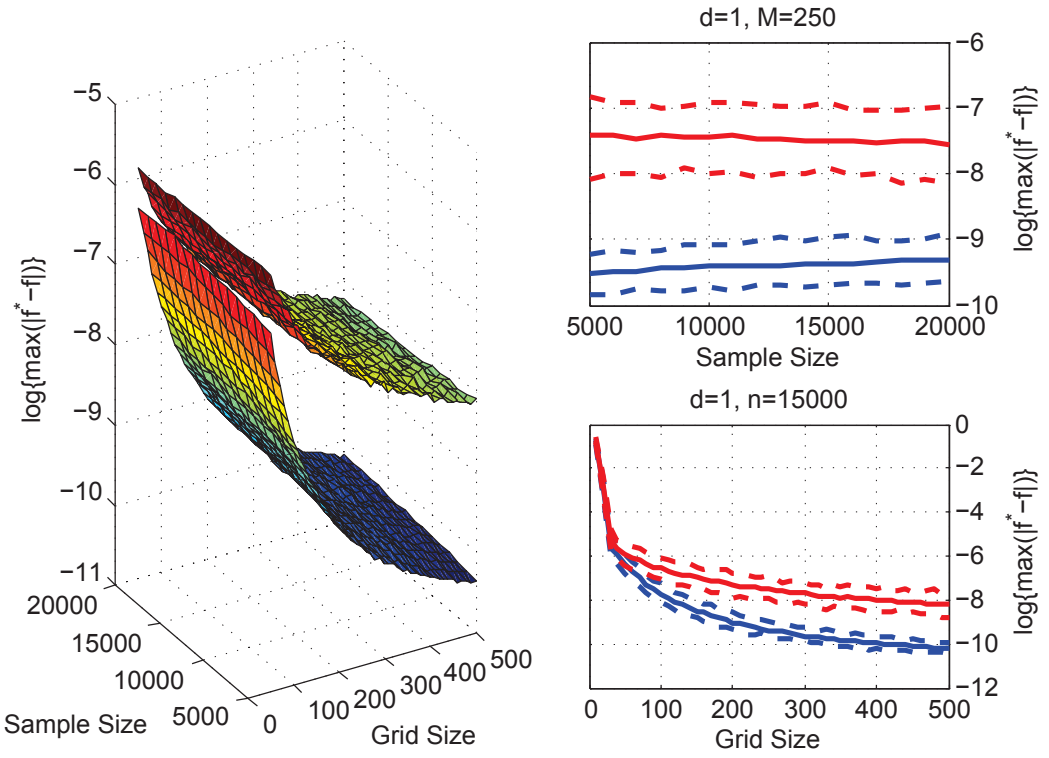

Figure 4: Maximum absolute Deviation for Constant and Linear Binning

face) is always higher then the error caused by linear binning (lower surface). The upper-right panel in figure (4) shows a cutout for a fixed grid size of $M = 250$ and varying sample size. The dashed lines represent the 2.5% and the 97.5% quantiles, respectively. It can be seen that the approximation error is almost independent of the sample size. Furthermore, it shows that linear binning (blue curve) is significantly better then constant binning (red curve). The lower-right panel shows the change in the approximation error for a fixed sample size and varying grid size. The result was to be expected: the error shrinks as the grid size grows.
Regarding the approximation error, the linear binning procedure seems to be preferable compared to constant binning, which confirms our theoretical findings. Since we always trade accuracy for speed, it is also necessary to compare the runtime of both algorithms to decide which one to use. Figure 5 shows the ratio of the runtime for both approaches:

$$\frac{t(\mathrm{LB})}{t(\mathrm{CB})}$$

It can be seen that for small grid sizes, constant binning runs about 4 times faster then linear binning. For larger sample sizes, however, the runtime is almost equal. This result also holds in higher dimensions. Since linear binning is almost always preferable, we neglect the constant binning procedure from this point on and use linear binning only.

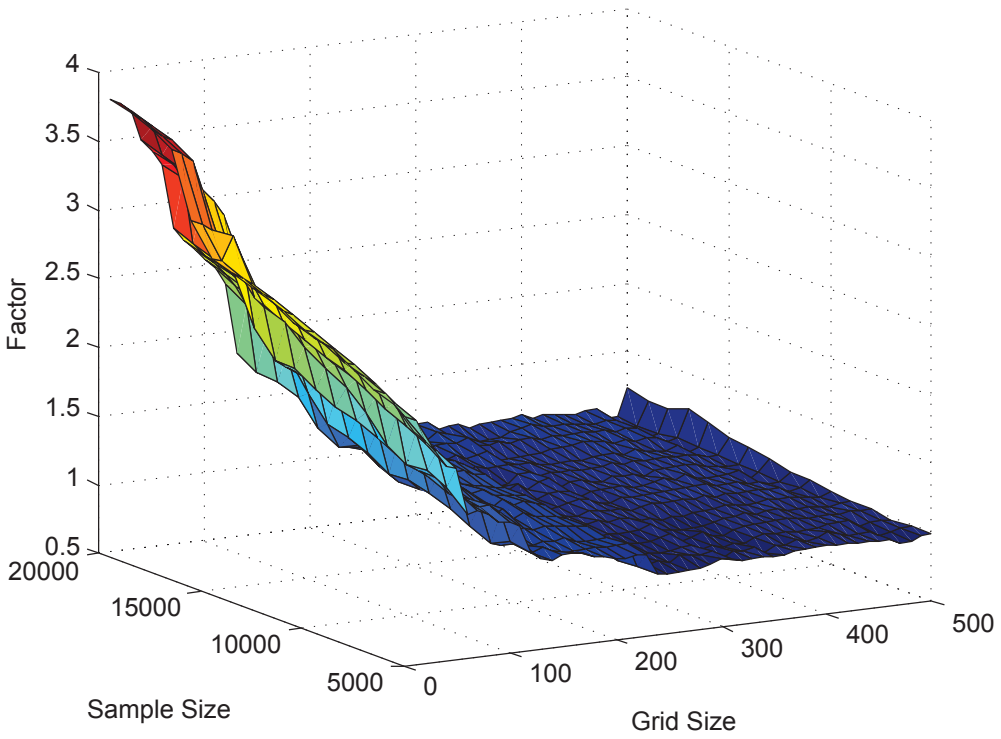

Figure 5: Runtime Constant Binning vs. Linear Binning

4.2 Differences in approaches

In the following we compare the performance of the FFT estimation procedure described in section (3.1) and the IFGT procedure from section (3.2) in terms of runtime. We especially focus on the change in runtime with increasing dimensionality. We take the following strategy: In a first step we compute the brute-force kernel density estimator $\widehat{f}(x)$. Next we compute the binned estimator $\widetilde{f}(x)$ using linear binning and FFT. Having the binned estimator at hand we can calculate the maximum absolute deviation relative to the sum of weights defined as:

$$\eta = \frac{\max_i \left\{ |\widetilde{f}(x_i) - \widehat{f}(x_i)| \right\}}{\sum_{i=1}^{n} |\omega_i|}$$

We set this quantity as an upper error bound that the IFGT algorithm has to guarantee. In this way we run both algorithms under the same conditions.
Nevertheless, a direct comparison has to be taken with caution since the performance is heavily driven by the quality of the implementation. For the FFT there exists a large number of highly optimized software packages like FFTW3 (Frigo and Johnson, 2005), Intels Math Kernel Library (MKL) and the cuFFT package from NVidia, to name only a few. These packages have gone through a continual optimization process during the past two decades and are pretty mature nowadays. In contrast, there exists only a single implementation of the IFGT procedure until

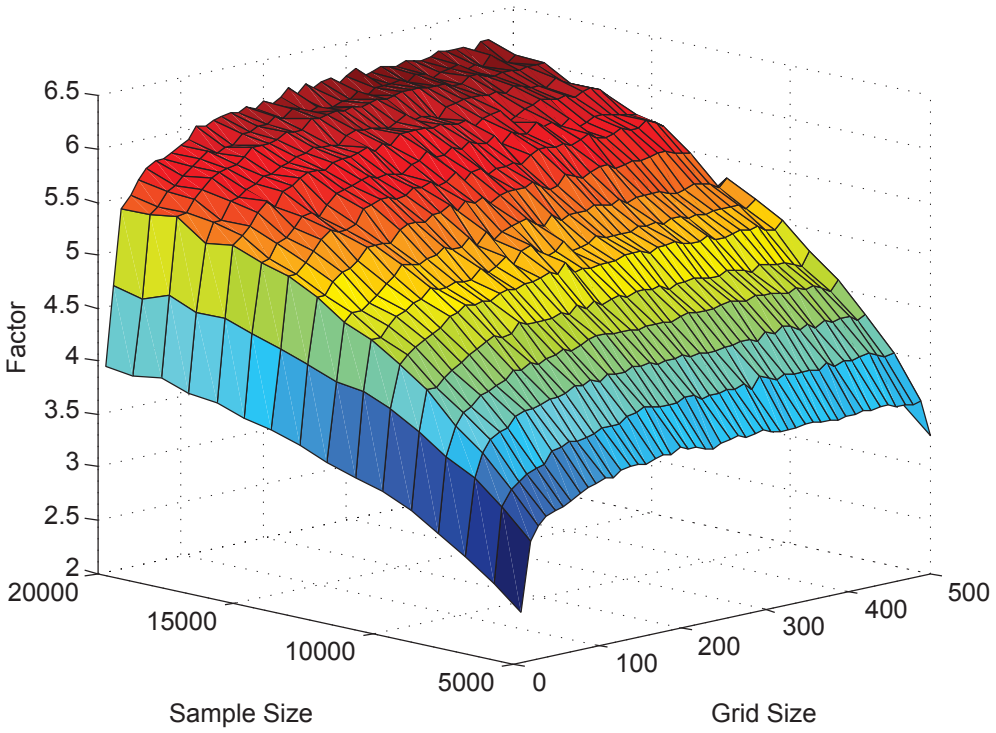

Figure 6: Ratio of runtime of IFGT to binned estimator in 1D

now of which we know. Therefore, we suppose that there is still scope for large improvements in the software architecture. The source code written in C++ with bindings to Matlab is publicly accessible at `http://www.umiacs.umd.edu/~morariu/figtree/`. A documentation can be found in Morariu et al. (2008). All calculations have been done on a Quadcore CPU with 16GB of RAM using Matlab 2012b 64bit.
Figures (6),(7) and (8) show the ratio of the runtime of both procedures in one, two and three dimensions, respectively. They show that the binned FFT procedure is notably faster than the IFGT for a given error bound, however collapses for high dimensions.
In the one-dimensional setting (6) the binned estimator is at least twice as fast as the IFGT estimator. The lead increases with the sample size. This was to be expected since the binned estimator is almost independent of the sample size.

From equations (13) and (19) we would expect that the runtime complexity of the binned estimator grows faster with the dimension d than the IFGT does. Figure (7) give first indications about that point. Even if the binned estimator is still faster in two dimensions, its lead shrinks considerably with a higher number of grid points. Moving to three dimensions the consequences of the rigid data clustering by binning become apparent. The upper panel in figure (8) shows that binning is only faster when using a fairly rough grid of $M \leq 50$ to $M \leq 80$ depending on the grid size. The lower panel shows only the sub-area for $M \geq 100$. It can be seen that now the binning procedure collapses for a higher number of grid points.

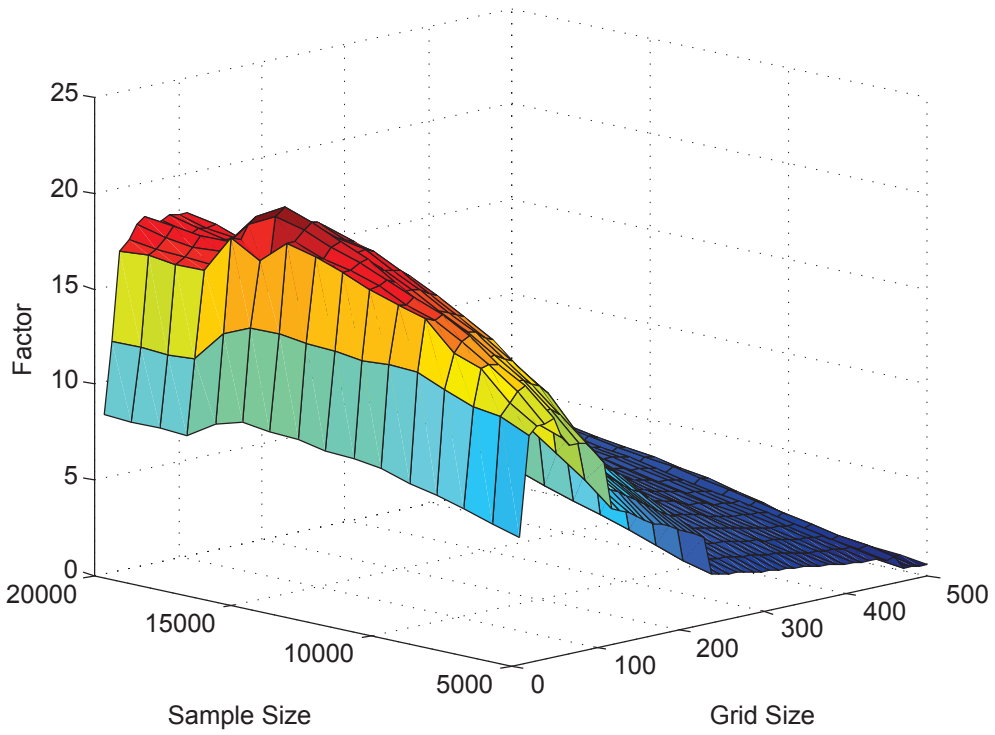

Figure 7: Ratio of runtime of IFGT to binned estimator in 2D

The simulation results support our considerations about the runtime depending on the sample size, the grid size and the dimensions. From that we draw the conclusion that the binning procedure is preferable in one and two dimensional cases regardless of the shape of the data. Although the IFGT has linear runtime, the overhead introduced by the more complex clustering and the parameter selection procedure cancel out this benefit. In higher dimensions, however, the opposite holds true.

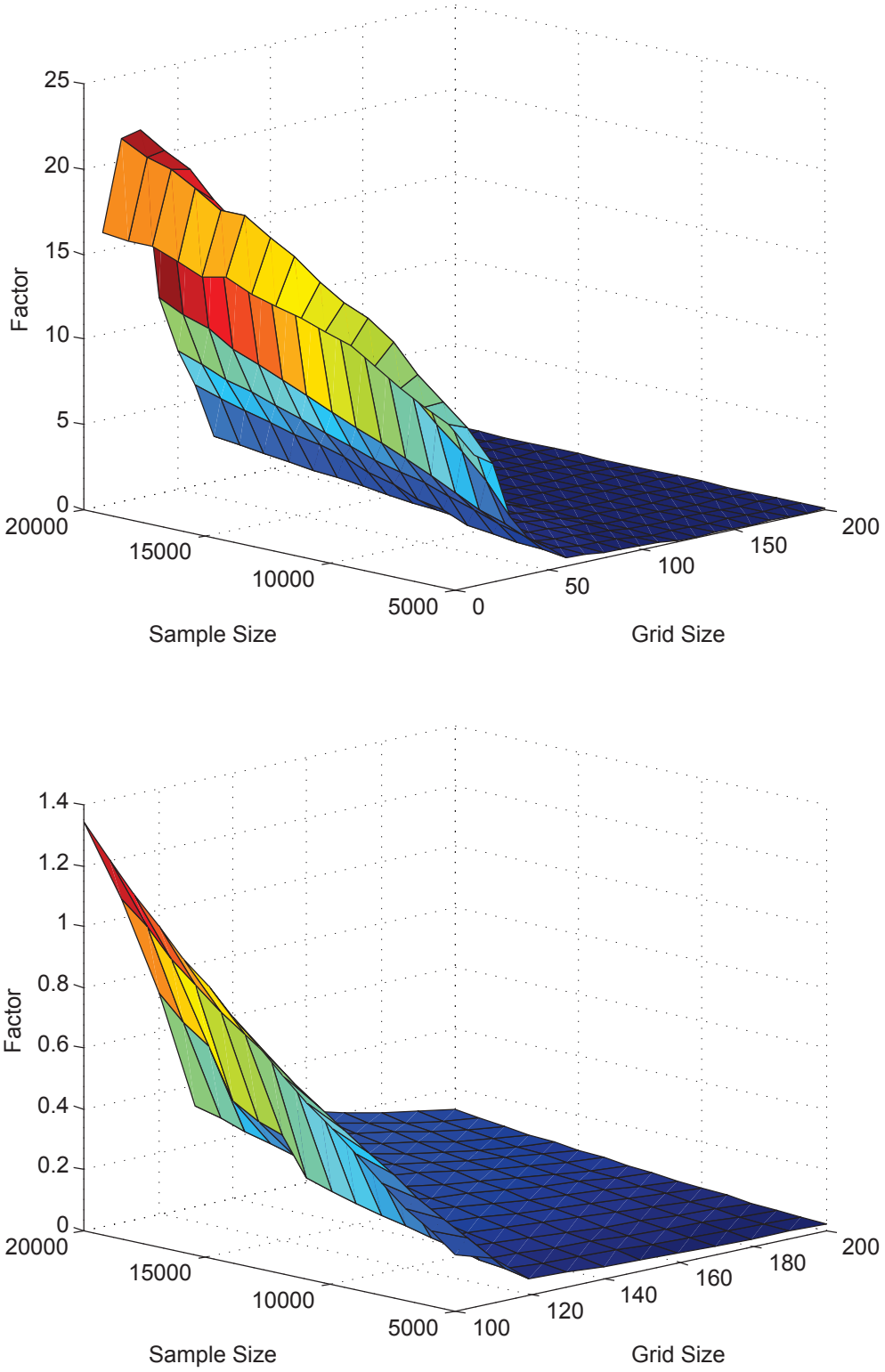

Figure 8: Ratio of runtime of IFGT to binned estimator in 3D

5 Application

The data set we use comes from a German company in the automotive supply industry. We investigate the energy consumption of a cooling unit. Among compressed air generation and heating, cooling is of particular interest since it is needed in many industrial processes. The sampling rate of the raw data is a five-minute interval, but we use hourly averages instead for two reasons. First, the outside temperature is only available on an hourly rate. Since this quantity is crucial in thermodynamical processes we are restricted to the coarse resolution. Second, taking averages over 12 observations in an hour reduces the effects of the system inertia as already pointed out in section (2.1). The time period of the measurements along with the number of observations is listed in Table (1).

	Time Period	n	valid n
Cooling Unit	01-Sep-2012 - 12-Jan-2014	11952	10951

Table 1: Time period of measurements

Due to measurement failure, not all observations are valid. We remove the whole observation if one of the variable is missing or an obvious outlier (e.g. negative energy consumption or temperatures of several hundred degrees).
In the following we split the data set into two disjoint groups. The reference period includes all values before 31-Aug-2013. The statistical model is build based on these values. The observations from 1-Sep-2013 onwards are used to evaluate the model. That is, we use the explanatory variables to predict the energy consumption in this period. In order to make predictions, we estimate the conditional mean with local constant and local linear estimators as well as the conditional quantiles using local constant quantile regression. The conditional mean can be estimated using one of the fast methods described in section (3). We utilize the Improved Fast Gauss Transform approach for two reasons. On the one hand, it shows better performance in settings with more then two dimensions. On the other hand, we re-use the clustering structure to speed-up quantile regression. The clustered version of the local constant quantile estimator from equation (7) is given by:

$$\min_{\beta_0} \sum_{k=1}^{K} \rho_\tau \left(\bar{Y}_k - \beta_0\right) \mathcal{K}_h(c_k - x)$$

5.1 The Cooling System

The energy consumption of a cooling unit is essentially influenced by two factors. First, the difference between the temperature of the inflowing and the outflowing refrigerant $\Delta(t)$. Secondly, the quantity of refrigerant to be cooled down. Additionally, the outside temperature is also of interest since the refrigerant is usually pre-cooled in a cooling tower at the roof of the factory building if the outside temperature is low enough. This reduces the energy consumption of the cooling machine but causes energy consumption of the cooling tower. The quantity of interest is the overall energy consumption of the whole cooling system. The air humidity might also be important since it influences the heat exchange in the cooling tower but the effect is usually negligible. Table (2) shows summary statistics for the variables mentioned.

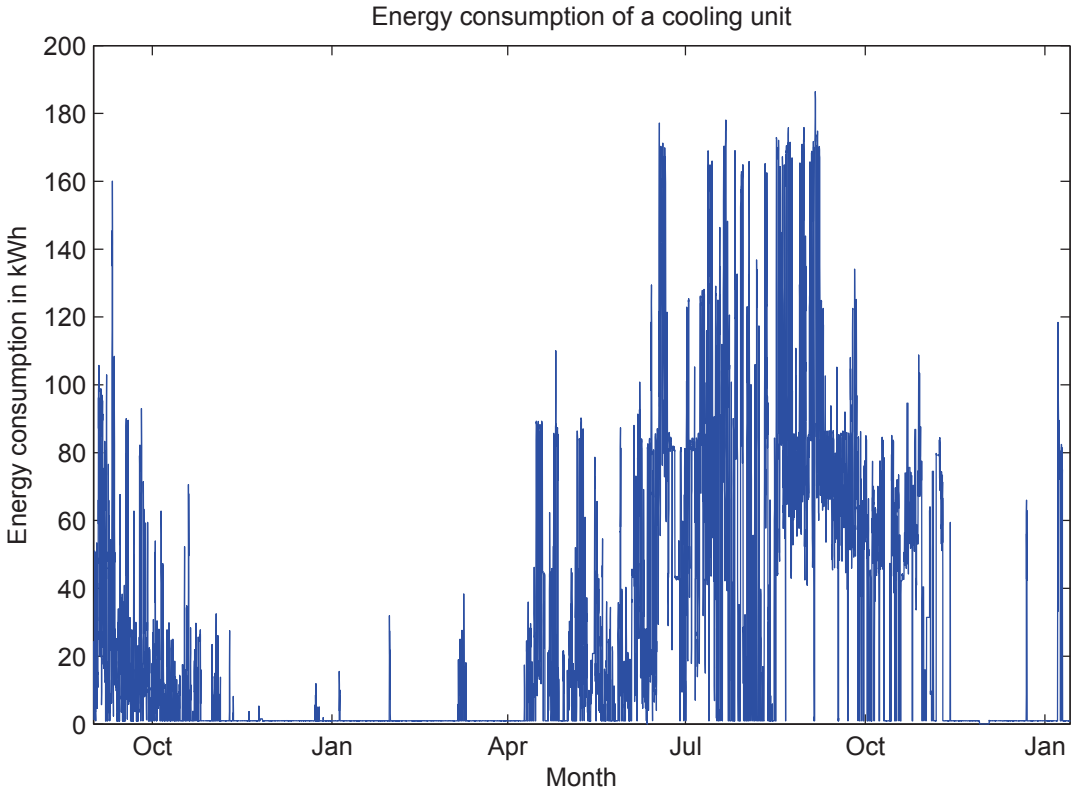

Figure 9: Energy Consumption of the cooling unit from Sep 2012 to Jan 2014

	mean	sd	min	max
Energy Consumption in kWh	24.6	37.8	0	186.44
Volume Flow in m^3/h	249.5	377.2	0	1638
Outside Temperature in C	9.0	7.9	-10.2	36.0
Outflow Temperature in C	14.4	3.8	0.1	22.9
Relative Humidity in %	79.2	16.2	19.5	100

Table 2: Summary Statistics Cooling Unit

The energy consumption from Sep-2012 to Jan-2014 is shown in figure (9). Not surprisingly, cooling is mostly needed in warm months. We remove all observations where the cooling unit is only in standby mode. Figure (10) shows scatterplots between the energy consumption and the explanatory variables. They suggest that the volume flow rate has notable influence on the energy consumption. The dependency looks fairly linear with increasing dispersion for higher flow rates. From the upper right panel it can be seen that the energy consumption tends to increase with the outside temperature. The opposite holds for the outflow temperature. The functional form, however, is not obvious from visual inspection. The relative humidity seems to have no visible influence on the energy consumption. Thus, visual inspection supports our theoretical considerations about the explanatory variables. We use the following model to draw

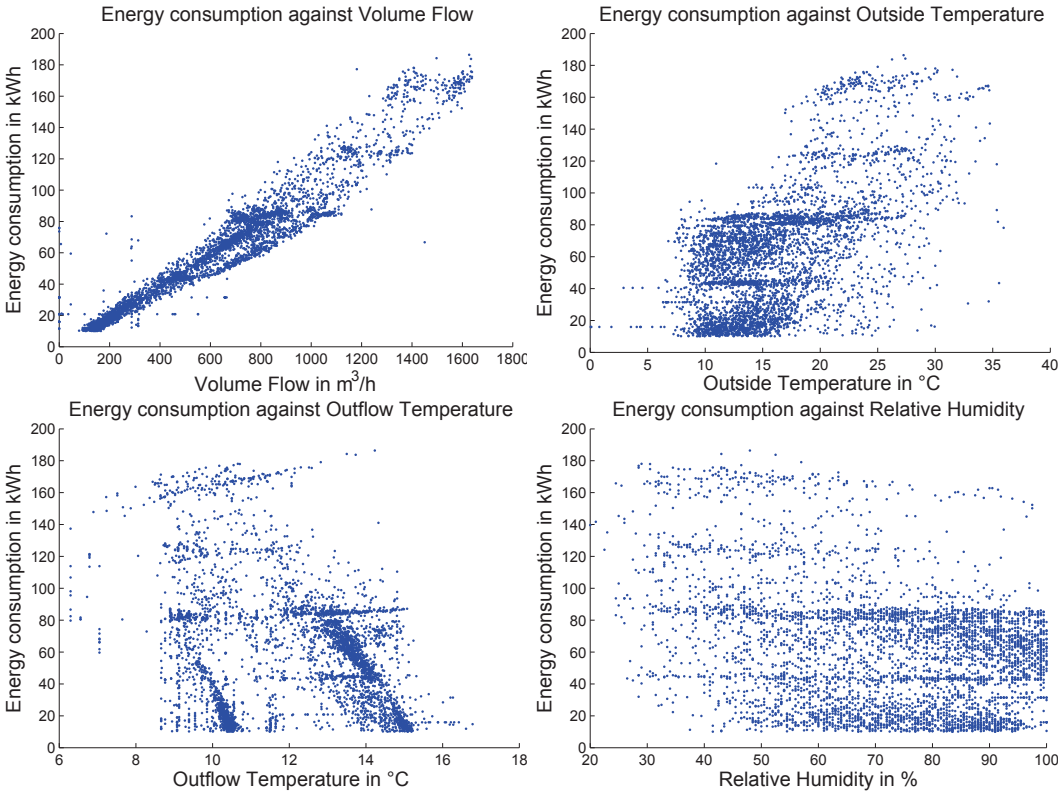

Figure 10: Scatterplot between Energy Consumption and the explanatory variables

conclusions from the data:

$$y_i = m(x_{1,i}, x_{2,i}, x_{3,i}) + \varepsilon_i$$

where x_1 denotes the volume flow, x_2 is the outside temperature and x_3 stands for the outflow temperature. Using matrix-vector product notation from section (2.1), the local constant estimator is given by

$$\widehat{m}_h^{\mathrm{LC}}(x) = \frac{T_{h,0}(x)}{S_{h,0}(x)}$$

with

$$\begin{pmatrix} T_{h,0}(x_1) & S_{h,0}(x_1) \\ \vdots & \vdots \\ T_{h,0}(x_m) & S_{h,0}(x_m) \end{pmatrix} = \begin{pmatrix} \mathcal{K}_h(x_1 - X_1) & \dots & \mathcal{K}_h(x_1 - X_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}_h(x_m - X_1) & \dots & \mathcal{K}_h(x_m - X_n) \end{pmatrix} \begin{pmatrix} y_1 & 1 \\ \vdots & \vdots \\ y_n & 1 \end{pmatrix}$$

The overall error bound for the IFGT algorithm is set to $\eta = 0.001$. The number of clusters and the truncation order are chosen as $K = 142$ and $p = 15$, respectively, according to the parameter selection procedure presented by Morariu et al. (2008). We skip the explicit formulation of the local linear estimator for sake of compactness. The predicted values of the energy consumption for the evaluation period are obtain by plugging the explanatory variables from the evaluation period into the estimator:

$$\widehat{y}^{\mathrm{eval}} = \widehat{m}_h(x^{\mathrm{eval}})$$

Table (3) shows the Root Mean Squared Error (RMSE), the maximum absolute deviation and the Mean Average Percentage Error (MAPE) as performance measures for the local constant and the local linear estimator. As to be expected from theory, the local linear estimator outperforms the local constant estimator in terms of accuracy.

	RMSE	$\max \lvert y - \widehat{y} \rvert$	MAPE
Local Constant	20.4	130.1	20.61
Local Linear	15.6	100.4	15.95

Table 3: Performance Measures for Local Constant and Local Linear estimators

Figure (11) shows the realized energy consumption in the evaluation period along with the estimated 95% quantile. It can be seen that in the period from end of September to the beginning of October the realized energy consumption is higher than the estimated quantile. As already mentioned in section (1) we read this as a type of malfunctioning of the machinery. The upper panel of Figure (12) shows this time period in greater detail. Additionally to the quantiles, the estimated values from the local linear estimator are plotted. The lower panel shows a period where the realized values are always below the estimated quantile. This indicates that the machine runs regularly.

In practice this means that the machine need maintenance. If the malfunctioning does not cause any visible or audible change in the running machine, “unusual” energy consumption would not be notified without having a statistical threshold. This can result in a further damage of the engine and causes unnecessary costs. The plot suggests that the malfunctioning has been fixed at beginning of October.

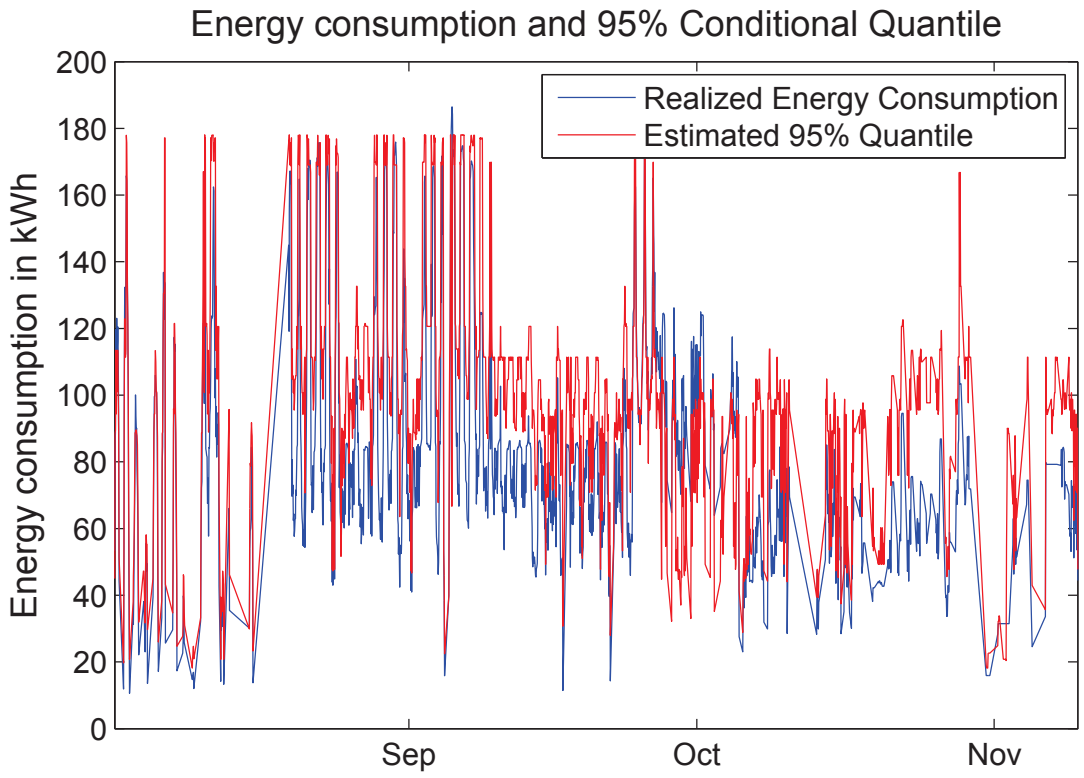

Figure 11: Realized Energy Consumption and Fitted Values in the Evaluation Period

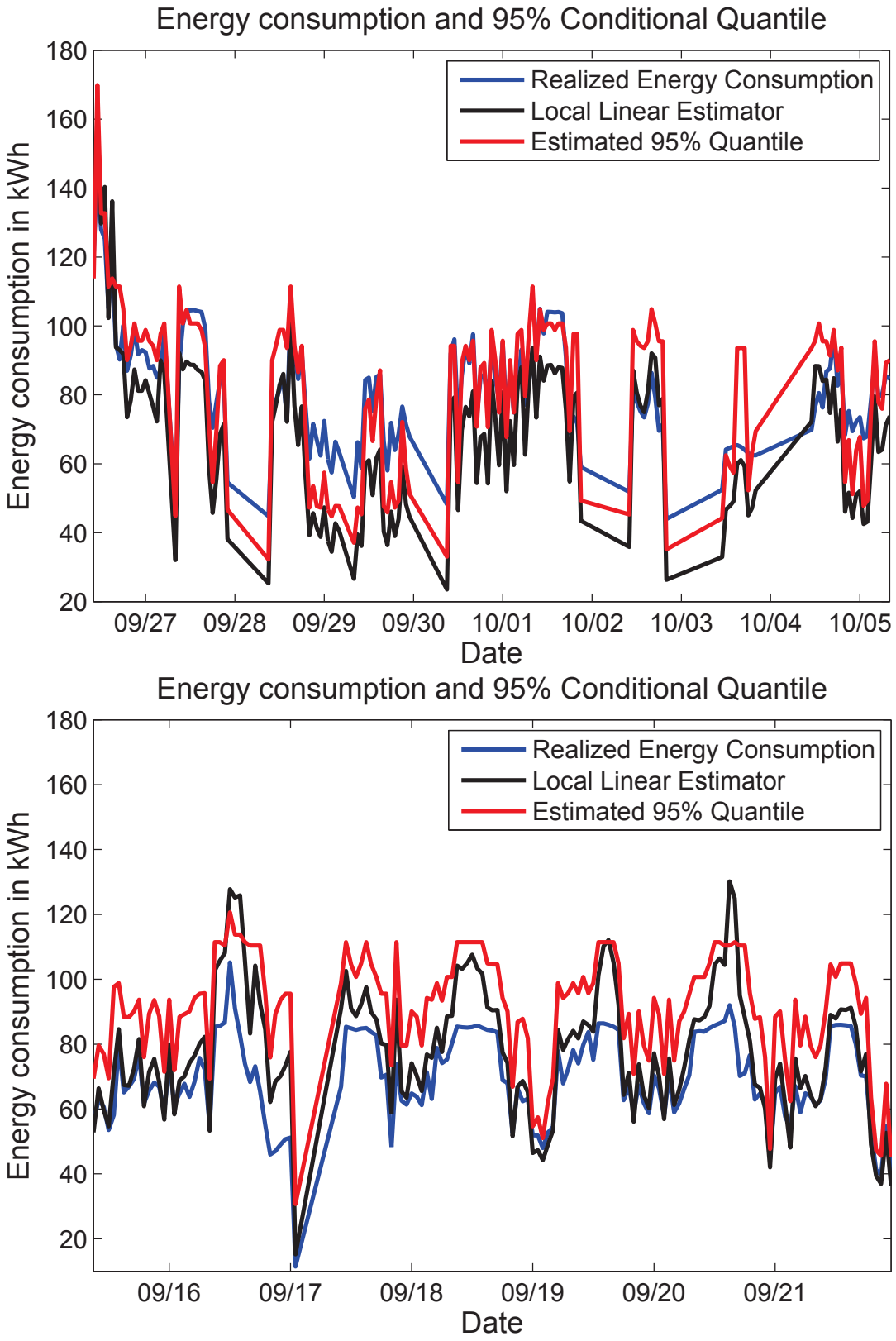

Figure 12: Malfunctioning (upper panel) and regular working (lower panel) periods

6 Conclusion

In this thesis the energy consumption of supply engineering machineries is studied. The focus is on statistical tools to detect potential savings due to replacement of old machineries and to identify situations where maintenance is needed. Because there exists a vast variety of supply engineering machineries statistical models must be highly flexible. Kernel regressions techniques provide this flexibility. Kernel mean regression is used to estimate the expected energy consumption given external influences. To estimate the "usual" dispersion of the energy consumption, Kernel quantile regression methods are employed.

The flexibility in modeling comes at the expense of high computational complexity. In practice, however, numerical performance is an issue because many engines are monitored simultaneously. To reconcile model flexibility and numerical usability we use approximative algorithms to speed-up calculations and approve losses in precision. Two types of algorithms are proposed. The "binned estimator" gains speed by mapping continuous data onto a regular grid and exploiting the regularity by the means of the Fast Fourier Transform. The "Fast Gauss Transform" approach uses truncated Taylor series expansion about cluster centers obtained by an approximative k-center clustering algorithm to improve numerical performance.

Our simulation study shows that the binned estimator runs orders of magnitude faster than the Fast Gauss Transform approach in a one and two-dimensional settings given the same loss in precision. In higher dimensions, however, the binned estimator suffers from the curse of dimensionality due to its rigid data clustering. The more complex clustering approach in the Fast Gauss Transform pays off in higher dimensions.

Application on data from a cooling unit show that the proposed methods are in fact useful to detect malfunctioning. Until now, it is common practice to maintain machineries in regular time intervals, e.g. once a year. If a malfunctioning happens in between it is usually not considered. This results in higher energy costs. The proposed methods can help to reduce energy costs and to extend the life of industrial machineries.

Appendix

1 Multi-index notation

According to Masry (1996) and Masry (1997) a multi-index $\alpha = (\alpha_1, \ldots, \alpha_d), \alpha \in \mathbb{N}^d$ is a tuple of non-negative integers with the following properties:

- $|\alpha| = \sum_{j=1}^{d} \alpha_j = n$
- $\alpha! = \alpha_1! \times \alpha_2! \times \ldots \times \alpha_d!$
- $x \in \mathbb{R}^d : x^\alpha = x_1^{\alpha_1} \times \ldots \times x_d^{\alpha_d}$
- Let $x, y \in \mathbb{R}^d$ and $xy = \sum_{j=1}^{d} x_j y_j$ then $(xy)^n = \sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha y^\alpha$

Bibliography

Beatson, R. and Greengard, L. (1997). A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, pages 1–37.

Bell, N. and Garland, M. (2008). Efficient sparse matrix-vector multiplication on cuda. Technical report, NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation.

Bern, M. and Eppstein, D. (1996). Approximation algorithms for geometric problems. *Approximation algorithms for NP-hard problems*, pages 296–345.

Bluestein, L. (1970). A linear filtering approach to the computation of discrete fourier transform. *Audio and Electroacoustics, IEEE Transactions on*, 18(4):451–455.

Brockwell, R. (1986). The fourier transform and its application.

Callen, H. B. (2006). *Thermodynamics and an Introduction to Thermostatistics 2nd Ed.* Wiley Online Library.

Chan, T. and Olkin, J. (1994). Circulant preconditioners for toeplitz-block matrices. *Numerical Algorithms*, 6(1):89–101.

Chan, T. F. (1988). An optimal circulant preconditioner for toeplitz systems. *SIAM journal on scientific and statistical computing*, 9(4):766–771.

Cipra, B. A. (2000). The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836.

Coifman, R., Rokhlin, V., and Wandzura, S. (1993). The fast multipole method for the wave equation: A pedestrian prescription. *Antennas and Propagation Magazine, IEEE*, 35(3):7–12.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

Duhamel, P. and Hollmann, H. (1984). Split radix'fft algorithm. *Electronics Letters*, 20(1):14–16.

Duhamel, P. and Vetterli, M. (1990). Fast fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299.

Dutt, A. and Rokhlin, V. (1993). Fast fourier transforms for nonequispaced data. *SIAM Journal on Scientific computing*, 14(6):1368–1393.

Fan, J. (2003). *Nonlinear time series: nonparametric and parametric methods*. Springer.

Fan, J. and Gijbels, I. (1996). Local polynomial modelling and its applications. *Chapman, Hall, London*.

Fan, J. and Marron, J. S. (1994). Fast implementations of nonparametric curve estimators. *Journal of Computational and Graphical Statistics*, 3(1):35–56.

Feder, T. and Greene, D. (1988). Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM.

Frigo, M. and Johnson, S. G. (2005). The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231.

Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 4. JHU Press.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

Gray, R. M. (2006). Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239.

Greengard, L., Huang, J., Rokhlin, V., and Wandzura, S. (1998). Accelerating fast multipole methods for the helmholtz equation at low frequencies. *Computational Science & Engineering, IEEE*, 5(3):32–38.

Greengard, L. and Lee, J.-Y. (2004). Accelerating the nonuniform fast fourier transform. *SIAM review*, 46(3):443–454.

Greengard, L. and Rokhlin, V. (1987). A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348.

Greengard, L. and Rokhlin, V. (1997). A new version of the fast multipole method for the laplace equation in three dimensions. *Acta numerica*, 6(1):229–269.

Greengard, L. and Strain, J. (1991). The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94.

Gumerov, N. A. and Duraiswami, R. (2005). *Fast multipole methods for the Helmholtz equation in three dimensions*. Access Online via Elsevier.

Hall, P. (1982). The influence of rounding errors on some nonparametric estimators of a density and its derivatives. *SIAM Journal on Applied Mathematics*, 42(2):390–399.

Hall, P. and Wand, M. (1996). On the accuracy of binned kernel density estimators. *Journal of Multivariate Analysis*, 56(2):165–184.

Härdle, W. (1987). Algorithm as 222: Resistant smoothing using the fast fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 36(1):104–111.

Härdle, W., Lütkepohl, H., and Chen, R. (1997). A review of nonparametric time series analysis. *International Statistical Review*, 65(1):49–72.

Härdle, W., Mueller, M., and Sperlich, S. (2004). *Nonparametric and semiparametric models*. Springer Verlag.

Hassanieh, H., Indyk, P., Katabi, D., and Price, E. (2012a). Nearly optimal sparse fourier transform. In *Proceedings of the 44th symposium on Theory of Computing*, pages 563–578. ACM.

Hassanieh, H., Indyk, P., Katabi, D., and Price, E. (2012b). Simple and practical algorithm for sparse fourier transform. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1183–1194. SIAM.

Hunt, B. (1971). A matrix theory proof of the discrete convolution theorem. *Audio and Electroacoustics, IEEE Transactions on*, 19(4):285–288.

Jones, M. C. (1989). Discretized and interpolated kernel density estimates. *Journal of the American Statistical Association*, 84(407):733–741.

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50.

Kolba, D. and Parks, T. (1977). A prime factor fft algorithm using high-speed convolution. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(4):281–294.

Lee, D. and Gray, A. (2006). Faster gaussian summation: Theory and experiment. In *In Proceedings of the Twenty-second Conference on Uncertainty in Artificial Intelligence*. Citeseer.

Lee, D., Moore, A. W., and Gray, A. G. (2005). Dual-tree fast gauss transforms. In *Advances in Neural Information Processing Systems*, pages 747–754.

Li, Q. and Racine, J. S. (2007). *Nonparametric econometrics: Theory and practice*. Princeton University Press.

Marron, J. S. and Wand, M. P. (1992). Exact mean integrated squared error. *The Annals of Statistics*, 20(2):712–736.

Masry, E. (1996). Multivariate local polynomial regression for time series: uniform strong consistency and rates. *Journal of Time Series Analysis*, 17(6):571–599.

Masry, E. (1997). Local polynomial estimation of regression functions for mixing processes. *Scandinavian Journal of Statistics*, 24(2):165–179.

Morariu, V. I., Srinivasan, B. V., Raykar, V. C., Duraiswami, R., and Davis, L. S. (2008). Automatic online tuning for fast gaussian summation. In *Advances in Neural Information Processing Systems*, pages 1113–1120.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Rader, C. M. (1968). Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108.

Raykar, V. C., Yang, C., Duraiswami, R., and Gumerov, N. (2005). Fast computation of sums of gaussians in high dimensions. Technical report, University of Maryland - Computer Science Department.

Sarma, M. S. (2001). *Introduction to electrical engineering*. Oxford University Press.

Scott, D. W. and Sheather, S. J. (1985). Kernel density estimation with binned data. *Communications in Statistics-Theory and Methods*, 14(6):1353–1359.

Silverman, B. (1982). Algorithm as 176: Kernel density estimation using the fast fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(1):93–99.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC press.

Stein, E. M. and Shakarchi, R. (2003). *Princeton Lectures in Analysis*. Princeton University Press.

Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics*, 5(4):595–620.

Ugural, A. (2003). *Mechanical design: an integrated approach*. McGraw-Hill Science/Engineering/Math.

Van Loan, C. (1992). *Computational frameworks for the fast Fourier transform*, volume 10. Siam.

Wand, M. (1994). Fast computation of multivariate kernel estimators. *Journal of Computational and Graphical Statistics*, 3(4):433–445.

Wand, M. M. P. and Jones, M. C. (1995). *Kernel smoothing*, volume 60. Crc Press.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372.

Wefers, F. and Vorländer, M. (2012). Potential of non-uniformly partitioned convolution with freely adaptable fft sizes. In *Audio Engineering Society Convention 133*.

Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., and Demmel, J. (2009). Optimization of sparse matrix–vector multiplication on emerging multicore platforms. *Parallel Computing*, 35(3):178–194.

Winograd, S. (1978). On computing the discrete fourier transform. *Mathematics of computation*, 32(141):175–199.

Wong, M. W. (2011). *Discrete fourier analysis*, volume 5. Springer.

Yang, C., Duraiswami, R., Gumerov, N. A., and Davis, L. (2003). Improved fast gauss transform and efficient kernel density estimation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 664–671. IEEE.

Yu, K. and Jones, M. (1998). Local linear quantile regression. *Journal of the American statistical Association*, 93(441):228–237.

Yuster, R. and Zwick, U. (2005). Fast sparse matrix multiplication. *ACM Transactions on Algorithms (TALG)*, 1(1):2–13.

List of Figures

List of Tables

Declaration of Authorship

I hereby confirm that I have authored this master thesis independently and without use of others than the indicated sources. Where I have consulted the published work of others, in any form (e.g. ideas, equations, figures, text, tables), this is always explicitly attributed.

Berlin, January 30, 2014 Simon Diehl