Title: Comparing numerical performance of second generation wavelets and the nonparametric estimators in random design regression models

By

Mohammadhossein Aberoumand

A Thesis submitted to

for the degree of STAT825: Statistical Project

Department of Statistics

June 2019

MACQUARIE
University
SYDNEY · AUSTRALIA

Typesetin $\LaTeX\,2_\varepsilon$.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Mohammadhossein Aberoumand

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Justin Wishart for his help, support and patience for my Statistical project.

Abstract

Since wavelet methods were introduced three decades ago they have captured the attention of scientists in different areas. In the discipline of statistics, these methods have been implemented in different areas such as nonparametric regression, time series analysis and image denoising. In this project we have compared the relatively new second generation wavelet method with the first generation method and some commonly used nonparametric methods. We have tested these methods on non-equally-spaced grids and recorded the MSEs and MAEs. Based on the results of this project, the second generation wavelet method performs as well as the common nonparametric methods and is substantially better than the first generation in terms of error.

Table of Contents

Dedication

This project is wholeheartedly dedicated to my family who have continuously supported me spiritually, emotionally and economically through my studies. To my friends whose love and kindness has given me the energy and courage to work hard and finish my studies.

Chapter 1

Introduction

One of the most important topics in statistics is to model the mean value of response variable and linear statistical model are a classical tool to do so. Due to their simplistic nature, these family of models have been at the centre of attention for a long time. These models define a relationship between a response variable Y and a set of independent variable $X_1, X_2, \ldots, X_k$ such as the following:

$$Y = \beta_0 + \beta_1 X_1 + \epsilon$$

$$\epsilon \sim N(0,1)$$

where β_0, β_1 unknown parameter that we wish to estimate. Linear regression are discussed in more details in(Wackerly, Mendenhall, & Scheaffer, 2014). Although, these models are based on linear relationships between variables, there will e many cases where we wish to investigate nonlinear relationships between the response and an independent variable. The general form of the model corresponds to it is the following:

$$Y = f(X) + \epsilon, \quad \epsilon \sim N(0,1) \tag{1.1}$$

In equation (1.1) function $f \in C^r$ which mean its derivatives exist till order r and it is bounded. When the relationship is nonlinear simple linear regression is not good choice because it is linear in nature and therefore not very flexible and although it will a have relatively small variance, its bias will be considerably large. Therefore, in chapter 2, we will briefly discuss some *nonparametric methods* that can be used to find f such as *Kernel smoothing*, *Splines* and *Wavelet methods* these method are more flexible compared to simple linear regression and will be less biased and by tuning their hyper parameters correctly we can avoid over-fitting. For instance, choice of bandwidth for *Kernel smoothing* and number of knots for *Spline* are crucial to get a good fit for the data. In this project we wish to compare first and second generation of *wavelet method* with two other class of estimators mentioned above. The *First generation wavelet method* are known to be applicable to the equally spaced grid However, They are not equally effective in the non equally spaced grid case due to its reliance on Fourier transformation. Since data-sets with non equally spaced grid are common in the real world cases over the past years a newly methods called *second generation wavelet* method has been developed by scientists which can handle a non equally spaced grid. This algorithm benefits from a framework called *lifting scheme* which does not rely on Fourier transformation. The second generation has been tested on real data-sets in (Hamilton, Nunes, Knight, & Fryzlewicz, 2018). However, no comparison has been made between first and second generation wavelet method on a non equally spaced grid data. In this project we wish to put these methods to the test in numerous scenarios.

In chapter three, we will discuss the methods to simulated the data from a non equally space grid. To create a non equally spaced grid we will use a *random design model*. In general, if in (1.1) only responses$Y_1, \ldots, Y_N$ are treated as random and the sample $X_1, \ldots, X_N$ treated to be fixed values this is called a *fixed design*. In contrast, if $(X_1, Y_1), \ldots, (X_N, Y_N)$ considered to be random pairs this is called *random design*(Hsu, Kakade, & Zhang, 2011). For demonstration purposes we will represent some graph of basic function and signals and the estimated curved by different methods.

In chapter four, we wish to compare these estimators by Mean square error and Mean absolute error. we will present the results of the simulations. Three type of grid will be generated from uniform, left skewed and right-skewed distributions. This is important due to nonsymmetric shape of the signals which will result in different amount of variation along the domain. In addition we will change the amount of noise added each time by changing the signal to noise ratio.

The preface pretty much says it all.

Second paragraph of abstract starts here.

Chapter 2

Non-parametric regression

In this chapter we will discuss different ideas and methods for nonparametric regression (Martinez & Martinez, 2007) but before explaining the nonparametric methods we will recall the linear regression model this will help us have a better understanding of different approaches to this problem. A refinement of the previous linear regression model that could move in nonlinear patterns could be a ploynomial regression:

$$Y_i = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_p X^p + \epsilon, \quad p \in \mathbb{N} \tag{2.1}$$

we have to note that besides β we have used polynomial term such as X^2 the model is considered to be *linear* with respect to coefficients β_i. This model is considered to be *parametric* because we assume a certain model for the existing relationship between predictors and the response and estimate the coefficients. In general, parametric model have certain assumptions and are considered to be less flexible. We usually fit these models by estimating the parameter using various estimation methods. In case of linear regression, the model can be written in matrix format $Y = X\beta + \epsilon$ and we assume $E(\epsilon) = 0$ and $V(\epsilon) = \sigma^2 I$. The estimate for the vector of parameter can be driven by solving the following equation: $\beta = (X^T X)^{-1} X^T Y$.

In nonparametric case we do not assume a parametric format for our model and we will try to fit a more flexible model that could be expressed in the following format:

$$Y = f(X_i) + \epsilon$$

Generally speaking, function f in this equation is consider to be smooth and bounded function that allow nonlinear relation between the predictors. In this project we focus on the cases where there is one response and one predictor variable. In this problem we are after the functions that can minimize the error term which is essentially the difference between the true value of the function and the estimated values. This has been given in the following expression:

$$E[(Y_i - \hat{f}(X_i))^2]$$

where $\hat{f}(X_i)$ is the estimate of the function f at $i - th$ position . In the following we will dicuss the *kernel smoothing* method:

2.1 Kernel Smoothing

Kernel smoothing estimator locally estimates regression which means it fits an estimate $\hat{y}_0$ for each point x_0. This estimate is a n-th degree polynomial which is estimated using weighted least square of the point in neighborhood of the point x_0. In order to get a better estimate, closer points will be associated with larger weight and further points will be associated will smaller weights. This will be achieved by using a kernel function and usually has a smoothing parameter which controls the bandwidth of the neighborhood; moreover, it influences the weights that points would take. The polynomial at each point can be express by the following equation:

$$\hat{Y} = \beta_0 + \beta_1(X_i - x) + \cdots + \beta_d(X_i - x)^d \tag{2.2}$$

Now we associate each point with the weight obtain from the our kernel function

$$K_h(X_i - x) = \frac{1}{h}K(\frac{X_i - x}{h}) \tag{2.3}$$

where in (2.3), h is the bandwidth. If the bandwidth is very small the curve become non smooth and wiggly because it only depends on the point in a very small neighborhood of the point this may result in over fitting. In contrast, If the h is chosen too large the estimated curve will become smooth and the bias of the model will be large. We are after $\hat{\beta}_i$ which minimize the following equation:

$$\Sigma_{i=1}^{n} K_h(X_i - x)(Y_i - \beta_0 - \beta_1(X_i - x) - \cdots - \beta_d(X_i - x)^d)^2$$

Now if we put our terms in a matrix format we can have the following:

$$X_x = \begin{pmatrix} 1 & X_1 - x & \dots & (X_1 - x)^d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n - x & \dots & (X_n - x)^d \end{pmatrix}$$

and weights can be put in a diagonal matrix as following:

$$W_x = \begin{pmatrix} K_h(X_i - x) & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_h(X_n - x) \end{pmatrix}$$

Now the estimates of the $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_d)$ can be obtain using the following equation:

$$\hat{\beta} = (X_x^T W_x X_x)^{-1} X_x^T W_x Y$$

We know that the estimator $\hat{y} = f(x)$ is the intercept coefficient β_0 of fitted estimate. We can calculate this $\hat{f}$ by simply multiple the estimated $\hat{\beta}$ by $e_1^T = (1, \underbrace{0, \dots, 0}_{(d-1)\text{-times}})$ which is the first element of the basis of $d + 1$ dimensional vector space where d is degree of the polynomial in (2.2) . Therefore we can have the following:

$$\hat{f}(x) = e_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x Y$$

More information on this topic can be found in (Martinez & Martinez, 2007).

2.2 Penalized spline

In this family of estimators instead of using a single polynomial for whole the data we can use different polynomials for different parts of the data. For example:

$$Y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \varepsilon_i & \text{if} \quad x < \phi \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \varepsilon_i & \text{if} \quad x \geq \phi \end{cases}$$

where ϕ is called knot. On the boundaries between the regions, polynomials should be continuous but it is more common to use polynomial with available derivatives up to order two. In general, a linear spline with knots ϕ_k $k = 1, \ldots, k$ is a piece-wise polynomial continuous at each knot and this means that the set of all linear spline with fixed knots is a vector space. Therefore, a basis can be chosen for it. In the following we will introduce two common basis for this vector space. A cubic spline is a piece-wise cubic polynomial with continuous derivatives up to order 2 at each knot, we can write the equation as the following

$$y = \sum_{m=1}^{K+4} \beta_m h_m(x) + \epsilon$$

where

$$h_k(x) = x^{k-1}, \quad k = 1, \ldots, 4$$

and

$$h_{k+4} = (x - \phi_k)_+^3, \quad k = 1, \ldots, K$$

. where $(x - \phi_k)_+ = x - \phi_k$ if $x > \phi_k$ and $(x - \phi)_+ = 0$ otherwise In general, *order-M* spline is defined as below

$$h_k(x) = x^{k-1}, \quad k = 1, \ldots, M.$$

$$h_{k+M}(x) = (x - \phi_k)_+^{M-1}, \ldots k = 1, \ldots, K.$$

$$Y_i = \begin{cases} (x - \phi_k)_+^{M-1} = (x - \phi_k)^{M-1} & \text{if} \quad x > \phi_k \\ (x - \phi_k)_+^{M-1} = 0 & \text{otherwise} \end{cases}$$

However, in practice cubic spline is the highest order we use. Another class of basis that has been frequently used is B-splines(Burden, Faires, & Reynolds, 2001), they were introduced in 1946 by I.J.Schoenberg (Schoenberg, 1946) and in 1972, Carl DE Boor (De Boor, 1972) described recursion formula for evaluation with high computational stability. Now let $\phi_0, \phi_1, \phi_2, \ldots, \phi_k$ be the knots and sorted in a non-decreasing order.

The element of the basis will be the following:

$$B_{i,n}(x) = \begin{cases} 1 & \text{if} \quad \phi_i \leq x \leq \phi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, \ldots, (K + 2M - 1)$ The term $B_{i,m}$ can be calculated by:

$$B_{i,m} = \frac{x - \phi_i}{\phi_{i+m-1} - \phi_i} B_{i,m-1} + \frac{\phi_{i+m} - x}{\phi_{i+m} - \phi_{i+1}} B_{i+1,m-1}(x)$$

where $i = 1, \ldots, K + 2M - m$. If the number of knots is large overfitting may occur so to avoid over fitting penalized spline were introduced. The penalized spline is a function which minimizes the following equation:

$$RSS(f, \lambda) = \Sigma_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int (f''(t))^2 dt \tag{2.4}$$

where λ is fixed and called smoothing parameter and RSS is called penalized residual sum of squares. The first term is the usual term for measuring the goodness of fit and the second term penalizes curvature of the function f. A larger λ will result in a less flexible and smoother curve. In contrast, a smaller λ will result in more a flexible model and more wiggly curve.

2.3 First generation wavelets

Wavelets and Wavelet transforms were first appeared in the mathematical literature around 30 years ago (Grossmann & Morlet, 1984). Continuous wavelet transform was created by modifying the windowed Fourier transform which has been presented below:

$$F_h[f](t, \omega) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(y) \overline{h(y - t)} e^{-i\omega y} dy$$

where $f \in L_2(\mathbb{R})$ is a time-continuous signal and $h \in L_2(\mathbb{R})$. We merge The window function h and the Fourier transform component $e^{i\omega y}$ into one window function which we call ψ and has the property of being scale-able(Jansen & Oonincx, 2005),(Apostol, 1974). After this modification we will end up with the continuous wavelet transform:

$$W_\psi[f](a,b) = \frac{1}{\sqrt{a}} \int_{\mathbb{R}} f(t)\overline{\psi(\frac{t-b}{a})}dt$$

which is a wavelet function $\psi \in L_2(\mathbb{R})$, $+\infty > a > 0$ is called the *scaling* parameter and $+\infty > b > 0$ is *localization* parameter and the wavelet transform continuously depends on them (Jansen & Oonincx, 2005). These properties correspond to the two desirable features of the wavelets. *Localization* mean that the discontinuities in the regression function $f(x)$ will only effect the $\psi(x)$ near it. This will give us the ability to detect localized pattern of the data. The *scaling* feature means the the domain of each $\psi(x)$ can be scaled separately. This means that we could 'zoom in' and analyze the regression function at a set of desired scales.

We can discretize the wavelet transform by restricting our domain to a discrete transform. We also replace the double integral by double summation; a common choice for the discrete subset is a dyadic lattice. The use a special case of basis of $L_2(\mathbb{R})$ for transformation which commonly has the following form

$$\{\psi(2^j t - k) | j, k \in \mathbb{Z}\}$$

Therefore, the reconstruction of the signal would be of the following form and should satisfy the following conditions:

$$m||\alpha||_2^2 \leq ||f||_2^2 \leq M||\alpha||_2^2$$

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \alpha_{j,k}\psi_{j,k}(t)$$

where α is a double indexed sequence that satisfies the equation above and $\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$ and $+\infty > m, M > 0$ this guarantees the f to be bounded. It can be shown that the computational complexity of this method is $\mathcal{O}(N)$ compared to the widely used fast Fourier transform which is $\mathcal{O}(N \log N)$ which adds the benefits of faster convergance(Jansen & Oonincx, 2005). In estimation we apply the wavelet transform to the data and get the wavelet coefficient. The coefficient that are below the error threshold will be dropped. Then the reverse transformation is applied to get the fitted value.

The are some problems surrounding this issue. First, in practice the number of observations should be a power of two $n = 2^l$ for some $l \in \mathbb{N}$. However this is not the major issue and it can be tackled rather easily. The major problem is that the observed data has to be on a regular spaced grid such as $t_i = \frac{i}{n}$. One of the approaches to solve this problem is to interpolate the data to a regularly spaced grid which has benn incorporated into the R package *Wavethresh*(Nason, 2016). This may seem a good idea but in two dimensional cases it has proved to be computationally intensive and not useful in practice(Herrick, 2000). This has encouraged scientists to develop new techniques called "second-generation Wavelets" to overcome this problem. In the next section we will discussed this in more detail.

2.4 Second generation Wavelets

In previous section we introduced the first generation wavelet method and we listed some prominent features of it. We talked about the *localization* and *scalability* of the wavelets and its relatively fast computational efficiency. This qualities give this method a remarkable place among the regression estimation methods. We also noted that the main disadvantage of this method is its reliance on Fourier transformation which make it restricted to an equally-spaced grid. While there are solution such as interpolating data on to an equally spaced grid and performing the wavelet transformation before interpolating it back to actual grid, this solutions have been proven to be computationally expensive.

Consequently a new approach called *second generation wavelets*(Sweldens, 1998) has been developed in past two decades which not only does not rely on Fourier transforms for computation but also contains all the good properties of the wavelet method. The idea behind this new method is called *Lifting schemes*(Sweldens, 1996).

We wish to construct the lifting scheme and briefly explain how it works. Suppose $\lambda_{0,k} = f(k)$ for $k \in \mathbb{Z}$. We like to be able to extract the information in the data with smaller sub-samples but with larger distance between elements. In other words, our aim is to capture the information in fewer coefficients; However, this may not be achievable and in that scenario an approximation with acceptable margins of error is plausible. We reduce the number of coefficients by dividing them into two separate sequences of even and odd samples using the definition below;

$$\lambda_{-1,k} := \lambda_{0,2k} \quad \text{for} \quad k \in \mathbb{Z}$$

we need to record the information that will be lost from $\lambda_{0,k}$ to $\lambda_{-1,k}$ in this procedure. This difference will be recorded in $\gamma_{-1,k}$ and we call them the wavelet coefficients. These coefficient will be calculated by averaging two adjacent even elements and subtracting from the odd sample which can be seen in the following:

$$\gamma_{-1,k} = \lambda_{0,2k+1} - \frac{\lambda_{-1,k} + \lambda_{-1,k+1}}{2}$$

and like the first generation wavelet methods the coefficients that are below the threshold we will ignored. In other words the wavelet coefficient are measuring the magnitude in which the function is different from being linear.This method does not depend on the actual distance between the point of the grid so it is not restricted to a dyadic or equally spaced grid. The package that has been used in this project is `CNLTreg`(Nunes & Knight, 2018) which has been written by the author of the article(Hamilton et al., 2018). In this article they have presented a new Complex-value approach to second generation wavelet, the lifting scheme has been constructed in two branches. One the real-value branch and imaginary branch.

Chapter 3

Methodology

3.1 Simulation

In this project we want to test our estimators on a signal whose function has been constructed on a non equally-spaced grid. The ultimate goal is to calculate the mean square error and mean absolute error associated which each estimator. This will give us a relative measure to compare the estimators and their strengths and weaknesses plus the possible conditions that will force them to fail to address the variation in the data correctly. To generate such data first we will obtain a random sample of size n from a probability distribution. First, we will use $X \sim \text{unif}(0, 1)$ which will provide us with a grid which is not skewed to either left or right. Then we will plug in this data in a function such as Quadratic function, Cubic function, Sinusoidal function, Cosinusoid function to derive the Ys.

$$f_1(x) = x, \quad \text{Linear}$$

$$f_2(x) = x^2, \quad \text{Quadratic}$$

$$f_3(x) = x^3, \quad \text{Cubic}$$

$$f_4(x) = sin(x), \quad \text{Sinusoid}$$

$$f_5(x) = cos(x), \quad \text{Cosinusoid}$$

In the figure 3.1 **A** is the *Linear* function, **B** is the *Quadratic* function, **C** is the *Cubic* function, **D** is *sinusoidal* function, **E** is *cosinusoidal* function and **F** is *Exponential* function. In addition to these basic functions the generated grid will be used to generate

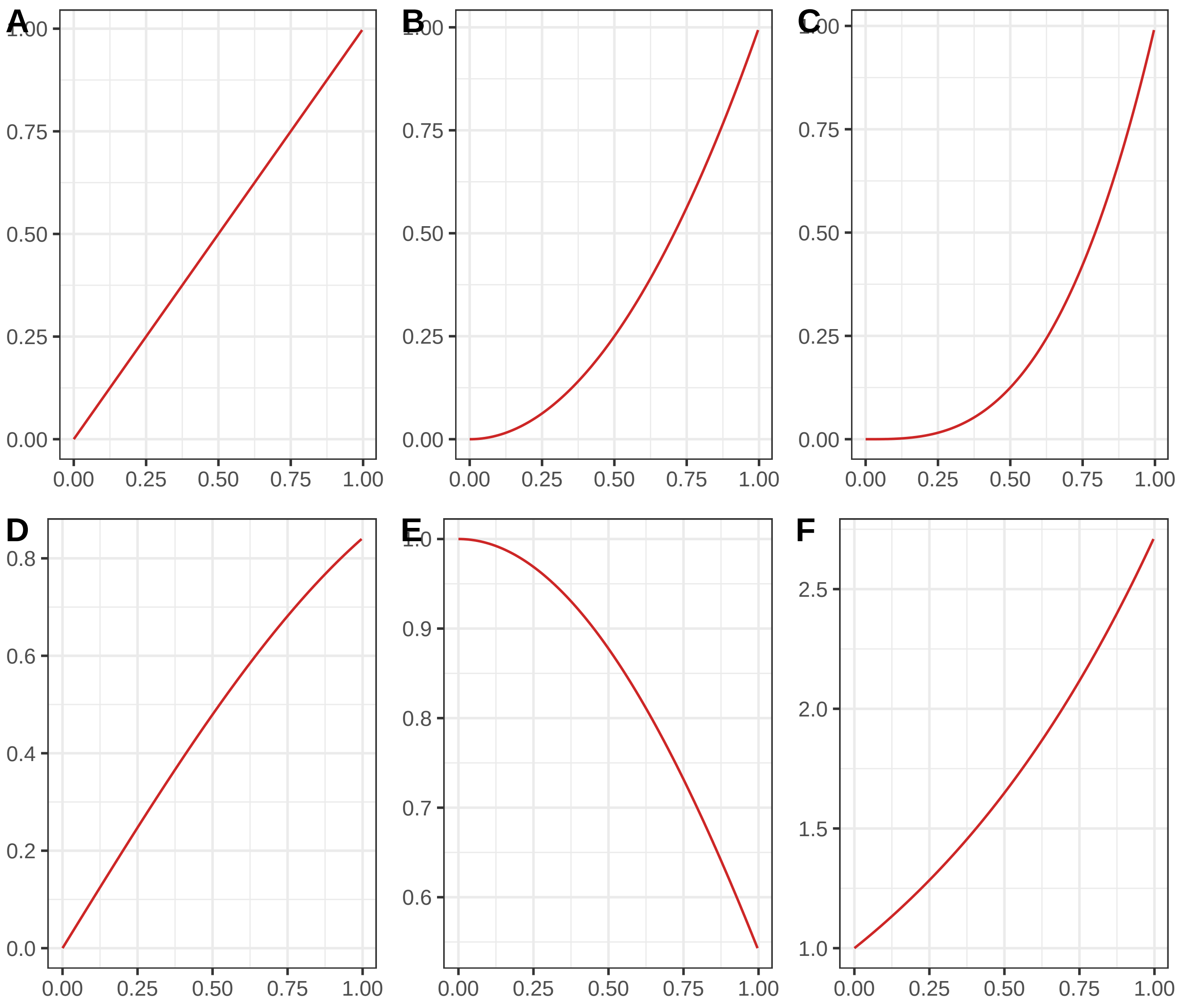

Figure 3.1: Plot of Basic functions analysed in this project

signals such as linchirp, Doppler, heavisine and the Mishmash. Definition of these signals is not simple; therefore, the plot of these signals has been provided in figure 3.3.

Back to our simulation, after generating the grid from $X \sim \text{unif}(0,1)$ the same procedure will be carried out for the data generated from a left skewed mixture of $\text{beta}(2,1)$ and $\text{unif}(0,1)$. And for the right skewed mixture of $\text{beta}(1,2)$ and $\text{unif}(0,1)$. These distributions have been generated using the regularized beta incomplete beta function from the `zipFR` package(Evert & Baroni, 2007). The density plot of these distribution has been provided below: We have chosen these distribtions to make sure the generated grid point do start close to zero in a left-skewed case and do finish near one in the rightskewed case. In other words we wanted to have a grid which contain point from the entire $(0,1)$ interval and this is important because the true value f

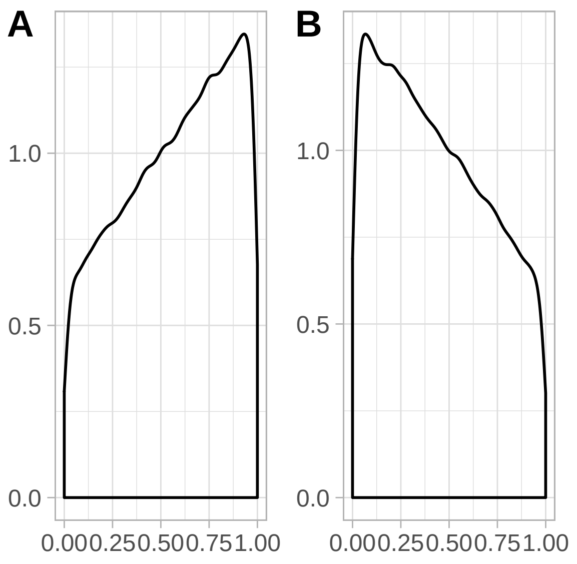

Figure 3.2: Density plot of Left and Right skewed distrtibution used for generating grid

that we want to compare our estimates $\hat{f}$ which is constructed from the previously mentioned functions and the equally spaced grid which starts from zero and goes all the way through one. It is important to try both left and right skewed grids and the reason for this is the non symmetric shape of signals. For instance, Doppler signals are more dense in the left hand side close to zero, so if the sample contains a small number of point on the right hand side it is not hard to see why some estimators will struggle to came up with a good estimate.

The signals in this project have been generate using the `make.signal2` in the `CNLTreg` (Nunes & Knight, 2018) package in **R** which is a modified version of `make.signal` in **S-plus** and has the ability to generate signals on an specified grid. In addition, the plot in this project has been created using the `ggplot2` package (Wickham, 2016) and the `cowplot`(Wilke, 2019) has used to put the `ggplot` generated plots into a panel. In figure 3.3 **A** is `Lincharp`, **B** is `Mishmash`, **C** is `Heavisine` and **D** is `Doppler`. Moreover, noise will be generated from a normal distribution with zero mean and a variance signal-to-noise ratio and will be added to the data. We will change the signal-to-noise ratio to observe the behavior of the estimators see (3.2). The estimated values from the estimators will be recorded.

In this project four different estimators have been used to estimate the regression function. First, for *first generation wavelet* the functions from `wavethresh` package. First we use the function `wd` which performs the wavelet discrete transform and then use

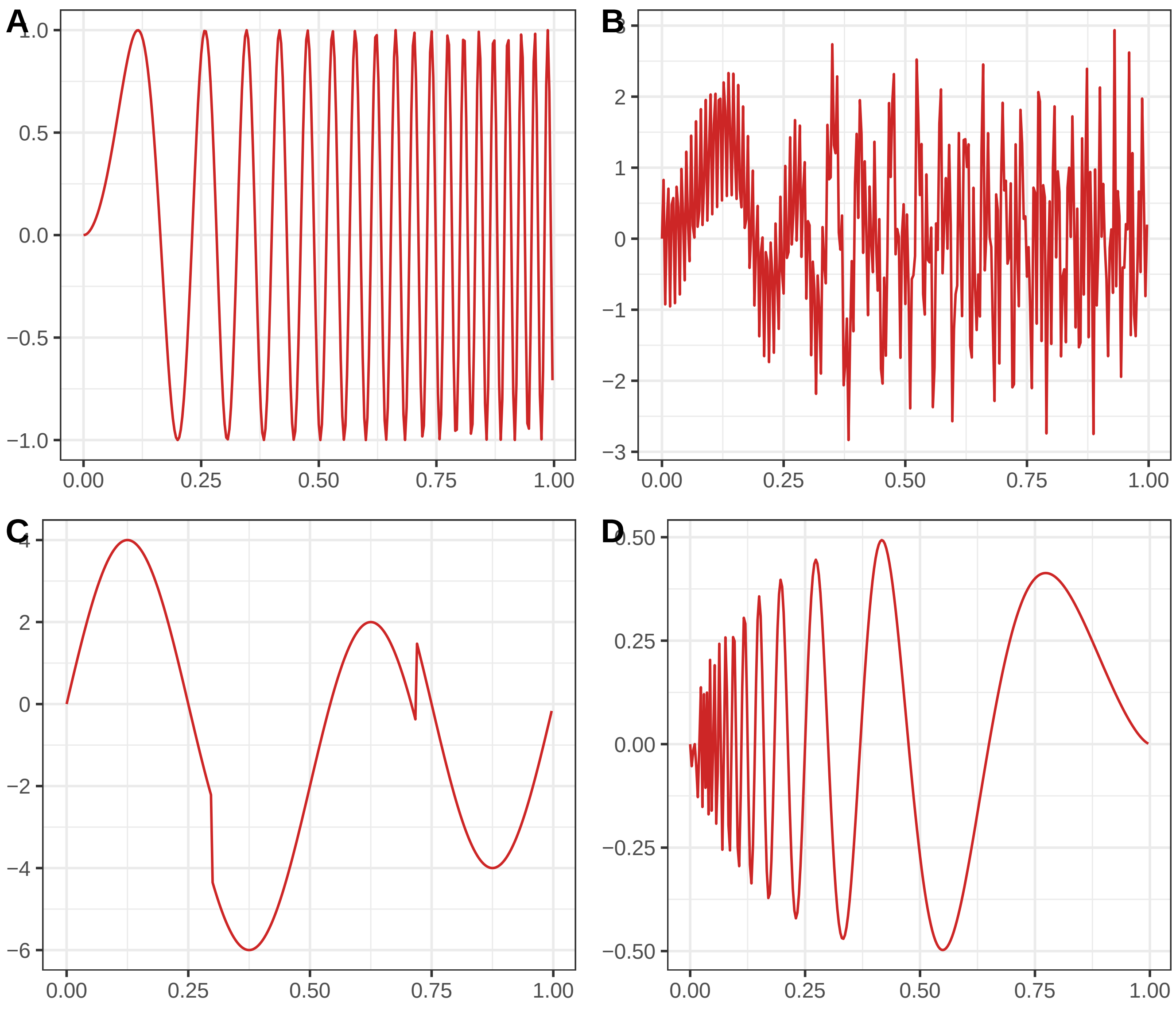

Figure 3.3: Plot of signals analysed in this project

`threshold` function drop the coefficent below the threshold and then we use function `wr` to perform the reverse wavelet trtansform. For these functions the parameters where kept as default. Second, for the *second generation wavelet* the regression function from the `CNLTreg` package (Nunes & Knight, 2018) has been used. This function gets takes the signal and grid value as argument returns the estimates. Parameter P in this function is the number of times that the algorithm gets applied to the data then it averages obtained estimates to obtain the final estimates. Moreover, selecting a P larger than 2 is computationally intensive and according to our tests, in our does not significantly improve the MSE, thus, we use P=2. Third, for *kernel smoothing regression* `np` package (Hayfield & Racine, 2008) has been used, this package contains all kind of nonparametric methods. The regression type for the `np` is chosen to be local constant and the bandwidth selection has been done by Kullback-leibler cross-validation(Hurvich, Simonoff, & Tsai, 1998). Fourth, for *spline* the `mgcv` package was used (Wood, 2003). This package has different functions built in to it including spline smoothing and generalized additive models. In this project to choose the optimal number of knots for penalized B-splines we have used the function gam.check which is provided in the package. The optimal K was 60 for all the signals. This is because the number of knots should be large enough to avoid underfitting. However, if the number of knots is more than enough it would avoid overfitting by penalizing term in equation (2.4).

3.2 Calculating MSE

Our purpose is to estimate the estimate f in equation below:

$$Y_{ij} = f(t_j) + \epsilon_{ij}, \quad \epsilon \sim N(0, \sigma^2) \tag{3.1}$$

In (3.1) we know that $f(t)$ which is the true value of the function of interest. We obtain it by pluging an equally space grid with desirable amount of time points in the function. Furthermore, σ is calibrated by the following equation with respect to signal-to-noise-ratio r:

$$\sigma^2 = \frac{Var(f(t_i))}{r}, \quad i = 1, \ldots, M \tag{3.2}$$

Where t_i is a fixed grid of lenght M, therefore $f(t_i)$ is a vector of lenght M and we calculate the variance of it. In each iteration we calculate the estimate for each time point and record it. Therefore, after N iterations we have N estimate of length M for the signal of interest. We have to note that for the estimators that take the unequal-spaced grid as an argument we have to interpolate the estimated value to a equal-space grid. This will allow us to compare different estimators by calculating the MSE. Now we will discus how to calculate MSE. The below formula will be used to calculate the MSE:

$$\begin{aligned}
MSE = \mathbb{E}(\hat{f}, f)(t) &= \mathbb{E}(\hat{f}(t) - f(t))^2 \\
&= \mathbb{E}[\hat{f}(t) - \mathbb{E}(\hat{f}(t))] + \mathbb{E}[\hat{f}(t)) - f(t)]^2 \\
&= \mathbb{E}[\hat{f}(t) - \mathbb{E}(\hat{f}(t))]^2 + \mathbb{E}[Bias(\hat{f}(t), f(t))]^2 \\
&= Var(\hat{f}(t)) + Bias^2(\hat{f}(t), f(t))
\end{aligned}$$

In the above formula $f(t)$ is the true value of the function or signal f at point t which we wish to estimate. To obtain true value first we generate a equally space grid which is effectively a sequence of numbers with constant increment. Then we plug it in to our function and now we have the true values for each time t. On the other hand, $\hat{f}(t)$ is a the estimate that we get from our estimator. Using the same values we also use the following formula to calculate the Mean absolute error:

$$MAE = \mathbb{E}(|\hat{f}(t) - f(t)|)$$

. We will use MAE because it is considered to be an alternative for MSE(Friedman, Hastie, & Tibshirani, 2001). As mentioned above after recording the estimates with length of M since the grid is of length M therefore the estimates would have the same length and associated interpolated values would have the same length too. Since we are repeated the procedure for N time we will end up with a dataset of interpolated estimated values which is a $M \times N$ matrix. Although we mentioned interpolation is not good practice, we do it only for purpose of calculating the *MSE* and *MAE*. To calculate the bias of the estimates at point t we will take the average of the each row of the data set and subtract it from the true value of the regression function on that point. We calculate the variance of estimator at each point by simply calculating variance of each row of the data. Then we can calculate *MSE* by the this formula $MSE = Bias^2 + Var$. Then we can calculate $MSIE$ by calculating the mean of MSE of all points.

Chapter 4

Results

4.1 Visual Pattern

In this chapter we will illustrate the result of the simulations. First we will start by showing some plots of different scenarios which in the estimations will be tested. The following graphs shows the scenario which the data of length 256 has been generated from a uniform distribution to make random grid is effectively non equally spaced and the curve $f(x) = 3x^3 + 4x^2$ is the function we want to estimate and the noise has been added to a signal-to-noise-ratio of 5. This curve has been chosen simply for demonstration purposes and we are primarily interested in estimating the function listed previously. Since this is a smooth function and the noise has been generated from a uniform distribution we expect that all the estimators could handle it well.
As expected, figure 4.2 show a good fit because there is not that much of variation in the data. Since the spline is less flexible than the other methods there is less variation in the spline estimated curve but obvious more biased and it is smoother. In the next graph we will move on to a more challenging curve to estimate. Just like the previous example the data has been sampled from uniform distribution with length 256 and the ratio for noise is the same. The curve that we are keen to estimate is the Doppler signal. We showed this signal in the previous section and it has the property of this has a considerable amount of variation close to zero and as we move away from zero the variation decreases. This property will make the estimation hard for the less flexible estimator. Moreover, non equally spaced nature in the data should cause the first generation Wavelet to struggle. According to the graphs 4.4 *spline* estimator (D)

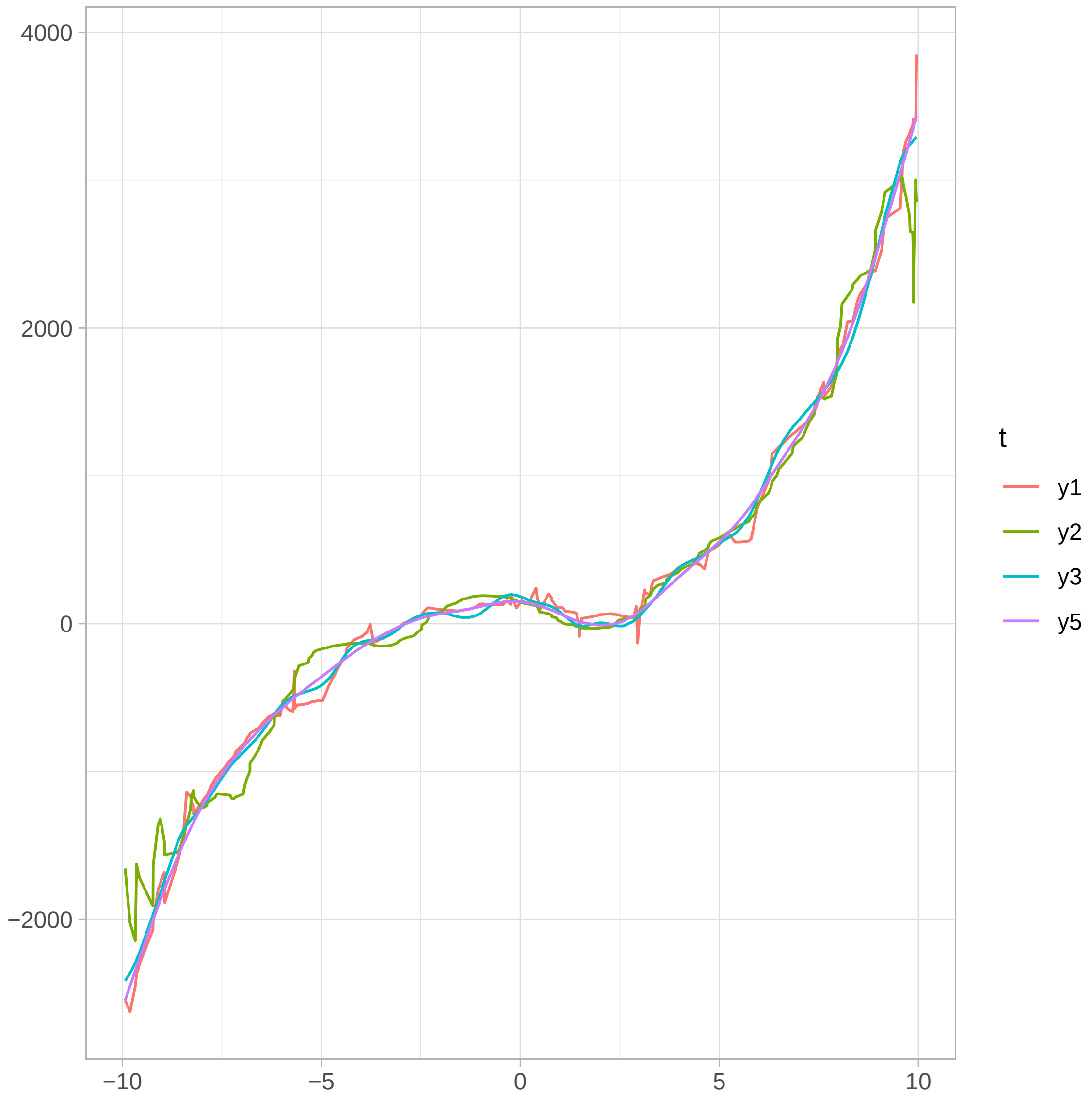

Figure 4.1: Estimate of the the curve $f(x) = 3x^3 + 4x^2$

could not capture the variation f when x is close to zero and it has avoided the peaks and it is biased. *First generation Wavelet* estimator(B) has explained some variation close to zero and it is similar to spline in the essence that struggles to capture the peaks in the data. In the next example the signal remains the same but the data will be generated from rigght-skewed to make it right-skewed. For this particular curve because the first half has more variation, having more data there will make it rather easier to estimate. As expected the same trend has happened in the graph 4.5 *second generation Wavelet*(A) and *Kernel smoother*(C) estimators have done a good job but the other two are struggling. In next scenario we are going to generate the data from a left-skewed distribution and this means the first will have less data points close to zero. Therefore, we expect less flexible estimators to estimate the neighborhood around the

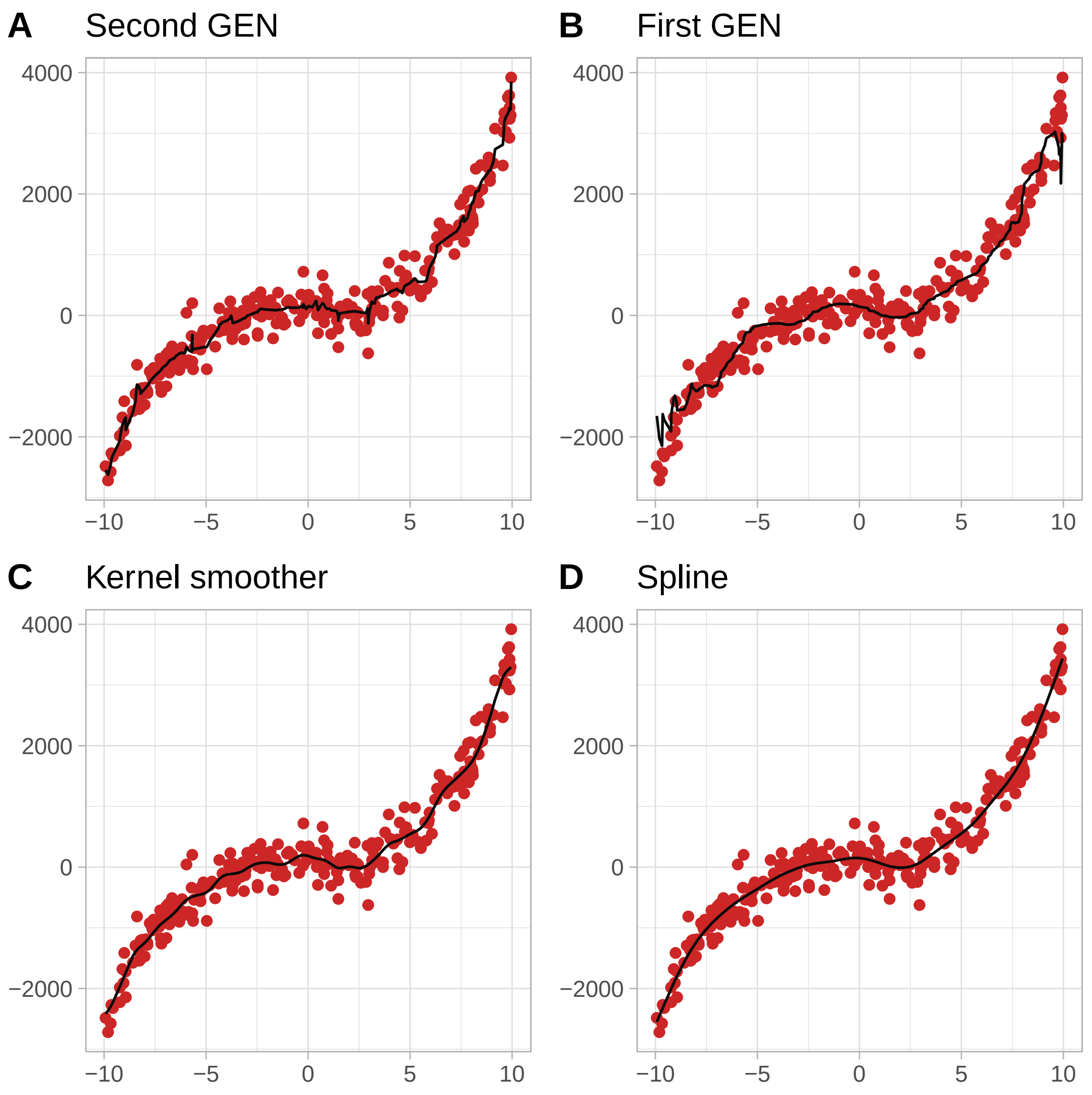

Figure 4.2: Estimate of the the curve $f(x) = 3x^3 + 4x^2$

zero. The figure 4.6 shows almost all estimators are biased near biased close to zero. The first generation has done worse and still far from being perfect it has avoided most of the peaks. The other three etimators have done a good job at capturing most of the variation and the peaks in the data. This result indicated as we mentioned before the second generation estimators is most of the time as good as other commonly used nonparametric estimators. However, we wish test this to simulated the data multiple time and calculated *MSE* and *MAE* to investigate this in a more precise way.

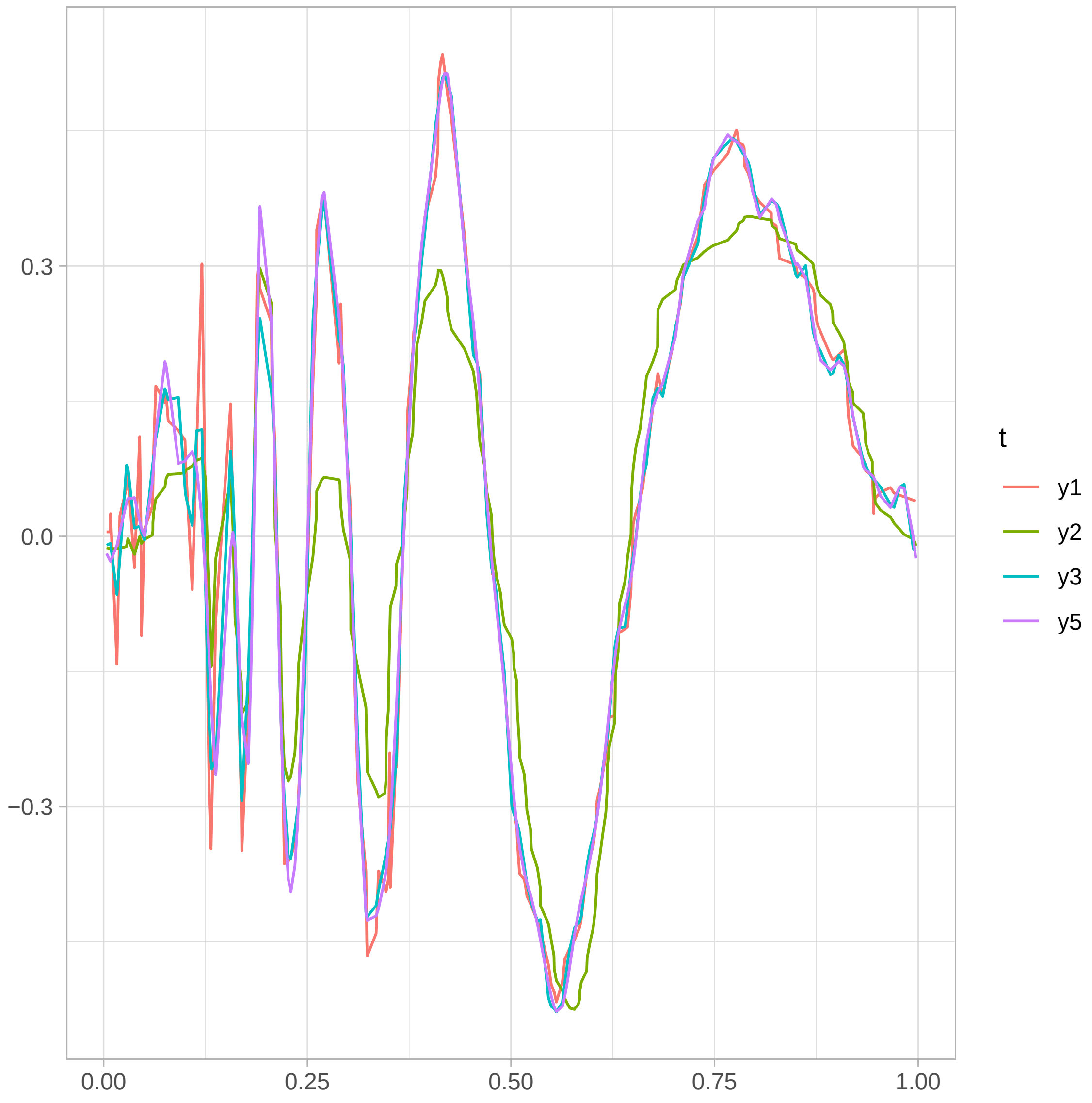

Figure 4.3: Estimates of Doppler signal on a uniforn grid

4.2 Basic functions

In this part we will present the results of simulation on some familiar functions including identity, quadratic, cubic, sine, cosine and exp. For this simulation data has been generated from unif$(0, 1)$ we will run the simulation for 50 time and the grid length will be 256. The signal-to-noise ratio has been kept 5 for this simulation. The results has been given below:

Table 4.1: MSE of estimates on signal on uniform grid with signal-to-noise-ratio 5 (continued below)

estimators	Linear	Quad	Cubic	Sin
First GEN	1.148	1.078	0.944	0.8078

estimators	Linear	Quad	Cubic	Sin
Second GEN	**0.8524**	0.9862	**0.7771**	0.6987
Kernel smoother	0.8526	0.9864	0.7774	0.6988
Spline	0.8524	**0.9862**	0.7771	**0.6987**

Cos	Exp
0.2671	3.323
0.1898	**2.411**
0.1899	2.411
0.1898	2.411

Table 4.3: MAE of estimates on signal on uniform grid with signal-to-noise-ratio 5 (continued below)

estimators	Linear	Quad	Cubic	Sin	Cos	Exp
First GEN	13.67	13.15	12	11.76	6.597	23.13
Second GEN	**11.86**	**12.64**	**10.46**	10.84	**5.671**	**20.43**
Kernel smoother	11.86	12.64	10.47	10.84	5.671	20.43
Spline	11.86	12.64	10.46	**10.84**	5.671	20.43

The results in 4.1 and 4.3 shows under different conditions the second generation wavelet and kernel smoother and the spline can each be the best estimator based on MSE and MAE, whereas the first generation wavelet has a worse performance. The tables show that this pattern consstently. We continue with another table which represent the results for simulation on the list of signals the parameters and the generated grid is the

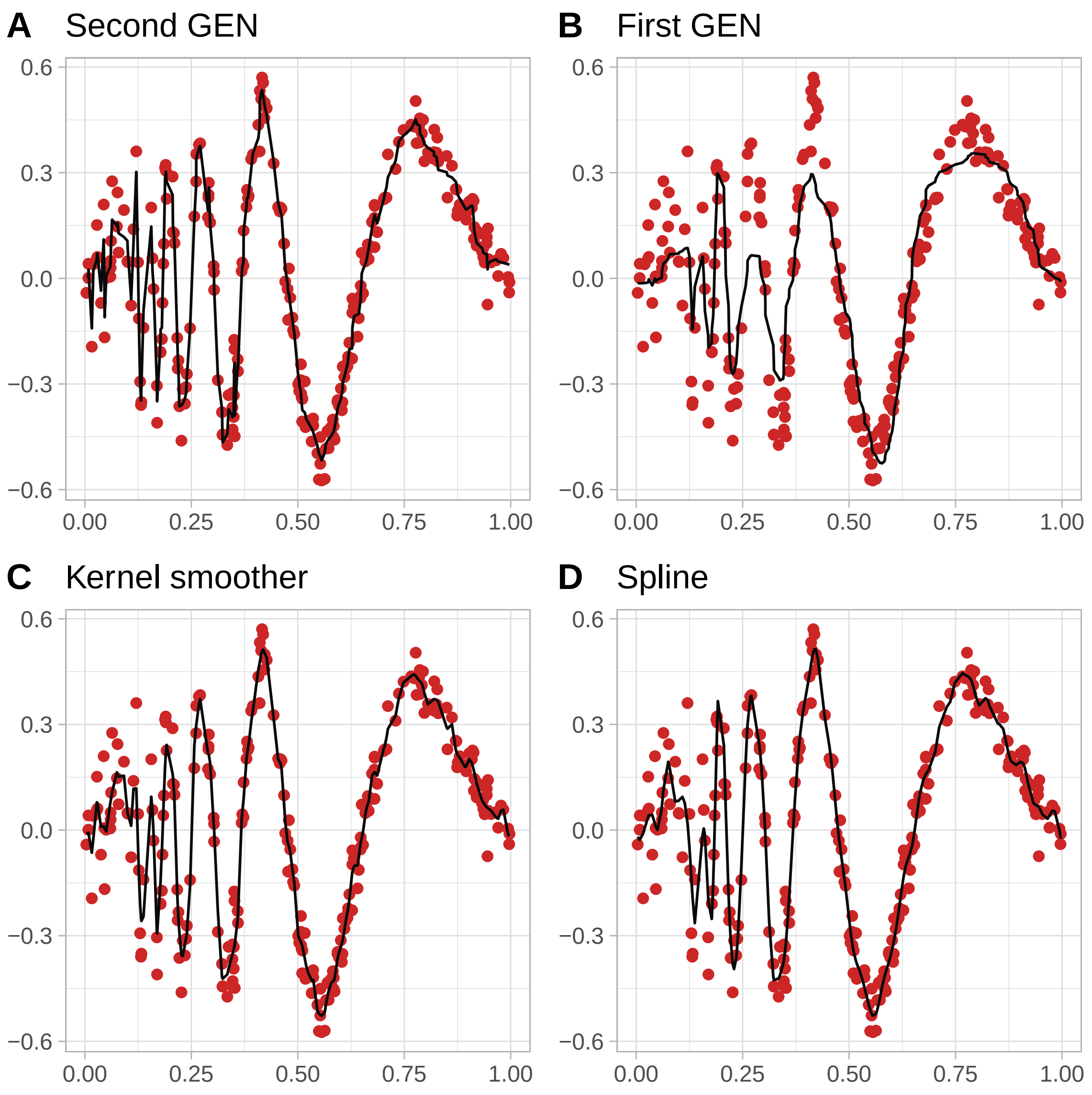

Figure 4.4: Estimates of Doppler signal on a uniforn grid

same as above.

4.3 Signals

4.3.1 Uniform grid

Table 4.5: MSE of estimates on signal on uniform grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	151.7	389.9	141.9	14.73
Second GEN	**10.38**	221.1	**90.6**	**1.554**

estimators	linchirp	mishmash	heavisine	doppler
Kernel smoother	13.9	**220.4**	90.79	1.782
Spline	18.59	288	94.53	2.654

Table 4.6: MAE of estimates on signal on uniform grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	158.6	256.6	156.8	45.31
Second GEN	**34.56**	184.5	**126.2**	**15.05**
Kernel smoother	39.44	**179**	126.3	15.93
Spline	46.96	219.8	128.2	18.09

The results in 4.5 and 4.6 show the first generation is not doing a performing well on the signal linchirp ,whereas the other three are relatively good. For the next signal *mishmash* the patterns gets repeated in both MSE and MAE the seconds generation very close to Kernel smoother. For the *heavisine* the same pattern gets repeated. At last, all the estimators seems to have better performance in performance to the first generation for the *doppler* signal both in MSE and MAE Now we will decrease the signal to noise ratio to 3 for the next simulation but all the parameter will be kept constant.

Table 4.7: MSE of estimates on signal on uniform grid with signal-to-noise-ratio 3

estimators	linchirp	mishmash	heavisine	doppler
First GEN	133.3	399.9	735.2	17.84
Second GEN	**16.88**	**238.4**	277.7	**2.77**
Kernel smoother	18.63	306.5	**277.6**	2.956
Spline	120.8	392.5	291.4	9.612

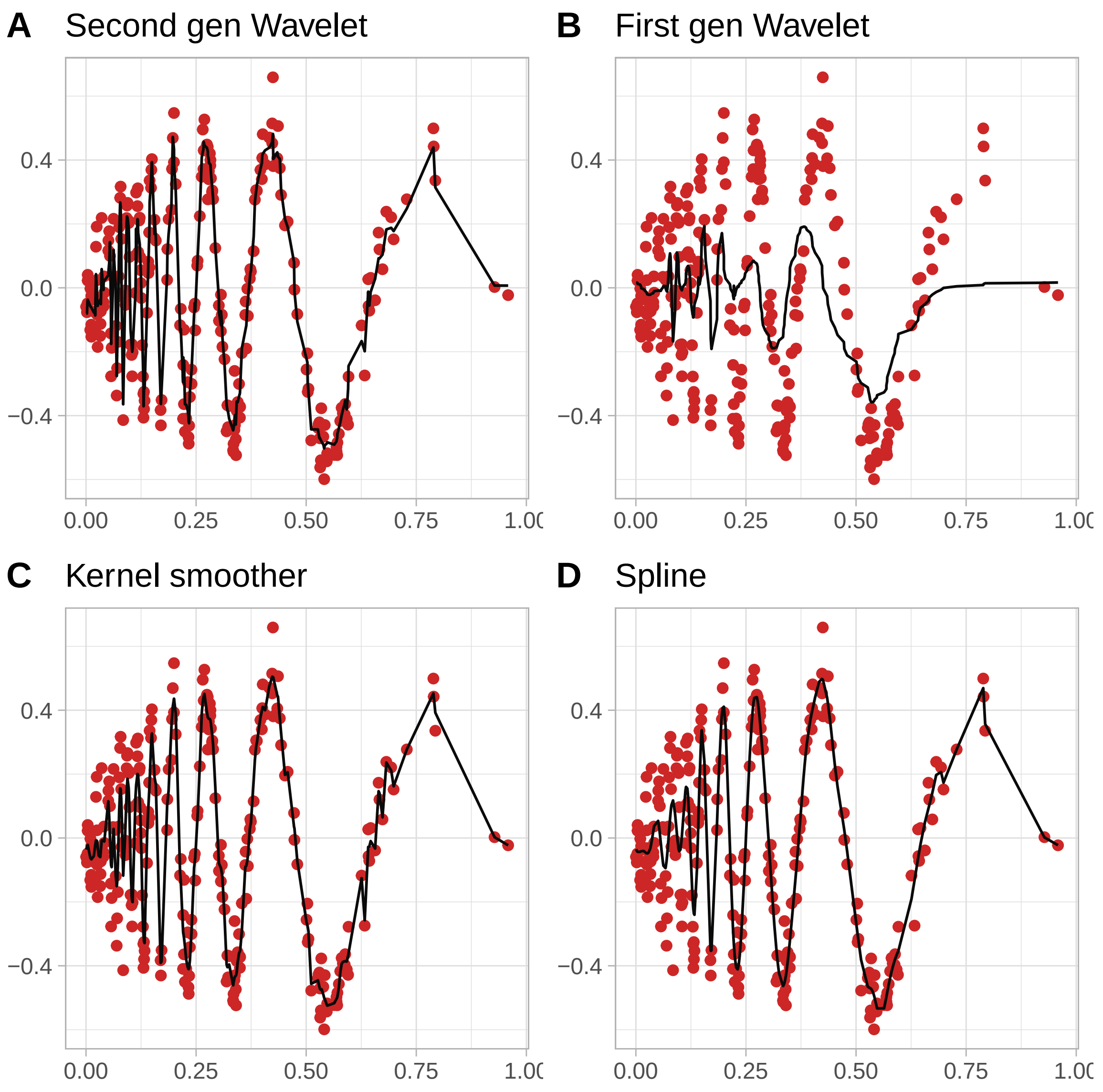

Figure 4.5: Estimate of Doppler signal on a right-skewewd grid

Table 4.8: MAE of estimates on signal on uniform grid with signal-to-noise-ratio 3

estimators	linchirp	mishmash	heavisine	doppler
First GEN	160	258.6	346.1	47.19
Second GEN	**48.06**	**188.8**	212.6	**19.72**
Kernel smoother	50.5	223	**212.6**	20.29
Spline	147.8	257	218.1	37.08

Tables 4.7 and 4.8 show that decreasing the signal-noise-ratio to three which effetively icraeses the variation has triggered an increasing pattern in both MSE and MAE. However, the relation between the performance of the estimators has remained the same. The second generation seems to be the estimator execpt for the heavisine in which Kernel smoother seems to be better. Now we will increase the signal to ratio to

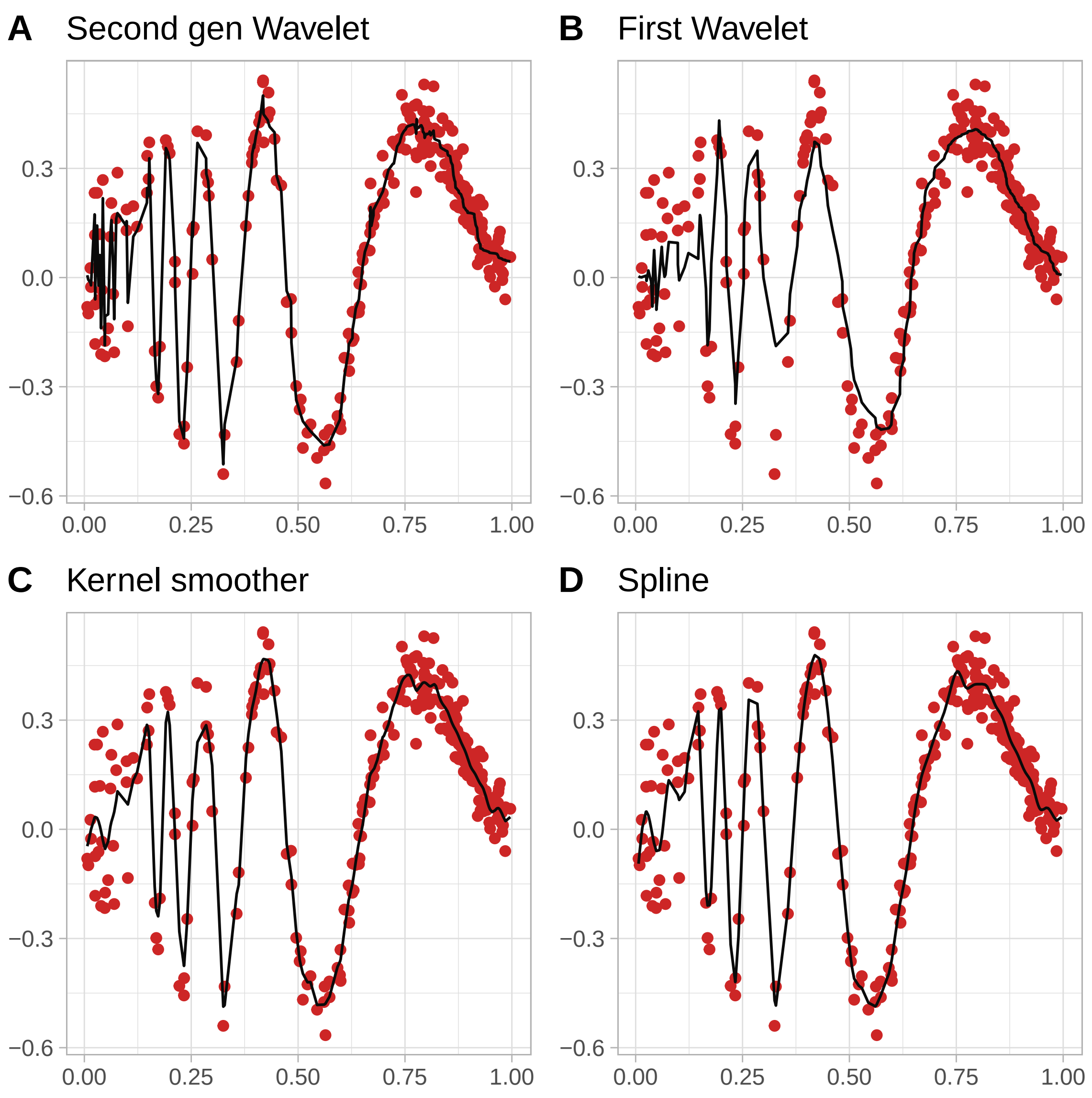

Figure 4.6: Estimates of a Doppler signal left-skewed grid

ratio 5. Now we will move on with testing left-skewed grid in the next section.

4.3.2 Left skewed grid

Table 4.11: MSE of estimates on signal on Left-skewed grid with signal-to-noise-ratio 4

estimators	linchirp	mishmash	heavisine	doppler
First GEN	123.5	382.6	1376	33.87
Second GEN	**7.568**	**209.2**	**169.2**	**1.887**
Kernel smoother	9.662	382.8	169.3	1.95

estimators	linchirp	mishmash	heavisine	doppler
Spline	123.8	386.1	188.4	9.14

Table 4.12: MAE of estimates on a Left-skewed grid with signal-to-noise-ratio 4

estimators	linchirp	mishmash	heavisine	doppler
First GEN	154.3	256	458.8	70.04
Second GEN	**32.77**	**184.7**	**155.1**	**15.67**
Kernel smoother	37.05	258.4	155.2	15.89
Spline	155.5	259.8	165.2	34.96

Based on results in table 4.11 and 4.12 on both MSE and MAE second generation wavelet estimator performs better than the first generation wavelet estimator. In addition, it seems the left skewed nature of the grid had a profound impact on accuracy of the other estimators too. Thereby, second generation wavelet estimator has beat all of them this time. Although, the kernel smoother is a very close in most of case for example the MAEs for example for heavisine. In the next part we will increase the signal-to-noise ratio to 5 to observe the changes.

Table 4.13: MSE of estimates on a Left-skewed grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	144.9	383.2	2168	34.18
Second GEN	**5.093**	**205.5**	**89.83**	**1.933**
Kernel smoother	7.124	210.6	89.83	1.969
Spline	113.1	374.1	106	8.357

Table 4.14: MAE of estimates on a Left-skewed grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	164.6	257.4	572.1	69.55

estimators	linchirp	mishmash	heavisine	doppler
Second GEN	**28.58**	180	120	**15.57**
Kernel smoother	33.16	**176.2**	**120**	16.16
Spline	144.4	254.2	129.8	33.34

Increasing the signal to noise ratio from 4 to 5 has increased the MSEs and MAEs for the heavisine of first generation wavelet method sunbstantially according to tables 4.13 and 4.14. In contrast, both MSE and MAE for the other signals have not change significantly. Overall relations between different estimators is similar to the previous simulation except in the MAE table Kernel smoother is slighty better than second generation for mishmash and heavisine. Now will will increase the signal-to noise ratio to 6.

Table 4.15: MSE of estimates on a Left-skewed grid with signal-to-noise-ratio 6

estimators	linchirp	mishmash	heavisine	doppler
First GEN	145.2	400	3330	34.53
Second GEN	**6.858**	**185.8**	**65.09**	**1.099**
Kernel smoother	9.211	186.3	65.12	1.22
Spline	119.2	353.9	84.77	8.068

Table 4.16: MAE of estimates on a Left-skewed grid with signal-to-noise-ratio 6

estimators	linchirp	mishmash	heavisine	doppler
First GEN	163.1	259.2	735.4	73.14
Second GEN	**31.41**	171.4	**103**	**12.32**
Kernel smoother	35.66	**164.9**	103	12.79
Spline	146.5	246.3	117.1	31.99

The results in 4.15 and 4.16 does not shows that much of difference from the the signal-to-noise ratio 5. The overall relations remain the same between estimators. Comparing the results to the uniformly generating grid MSE and MAE of first generation estimators has gone up for all the signals. However, they are very close to the previous part for other estimators.

4.3.3 Right skewed grid

In this part the grid is chosen to be right-skewed. The below table shows the simulation using signal to noise ratio 4 and the other parameter remain constant.

Table 4.17: MSE of estimates on a right-skewed grid with signal-to-noise-ratio 4

estimators	linchirp	mishmash	heavisine	doppler
First GEN	136.5	447.1	3013	29.92
Second GEN	**24.94**	**236**	**132.7**	**1.511**
Kernel smoother	26.59	251.2	132.9	1.634
Spline	124.9	396.9	154.1	8.329

Table 4.18: MAE of estimates on a right-skewed grid with signal-to-noise-ratio 4

estimators	linchirp	mishmash	heavisine	doppler
First GEN	157.9	279.1	698.7	72.83
Second GEN	**50.98**	185.9	**147.6**	**14.48**
Kernel smoother	52.51	**183.7**	147.8	15
Spline	151.2	260.1	157.6	33.41

In the table 4.17 the results for second generation and Kernel smoother are close. In contrast, the estimates for first generation is substantially larger. The same pattern can be observed in the 4.18. Second generation wavelet method is the best except for the mishmash signal. In the following signal-to noise-ratio will be increase to 5.

Table 4.19: MSE of estimates on a right-skewed grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	144.9	442.7	1390	47.83

estimators	linchirp	mishmash	heavisine	doppler
Second GEN	**7.783**	**224.6**	**118.1**	**1.324**
Kernel smoother	9.761	271.3	118.2	1.561
Spline	113.6	367.5	132.2	8.326

Table 4.20: MAE of estimates on a right-skewed grid with signal-to-noise-ratio 5

estimators	linchirp	mishmash	heavisine	doppler
First GEN	158	268.1	475.3	89.54
Second GEN	**28.57**	**184.6**	**113.7**	**12.58**
Kernel smoother	33.07	210.9	113.8	13.38
Spline	143.9	248.1	125.4	32

According to table 4.19 and 4.20 change in signal-to-noise-ratio does not change the estimates substantially; however, the estimates of the heavisine seems to decrease both in 4.19 and 4.20. In the following able the signal-to-noise-ratio will be increased to 6.

Table 4.21: MSE of estimates on a right-skewed grid with signal-to-noise-ratio 6

estimators	linchirp	mishmash	heavisine	doppler
First GEN	1.91	1.642	2.502	1.711
Second GEN	**0.7898**	**0.5211**	**0.4299**	**0.4406**
Kernel smoother	0.7898	0.5216	0.4299	0.4407
Spline	0.7898	0.5211	0.4299	0.4406

Table 4.22: MAE of estimates on a right-skewed grid with signal-to-noise-ratio 6

estimators	linchirp	mishmash	heavisine	doppler
First GEN	18.46	17.23	21.12	17.93
Second GEN	10.87	**9.334**	8.195	8.89
Kernel smoother	10.87	9.337	8.195	**8.889**

estimators	linchirp	mishmash	heavisine	doppler
Spline	**10.87**	9.334	**8.195**	8.89

For this analysis the results in both 4.21 and 4.22 are substantialy smaller than the previoous part, it seems that the signal-to-noise-ratio noticeably effect the estimates and pushed them down.

Chapter 5

Conclusion

The simulations based on the basic functions did not show considerable difference between estimators. However, *first generation wavelet* estimator had a higher MSE and MAE compared to other estimators in the basic function case. In case of signals with uniform grids and different signal-to-noise-ratio, second generation wavelet estimators had much smaller MSE and MAE than other estimator except for a small number of cases where nonparametric estimators has smaller MSE or MAE.

The results show that for the uniform grid cases and for all tested signal-to-noise-ratio, first generation wavelet had a much larger MSE and MAE compared to second generation method. Changing the grid to the left-skewed and right-skewed did not have any noticeable effect on the MSE and MAE of the second generation wavelet and kernel smoothing, but results an increase in the MSE and MAE of first generation and spline.

In conclusion, the results showed second generation wavelet estimator is a suitable and powerful tool for analyzing the datasets with irregular grid. In would be interesting to study this estimators in the case with more than one independent variable and comparing the results with other estimators and applying methods to time series data which are correlated.

Appendix A

Appendix

The output below shows the codes and the function that has been used in this project:

```r
library(CNLTreg)
library(tidyverse)
library(cowplot)
n_t=300
t = seq(from = 0, to = 1 - 1/n_t, by = 1/n_t)
signals<-list("linchirp","mishmash1","heavisine","doppler")
map(signals,function(y) make.signal2(y,x=t)) %>%
  map(function(y) ggplot(as.data.frame(y),aes(y=y,x=t))+
        geom_line(color="firebrick3")+theme_bw()+
        theme(axis.title.x=element_blank(),
              axis.title.y=element_blank()))%>%
  plot_grid(plotlist = .,
            labels = c("A","B","C","D"))

x1<-sort(runif(256,-10,10))
y<-3*x1^3+4*x1^2
noise<-rnorm(256,mean=0,sd=sqrt(var(y))/5)
y<-y+noise
est1<-CNLTreg::cnlt.reg(x1,y,P=2,
                        LocalPred=AdaptPred,neighbours=1,
                        returnall=FALSE)
estwd <- wavethresh::wd(y)
```

```
estwr <- wavethresh:: wr(wavethresh::threshold(estwd,
                                              by.level=TRUE,
                                              policy="universal"))


npbwrh <-np:: npregbw(y ~ x1, regtype = "lc",
                      bwmethod = "cv.aic")
nprh1 <- np::npreg(npbwrh)



bk <- mgcv:: gam(y~ s(x1,bs="ps",k=60),family=gaussian)
bk<-bk$fitted.values


estfr<-cbind.data.frame(x1,y,
                        y1=t(est1),
                        y2=estwr,
                        y3=fitted(nprh1),
                        y5=bk)
estfr1<- estfr %>% gather(key="t",value="para",y1,y2,y3,y5)




estfr1 %>% ggplot(aes(x=x1,y=para,color=t,group=t))+geom_line()+theme_light()+theme(axi




pr1<-ggplot(estfr,aes(x= x1,y= y))+
  geom_point(color="firebrick3")+
  geom_line(aes(x=x1,y=y1),color="gray4")+
  ggtitle("Second GEN")+theme_light()+ theme(axis.title.x=element_blank(),axis.title.y=


pr2<-ggplot(estfr,aes(x= x1,y=y))+
  geom_point(color="firebrick3")+
  geom_line(aes(x=x1,y=y2),color="gray4")+
  ggtitle("First GEN")+theme_light()+ theme(axis.title.x=element_blank(),axis.ti

pr3<-ggplot(estfr,aes(x= x1,y= y))+
  geom_point(color="firebrick3")+
  geom_line(aes(x=x1,y=y3),color="gray4")+
  ggtitle("Kernel smoother")+theme_light()+ theme(axis.title.x=element_blank(),a

pr4<-ggplot(estfr,aes(x= x1,y= y))+
  geom_point(color="firebrick3")+
  geom_line(aes(x=x1,y=y5),color="gray4")+
  theme_light()+ theme(axis.title.x=element_blank(),
                       axis.title.y=element_blank())+
  ggtitle("Spline")

plot_grid(pr1,pr2,pr3,pr4,labels = c("A","B","C","D"))

sim_npsiguniffunc <-function(f,distribution,n=256,m=50,r){
  # Number of (X, Y) pairs
  library(tidyverse)
  library(CNLTreg)
  library(wavethresh)

  n_t = n # number of points on t grid
  t = seq(from = 0, to = 1 - 1/n_t, by = 1/n_t)
  # Number of simulations
  #approx
  x=matrix(rep( 0, len=n*m), nrow = n)
  y<-matrix(rep( 0, len=n*m), nrow = n)
  ft<-f(t)
  t_2 = seq(from = 1/(2 * n), by = 1/n, length = n)
  ft2 = f(t_2)
  #approx
  x=matrix( rep( 0, len=n*m), nrow = n)
```

```
y=matrix( rep( 0, len=n*m), nrow = n)
ft=f(t)
noise<-rnorm(n,mean=0,sd=sqrt(var(ft))/r)
estwr=matrix( rep( 0, len=n*m), nrow = n)
est1<-matrix( rep( 0, len=n*m), nrow = n)
est2<-matrix( rep( 0, len=n*m), nrow = n)
est3<-matrix( rep( 0, len=n*m), nrow = n)
est4<-matrix( rep( 0, len=n*m), nrow = n)
gridt1<-matrix( rep( 0, len=n*m), nrow = n)
gridy1<-matrix( rep( 0, len=n*m), nrow = n)
gridt2<-matrix( rep( 0, len=n*m), nrow = n)
gridy2<-matrix( rep( 0, len=n*m), nrow = n)
gridt3<-matrix( rep( 0, len=n*m), nrow = n)
gridy3<-matrix( rep( 0, len=n*m), nrow = n)
gridt4<-matrix( rep( 0, len=n*m), nrow = n)
gridy4<-matrix( rep( 0, len=n*m), nrow = n)

mu=matrix( rep( 0, len=n*3), nrow = n)
for (i in 1:m){
  x[,i]=sort(distribution)
  noise<-rnorm(n,mean=0,sd=sqrt(var(ft))/r)
  #generating the y argument
  y[,i]=f(x[,i])+noise[i]
  #wavelet first generation
  estwd=wavethresh::wd(y[,i])
  estwr[,i]=t(wavethresh::wr(wavethresh::threshold( estwd,
                                                    by.level=TRUE,
                                                    policy="universal")))
  #second generation
  est1[,i]<-CNLTreg:: cnlt.reg(x[,i],y[,i],P=2,
                               LocalPred=AdaptPred,
                               neighbours=1,returnall=FALSE)
  #kernelsmoother
  npbwrh =np:: npregbw(xdat = x[,i],
                       ydat = y[,i], regtype = "lc",
                       bwmethod = "cv.aic")
  nprh1 = np:: npreg(npbwrh)
  est3[,i]=fitted(nprh1)
  #Spline-
  bk <- mgcv:: gam(y[,i]~ s(x[,i],bs="ps",k=60),
                 family=gaussian)
  est4[,i]<-bk$fitted.values

  gridapprox1<-makegrid(x[,i],est1[,i])
  gridt1[,i]<-gridapprox1$gridt
  gridy1[,i]<-gridapprox1$gridy
  gridapprox3<-makegrid(x[,i],est3[,i])
  gridt3[,i]<-gridapprox3$gridt
  gridy3[,i]<-gridapprox3$gridy
  gridapprox4<-makegrid(x[,i],est4[,i])
  gridt4[,i]<-gridapprox4$gridt
  gridy4[,i]<-gridapprox4$gridy
}

#mse of first generation

bias_firstgen=rowMeans(estwr)-ft
varest_firstgen=apply(estwr,1,var)
mse_firstgen=varest_firstgen+bias_firstgen^2
mae_firstgen=rowMeans(abs(estwr[,1:m]-ft2))



bias_secondgenp2= rowMeans(gridy1) - ft2
varest_secondgenp2=apply(est1,1,var)
mse_secondgenp2=varest_secondgenp2+bias_secondgenp2^2
```

```
  mae_secondgenp2=rowMeans(abs(gridy1[,1:m]-ft2))


  bias_np= rowMeans(gridy3) - ft2
  varest_np=apply(est3,1,var)
  mse_np=varest_np+bias_np^2
  mae_np=rowMeans(abs(gridy3[,1:m]-ft2))


  bias_spline= rowMeans(gridy4) - ft2
  varest_spline=apply(est4,1,var)
  mse_spline=varest_spline+bias_spline^2
  mae_spline=rowMeans(abs(gridy4[,1:m]-ft2))



  vareste<-cbind(varest_firstgen,varest_secondgenp2,
                 varest_np,varest_spline)
  biase<-cbind(bias_firstgen,bias_secondgenp2,
               bias_np,bias_spline)
  msee<-cbind(mse_firstgen,mse_secondgenp2,
              mse_np,mse_spline)
  maee<-cbind(mae_firstgen,mae_secondgenp2,
              mae_np,mae_spline)


  return_list<-list(vareste,biase,msee,maee)
  return(return_list)
}
library(ggplot2)
revbetarec<-function(n = 1e5,theta = 0.4,a = 1, b = 2){
  library(zipfR)
  U=sort(runif(n,0,1))


  mix = sample(paste0("component ", 1:2),
               size = n, replace = TRUE, prob = c(theta,1 - theta ))
  X = ifelse(mix == "component 1",  Rbeta.inv(U, a , b ),U)
}
betarec<-function(n = 1e5,theta = 0.4,a = 2, b = 1){
  library(zipfR)
  U=sort(runif(n,0,1))

  mix = sample(paste0("component ", 1:2),
               size = n, replace = TRUE, prob = c(theta, 1 - theta))

  X = ifelse(mix == "component 1", Rbeta.inv(U, a , b ), U)
}

df<-as.data.frame(betarec())
#a<-revbetarec()

p1<- ggplot(df, aes(x=betarec())) +
  geom_density()+theme_light()+
  theme(axis.title.x=element_blank(),axis.title.y=element_blank())
df<-as.data.frame(revbetarec())

p2 <- ggplot(df, aes(x=revbetarec())) +
  geom_density()+theme_light()+
  theme(axis.title.x=element_blank(),axis.title.y=element_blank())
cowplot::plot_grid(p1,p2,labels = c("A","B"))
sim_npsig <-function(f,distribution,n=256,m=50,r){
  # Number of (X, Y) pairs
  library(tidyverse)
  library(CNLTreg)
  library(wavethresh)
  library(pander)
  n_t = n # number of points on t grid
  t = seq(from = 0, to = 1 - 1/n_t, by = 1/n_t)
```

```
x=matrix(rep( 0, len=n*m), nrow = n)
y<-matrix(rep( 0, len=n*m), nrow = n)
ft<-make.signal2(f,x=t)
t_2 = seq(from = 1/(2 * n), by = 1/n, length = n)
ft2 = make.signal2(f,x=t_2)

estwr=matrix( rep( 0, len=n*m), nrow = n)
est1<-matrix( rep( 0, len=n*m), nrow = n)
est2<-matrix( rep( 0, len=n*m), nrow = n)
est3<-matrix( rep( 0, len=n*m), nrow = n)
est4<-matrix( rep( 0, len=n*m), nrow = n)
gridt1<-matrix( rep( 0, len=n*m), nrow = n)
gridy1<-matrix( rep( 0, len=n*m), nrow = n)
gridt2<-matrix( rep( 0, len=n*m), nrow = n)
gridy2<-matrix( rep( 0, len=n*m), nrow = n)
gridt3<-matrix( rep( 0, len=n*m), nrow = n)
gridy3<-matrix( rep( 0, len=n*m), nrow = n)
gridt4<-matrix( rep( 0, len=n*m), nrow = n)
gridy4<-matrix( rep( 0, len=n*m), nrow = n)

mu=matrix( rep( 0, len=n*3), nrow = n)
for (i in 1:m){
  x[,i]=sort(distribution)
  noise<-rnorm(n,mean=0,sd=sqrt(var(ft))/r)
  #generating the y argument
  y[,i]=make.signal2(f,x=x[,i])+noise[i]
  #wavelet first generation
  estwd=wavethresh::wd(y[,i])
  estwr[,i]=t(wavethresh::wr(wavethresh::threshold(estwd,
                                                      by.level=TRUE,
                                                      policy="universal")))
  #change mse
  est1[,i]<-CNLTreg:: cnlt.reg(x[,i],y[,i],P=2,
                                LocalPred=AdaptPred,
                                neighbours=1,
                                returnall=FALSE)

  npbwrh =np:: npregbw(xdat = x[,i],ydat = y[,i],
                        regtype = "lc",
                        bwmethod = "cv.aic")

   nprh1 = np:: npreg(npbwrh)
  est3[,i]=fitted(nprh1)

      bk <- mgcv:: gam(y[,i]~ s(x[,i],bs="ps",k=60),
                    family=gaussian)
  est4[,i]<-bk$fitted.values

  gridapprox1<-makegrid(x[,i],est1[,i])
  gridt1[,i]<-gridapprox1$gridt
  gridy1[,i]<-gridapprox1$gridy
  gridapprox3<-makegrid(x[,i],est3[,i])
  gridt3[,i]<-gridapprox3$gridt
  gridy3[,i]<-gridapprox3$gridy
  gridapprox4<-makegrid(x[,i],est4[,i])
  gridt4[,i]<-gridapprox4$gridt
  gridy4[,i]<-gridapprox4$gridy
}

#mse of first generation

bias_firstgen=rowMeans(estwr)-ft
varest_firstgen=apply(estwr,1,var)
mse_firstgen=varest_firstgen+bias_firstgen^2
mae_firstgen=rowMeans(abs(estwr[,1:m]-ft2))
```

```
  bias_secondgenp2= rowMeans(gridy1) - ft2
  varest_secondgenp2=apply(est1,1,var)
  mse_secondgenp2=varest_secondgenp2+bias_secondgenp2^2
  mae_secondgenp2=rowMeans(abs(gridy1[,1:m]-ft2))

  bias_np= rowMeans(gridy3) - ft2
  varest_np=apply(est3,1,var)
  mse_np=varest_np+bias_np^2
  mae_np=rowMeans(abs(gridy3[,1:m]-ft2))

  bias_spline= rowMeans(gridy4) - ft2
  varest_spline=apply(est4,1,var)
  mse_spline=varest_spline+bias_spline^2
  mae_spline=rowMeans(abs(gridy4[,1:m]-ft2))



  vareste<-cbind(varest_firstgen,varest_secondgenp2,
                 varest_np,varest_spline)
  biase<-cbind(bias_firstgen,bias_secondgenp2,
               bias_np,bias_spline)
  msee<-cbind(mse_firstgen,mse_secondgenp2,
              mse_np,mse_spline)
  maee<-cbind(mae_firstgen,mae_secondgenp2,
              mae_np,mae_spline)

  return_list<-list(vareste,biase,msee,maee)
  return(return_list)
}
msesum<-function(x){
  apply(x,2,sum)
}
```

```
library(tidyverse)
library(pander)
library(kableExtra)
library(knitr)
printmse<-function(data_ms,m,cap,function_name=c("linchirp","mishmash","heavisine
  df<-map_dfc(c(1:n),function(i)map(data_ms[[i]][m],msesum)%>%unlist(use.names=T
  emphasize.strong.cells(cbind( apply(df[,2:(n+1)], 2, which.min),2:(n+1)))
  pander::pander(df,caption=cap, style = "rmarkdown")
}
```

References

Apostol, T. M. (1974). *Mathematical analysis; 2nd ed.* Reading, MA: Addison-Wesley. Retrieved from `https://cds.cern.ch/record/105425`

Burden, R. L., Faires, J. D., & Reynolds, A. C. (2001). Numerical analysis. Brooks/cole Pacific Grove, CA.

De Boor, C. (1972). On calculating with b-splines. *Journal of Approximation Theory*, *6*(1), 50–62.

Evert, S., & Baroni, M. (2007). *ZipfR*: Word frequency distributions in R. In *Proceedings of the 45th annual meeting of the association for computational linguistics, posters and demonstrations sessions* (pp. 29–32). Prague, Czech Republic.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics New York.

Grossmann, A., & Morlet, J. (1984). Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, *15*(4), 723–736.

Hamilton, J., Nunes, M. A., Knight, M. I., & Fryzlewicz, P. (2018). Complex-valued wavelet lifting and applications. *Technometrics*, *60*(1), 48–60.

Hayfield, T., & Racine, J. S. (2008). Nonparametric econometrics: The np package. *Journal of Statistical Software*, *27*(5). Retrieved from `http://www.jstatsoft.org/v27/i05/`

Herrick, D. (2000). Wavelet methods for curve estimation. *PhD Thesis.*

Hsu, D., Kakade, S. M., & Zhang, T. (2011). An analysis of random design linear regression. *arXiv Preprint arXiv:1106.2363.*

Hurvich, C. M., Simonoff, J. S., & Tsai, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(2), 271–293.

Jansen, M. H., & Oonincx, P. J. (2005). *Second generation wavelets and applications.* Springer Science & Business Media.

Martinez, W. L., & Martinez, A. R. (2007). *Computational statistics handbook with matlab.* Chapman; Hall/CRC.

Nason, G. (2016). *Wavethresh: Wavelets statistics and transforms.* Retrieved from `https://CRAN.R-project.org/package=wavethresh`

Nunes, M., & Knight, M. (2018). *CNLTreg: Complex-valued wavelet lifting for signal denoising.* Retrieved from `https://CRAN.R-project.org/package=CNLTreg`

Schoenberg, I. J. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. Part b. On the problem of osculatory interpolation. A second class of analytic approximation formulae. *Quarterly of Applied Mathematics*, *4*(2), 112–141.

Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, *3*(2), 186–200.

Sweldens, W. (1998). The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, *29*(2), 511–546.

Wackerly, D., Mendenhall, W., & Scheaffer, R. L. (2014). *Mathematical statistics with applications.* Cengage Learning.

Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis.* Springer-Verlag New York. Retrieved from `https://ggplot2.tidyverse.org`

Wilke, C. O. (2019). *Cowplot: Streamlined plot theme and plot annotations for 'ggplot2'.* Retrieved from `https://CRAN.R-project.org/package=cowplot`

Wood, S. N. (2003). Thin-plate regression splines. *Journal of the Royal Statistical Society (B)*, *65*(1), 95–114. Retrieved from `https://CRAN.R-project.org/package=mgcv`