Bayesian Additive Modeling

Zad Chin & Jarell Cheong Tze Wen

May 8, 2023

Abstract. This project is our attempt at understanding and implementing additive models from both the frequentist and Bayesian perspective. In Sections 1 and 2, we'll introduce the theory underlying additive models, mostly from the frequentist perspective following Wood [40]. In Sections 3 and 4, we'll look at some recent developments to the Bayesian approach to additive modeling using Gaussian process priors and difference priors. Here, we follow the recent paper by Solonen and Staboulis [37]. Finally, we'll implement all of these methods to three different datasets. Our new contributions are (1) the replication, from scratch, of various figures in Wood [40] and Solonen and Staboulis [37], (2) the full derivation of results which are just stated in Wood [40], and (3) the entirety of Section 5.

This paper is written as a final project for Statistics 220: Bayesian Data Analysis, a class both authors are enrolled in under the instruction of Professor Jun Liu and Teaching Fellow Buyu Lin.

Contents

1 Introduction

Suppose we have n datapoints $\{(x_i, y_i)\}_{i=1}^n$, where x_i has p features (put differently, x_i is p-dimensional). In Statistics 220, we learned about linear models of the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \ldots, n, \tag{1}$$

where $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently. These were the initial types of prediction models to be studied rigorously, and to be used extensively in practical applications [42]. After all, they turned out to be easier to fit than models which were nonlinearly related to their parameters, and the statistical properties of their corresponding estimators were easier to determine as well. However, in many areas of prediction such as financial mathematics, computational biology, medicine, chemistry, and environmental science, departures from linearity could be easily detected through residual and partial residual plots [38]. For example, in plant ecology, there are many examples of response variables like the percentage of pollen which are bimodal or skewed, and thus cannot be adequately modeled by (1) even with transformations of variables or the addition of higher order terms [43]. It is also possible to obtain coefficient estimates which obviously contradict biological reality. Other examples include modeling short and long term impact and risks of air pollution on various unfavorable health outcomes [32], controlling for covariates when mapping spatial distributions of disease occurence and risk [39], and prediction in settings where a large amount of data is available [41].

The primary limitation of the linear models in (1) in these settings arises from their parametric, model-driven nature, resulting in a high degree of dependence on the assumptions of the model rather than focusing on data. The development of additive models (AMs) and supervised machine learning-based algorithms such as random forests are examples of efforts to address this issue. AMs are considered to be an inherently "data-driven" method since data, rather than model assumptions, determines much of the fit and prediction. In this project, we will investigate AMs as direct extensions of linear models due to their high interpretability relative to other machine learning methods and their model flexibility, especially in dealing with highly nonlinear and nonmonotonic relationships between the response and explanatory variables. In fact, all but one of the papers mentioned in the previous paragraph make use of AMs. More specifically, this project has two main goals:

(a) Study, in great detail, the frequentist and Bayesian approaches to AMs.

(b) Implement these approaches, as well as other benchmark models, to data.

The rest of Section 1 is organized as follows. Subsection 1.1 will be where we will define AMs for the first time. In Subsection 1.2, we shall discuss how smoothing, in the context of AMs, is equivalent to imposing a prior belief that true functions are more likely to be smooth than wiggly. We formalize what this means as well as argue why this insight is not enough for a full Bayesian approach to AMs [40]. In Subsection 1.3, we review the existing literature on Bayesian AMs and justify our subsequent narrowing of focus to two specific choices of priors for functions in an AM. We discuss the extension of AMs to Generalized AMs (GAMs), in the same way generalized linear models extend the linear models in (1), in Subsection 1.4. Here, we also discuss the nuances of this extension, namely the steps which need to be taken to extend the methods from the rest of this paper to the general case. In Subsection 1.5, we briefly outline the contents of the rest of the paper.

1.1 Introducing Additive Models

Additive models (and more broadly, generalized additive models) were originally introduced by Trevor Hastie and Robert Tibshirani in 1986 [18]. This framework is based on a rather appealing and simple to understand mental model [26]:

(a) Relationships between the individual predictors and the dependent variable follow smooth patterns that can either be linear or nonlinear.

(b) Estimations of these smooth relationships can be done simultaneously, and then prediction of the dependent variable can be done via addition.

Thus, AMs are a modeling technique where the impact of the predictive variables is captured through smooth functions which, depending on underlying patterns in the data, can be nonlinear. The AM is a model taking on the following form:

$$y_i = \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip}) + \epsilon_i, \quad i = 1, \ldots, n, \tag{2}$$

where $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently; $f_1, \ldots, f_p$ are smooth, nonparametric functions to be estimated from the data. Note that, in the context of regression models, the terminology "nonparametric" means that the shape of the predictor functions are fully determined by the data as opposed to parametric functions that are defined by a typically small set of parameters. This can allow for more flexible estimation of the underlying predictive patterns without knowing upfront what these patterns look like [26]. Observe that AMs could also contain parametric terms as well. Indeed, the model (1) is a special case of the AM where $f_j = \beta_j x_{ij}$ for each $j = 1, \ldots, p$. We follow [26] to explain the advantages of AMs.

In general, there are at least three good reasons to wield AMs: interpretability, flexibility/automation, and regularization. Thus, when a given set of data contains nonlinear effects, AMs provide a regularized and interpretable solution, in stark contrast to other methods, which generally lack at least one of these three features. In other words, AMs strike a nice balance between the interpretable, yet biased, linear model, and the extremely flexible "black box" machine learning algorithms.

Interpretability. When a regression model is additive, the interpretation of some marginal impact of a single variable does not depend on the values of the other variables in the model. Thus, just by looking at the output of the model, one can make simple statements about the effects of the predictive variables that would make sense to a nontechnical person. In addition, an important feature that AMs have is the ability to control the smoothness of the predictor functions. For AMs, one can avoid wiggly, nonsensical predictor functions by simply adjusting the level of smoothness. Put differently, one can impose the prior belief that predictive relationships are inherently smooth in nature, even though the dataset at hand may suggest a more noisy relationship (look at Subsection 1.2 for more details).

Flexibility and Automation. AMs can capture common nonlinear patterns that a classic linear model would miss. These patterns range from "hockey sticks" that occur when there exists a sharp observed change in the response variable to many types of "mountain shaped" curves. Examples of such curves are in Figure 1.

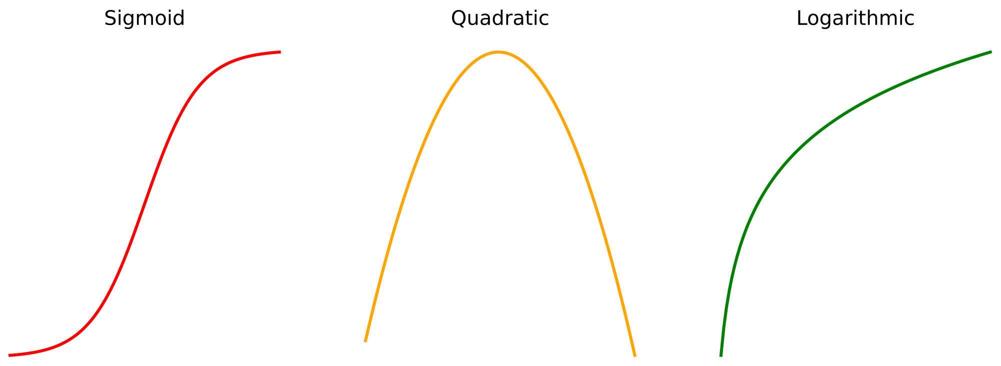


Figure 1: Three examples of "hockey stick" and "mountain shaped" curves.

When fitting parametric regression models, these types of nonlinear effects are typically captured through binning or polynomials. This leads to clumsy model formulations with many correlated terms and counterintuitive results. Moreover, selecting the best model involves constructing a whole slew of transformations, followed by a search algorithm to select the best option for each predictor. This potentially greedy step can easily become computationally infeasible. Nevertheless, this problem does not surface with AMs: predictor functions are automatically derived during model estimation, and one does not need to know upfront which "type" of functions they will need. This not only saves time, but may also assist with discovering patterns which may have been missed with a parametric model. Obviously, it is entirely possible to find parametric functions that "look like" the relationships extracted by an additive model. However, the work to get there is usually tedious, and one does not have 20/20 hindsight prior to model estimation.

Regularization. The AM framework allows for the control of smoothness of the predictor functions to prevent overfitting. Indeed, by controlling the wiggliness of the predictor functions, one can directly tackle the bias-variance tradeoff via explicit control of a smoothing parameter. With this apparatus, one can impose smoother curves, making the curve have more bias (in-sample error) but also less variance. Such curves tend to make more sense in addition to validating better in out-of-sample tests. However, if the curve is too smooth, some important pattern may be missed. Either way, having the ability to tweak smoothness will surely add another dimension of flexibility to the model, making it possible to avoid the sort of cumbersome and unwieldly models which are usually specified for the GLM setting (see Section 3.3.5 of [40] for an example of such tedious specifications).

Challenges With AMs. However, the flexibility and convenience of AMs comes at the cost of two new theoretical problems: it shall be necessary both to represent the smooth functions in some way and to choose how smooth they should be. It is these theoretical considerations that the rest of this paper hopes to investigate.

1.2 The Bayesian Approach to AMs

As is continuously mentioned in Statistics 220, the Bayesian approach to statistical modeling possesses the main advantage of being able to recover the whole range of inferential solutions, rather than a point estimate plus a confidence interval as in the frequentist approach. Consequently, we would like to immediately explore some Bayesian considerations with the AMs we have talked about so far. To do this, we follow Chapter 4 in Wood's textbook [40]. For simplicity, in this section, we will consider a model containing only one function of one covariate. Hence, in other words, our model, given datapoints $\{(x_i, y_i)\}_{i=1}^n$ (with the x_i's here being one-dimensional), will be of the following simplified form that is expressed below:

$$y_i = f(x_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{3}$$

where f is a smooth function and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently. Our first immediate hurdle is the question of how to estimate f. One idea to tackle this hurdle is to represent f in such a way that (3) becomes a linear model, for if we can make this happen, we can then apply the theory of linear modeling to the problem.

This can be done by choosing a basis, defining the space of functions of which f (or a very close approximation to it) is an element. Choosing a basis amounts to choosing some basis functions, which will be treated as completely known: if $b_j(x)$ is the jth such basis function, then f is assumed to have a representation

$$f(x) = \sum_{j=1}^{k} b_j(x)\beta_j, \tag{4}$$

for some values of the unknown parameters, β_j. Substituting (4) into (3) will yield a linear model. After all, writing $f(x)$ as in (4), we have the compact expression

$$y = X\beta + \epsilon, \tag{5}$$

where the vectors y, β, ϵ and the matrix X are defined in the following manner:

$$y = (y_1, \ldots, y_n)^t, \tag{6}$$
$$\beta = (\beta_1, \ldots, \beta_k)^t, \tag{7}$$
$$\epsilon = (\epsilon_1, \ldots, \epsilon_n)^t, \tag{8}$$
$$X_{ij} = b_j(x_i), \quad i = 1, \ldots, n, \quad j = 1, \ldots, k. \tag{9}$$

The two challenges we posed at the end of Subsection 1.1 still stand; namely, we would still have to find suitable basis functions $b_j(x)$, as well as choose the basis dimension k, which clearly controls the "smoothness" of the resulting model.

To tackle the latter problem, the solution proposed in Chapter 4 is to "control smoothness by penalizing wiggliness" (cf. Section 4.2.2 of [40]). Specifically, this means keeping the basis dimension k fixed at a size a little larger than it is believed could reasonably be necessary, and then controlling the model smoothness using a "wiggliness" penalty to the least squares fitting objective. For instance, rather than fitting the model by minimizing the classical penalty $||y - X\beta||^2$, minimize

$$||y - X\beta||^2 + \lambda\beta^t S\beta \tag{10}$$

with respect to β. Here, S is a rank deficient matrix determined by the choice of basis functions $b_j(x)$ (we shall introduce S more formally when we talk about a few examples of actual basis functions in Section 2). The smoothing parameter, λ, controls the tradeoff between smoothness of the estimated $f(x)$ and fidelity to the data (how well the estimate fits the data). Indeed, in the limit $\lambda \to \infty$, we have a straight line estimate for $f(x)$, while $\lambda = 0$ will result in an unpenalized linear regression estimate. Thus, the problem of estimating the degree of smoothness is now the problem of estimating the smoothing parameter λ. After all, given λ, estimation of β is fairly straightforward. This computation is the result below.

Theorem 1.1. *The formal expression for the minimizer of* (10), $\hat{\beta}$, *is given below:*

$$\hat{\beta} = (X^t X + \lambda S)^{-1} X^t y. \tag{11}$$

Proof. This expression can be realized by setting partial derivatives to zero. Set

$$f_{\text{obj}}(\beta) = \|y - X\beta\|^2 + \lambda \beta^t S \beta. \tag{12}$$

Then, noting that $\beta^t S \beta$ is a quadratic form, the relevant partial derivative is just

$$\frac{\partial f_{\text{obj}}(\beta)}{\partial \beta} = -2X^t(y - X\beta) + 2\lambda S \beta. \tag{13}$$

Setting this to zero, cancelling the constant term 2, and rearranging the resulting equation in terms of β, we get, from this simplification, $\hat{\beta} = (X^t X + \lambda S)^{-1} X^t y$. □

The frequentist approach to answering the question of which λ is "best" would be to use cross validation; we cover this approach in Subsection 2.3. However, it is possible to approach this problem from a Bayesian perspective, and it is at this point where we first discover a possible Bayesian interpretation for the additive models that we have studied so far. This interpretation just amounts to noticing that at some level, we introduce smoothing penalties because we believe that the truth is more likely to be smooth than wiggly, and then noticing that we might as well formalize this belief in a Bayesian way. We can achieve this by specifying a prior distribution on function wiggliness, $p(\beta)$. Perhaps the simplest choice is

$$p(\beta) \propto \exp(-\lambda \beta^t S \beta / \sigma^2), \tag{14}$$

an exponential prior. However, this prior is recognizable as being equivalent to an improper Multivariate Normal prior. Explicitly, this improper prior is simply

$$\beta \sim \mathcal{N}(0, \sigma^2 S^- / \lambda), \tag{15}$$

where S^{-} is a pseudoinverse of S (because S is rank deficient by the dimension of the penalty null space, it does not necessarily have an inverse). Here, if we use an eigendecomposition $S = U\Lambda U^t$, set Λ^{-} to be the diagonal matrix of the inverse of the nonzero eigenvalues, with zeros in place of the inverse for any eigenvalues which are zero; then, we have $S^{-} = U\Lambda^{-}U^t$. Next, our Bayesian interpretation of the smoothing penalty gives the model the structure of a linear mixed model, so the MAP estimate of β is the solution (11) to (10), while the posterior distribution

$$\beta \mid y \sim \mathcal{N}(\hat{\beta}, (X^tX + \lambda S)^{-1}\sigma^2). \tag{16}$$

At this point, one might be satisfied and proclaim that we have a fully Bayesian approach to AMs. However, recall that basis functions are not yet found: a fully Bayesian approach would also have to specify prior distributions to then obtain basis functions! The frequentist approach to this problem is covered in Section 2. However, the Bayesian approach to this problem is a lot less obvious; we survey various different attempts at such an approach which exist in the literature next.

1.3 Literature Review

In this section, we detail the various Bayesian approaches for specifying a prior distribution on the space of smooth functions used in AMs that we have explored in the literature. These approaches can be broadly classified into three categories: Gaussian Markov random field (GMRF) priors, Gaussian process (GP) priors, and local priors. We elaborate on the specific approaches within each category below, as well as analyze the most reasonable approaches to explore further in this paper.

Gaussian Markov Random Fields. One approach for setting a prior on AM functions comes directly from Wood [40]. If $\beta \sim \mathcal{N}(0, S^{-}/\lambda)$ (cf. Equation (15)), then

$$\underline{\mathrm{f}} \sim \mathcal{N}(0, XS^{-}X^t), \tag{17}$$

where $\underline{\mathrm{f}}$ denotes the vector containing the values of the smooth evaluated at the observed covariate values. This emphasizes the fact that we can view the smooth as a Gaussian random field. When the precision matrix S is sparse, we can view the smooth additionally as a Markov field, so in total, it will be a Gaussian Markov random field (GMRF). Such priors have a rich and deep theory (outlined within Section 5.8 of Wood's textbook [40]). The equivalence between smooths, random effects, and Gaussian random fields is further discussed in Kimeldorf and Wahba [23] and Silverman [36]. Additionally, exploring Gaussian Markov random fields, we were led to the fitting of AMs with integrated nested Laplace approximations (INLA) in [34]. This paper discusses hyperparameter estimation and also various non-Gaussian likelihoods. But importantly, this paper discusses how to efficiently approximate the posterior distribution under the GMRF prior. Finally, on the topic of INLA, we also looked at the recent paper by Gressani and Lambert [16], where even more techniques are introduced for fast Bayesian inference using INLA.

Gaussian Processes. Another approach for setting a prior on AM functions would involve starting with the simple case where there is only one covariate, looking at existing methods for the estimation of one smooth function. Here, the structure of our AM is exactly that of a Gaussian process. Hence, with this, the challenge is then to investigate how to generalize the Gaussian process approach when there is more than one covariate. One approach to do this is suggested by Solonen and Staboulis [37]: set the basis functions for the AM in question as the eigenfunctions of a Gaussian Process. In their paper, Solonen and Staboulis explain concretely how to approximate such eigenfunctions in practice, and also how to solve various computational problems that might arise with high dimensional data. Speaking in broad terms, this approach involves setting a Gaussian process prior for each f_j function in (2), and then adding the kernel functions of the f_j's up. An extension to this approach is what is often called additive Gaussian processes, as introduced by Duvenaud, Nickisch, and Rasmussen [10]. The approach mentioned here tries to generalize both AMs and Gaussian processes which use Gaussian kernels. In specific, this is achieved through a kernel which allows additive interactions of all orders, ranging from first order interactions (as in [37]) all the way to pth order interactions (as in a Gaussian process with a Gaussian kernel). Constructing the kernels which satisfy such a property is done by the authors by multiplying and adding kernels of smaller "orders" in a specific way; that is, for each dimension $i \in \{1, \dots, p\}$, we can assign a one-dimensional base kernel $k_i(x_i, x_i')$. Then, the authors define the first order, second order, and mth order additive kernel as:

$$k_{\text{add}_1}(x, x') = \sigma_1^2 \sum_{i=1}^{p} k_i(x_i, x_i'), \tag{18}$$

$$k_{\text{add}_2}(x, x') = \sigma_2^2 \sum_{i=1}^{p} \sum_{j=i+1}^{p} k_i(x_i, x_i') k_j(x_j, x_j'), \tag{19}$$

$$k_{\text{add}_m}(x, x') = \sigma_m^2 \sum_{1 \le i_1 < \cdots < i_n \le p} \prod_{d=1}^{n} k_{i_d}(x_{i_d}, x_{i_d}'), \tag{20}$$

where p is the dimension of the input space and σ_m^2 is the variance assigned to all mth order interactions (see [10] for details). However, such additive GP models require high-dimensional interaction terms, and to remedy this, Boukouvalas, Lu, and Hensman [27] proposed the orthogonal additive kernel (OAK). This kernel imposes an orthogonality constraint on the additive functions, and this, in turn, enables an identifiable as well as low-dimensional representation of the functional relationship. In [27], it is further demonstrated that with only a small number of additive low-dimensional terms, the OAK model achieves similar, or even better, predictive performance compared to black-box models while being interpretable.

Various Local Approaches. The frequentist approach to the problem of specifying a prior on AM functions would be to take a local approach, approximating each point on the function in a neighborhood, and then combining these local estimates to form a full estimate of the desired function. In contrast to the global approaches mentioned in the preceding paragraphs, the the local approach is much better at describing sharp local features and additionally can have the property known as spatially varying smoothness (cf. Section 4). So, it might be of substantial value to investigate Bayesian approaches which are of a local nature. In this regard, the paper by Solonen and Staboulis [37] recommends starting with a typical B-spline basis, and then placing Gaussian priors on their derivatives instead of restricting directly the values of the basis functions. These so-called "difference priors" may lead to improper priors, but this is not a problem as long as one has enough data to get a proper posterior. Meanwhile, Lang and Brezger [25] take a more refined approach. Namely, they use P-splines, low rank smoothers which use a B-spline basis [40], and replace difference penalties by their stochastic analogues, which are Gaussian (intrinsic) random walk priors that serve as smoothness priors for unknown regression coefficients. This results in a fully Bayesian inference, plus the authors use recent MCMC techniques for drawing random samples from the posterior [25]. Last, Chipman, George, and McCulloch [5] introduce perhaps the most flexible method to address the problem with the invention of the Bayesian additive regression tree (BART). BART is a Bayesian "sum-of-trees" model, where each tree is constrained by a regularization prior to be a weak learner, and fitting and inference are accomplished via an iterative Bayesiann backfitting MCMC algorithm that generates samples from a posterior. In some sense, BART could be considered to be an effective generalization of the AM, as it uses dimensionally adaptive random basis elements (the basis elements within an AM would of course be fixed). In other words, decision trees, instead of basis elements, are utilized.

Analysis and Subsequent Focus. As a matter of practicality, we make the decision to narrow our focus on one global approach and one local approach in Section 3 and Section 4, respectively, for our coverage of the Bayesian approach for AMs. Among the global approaches, a Gaussian process approach would align closer to the material we studied in Statistics 220, and thus would have a body of theory we can already rely on. Moreover, the theory of Gaussian Markov random fields, while quite interesting and rich, would take a rather lengthy exposition to fully introduce and motivate from the ground up. Then, among the Gaussian process approaches, we will choose to follow the approach of Solonen and Staboulis [37] as it adheres the closest to the AM formulation as per our introduction (and does not attempt to generalize the AM, which would lead to a fundamentally different model). Among the local approaches, we shall narrow our focus to the simplest one involving B-splines and difference priors. After all, the theory of splines is not one that is covered in Statistics 220, so we would have to develop it from the basics. For this regard, having a subsequent simplicity in our spline formulation of AMs would allow for a more detailed analysis of the model instead of having to develop more and more theory. Although we have unfortunately had to narrow our focus to only two Bayesian approaches, all the methods mentioned within Subsection 1.3 are extremely interesting approaches in their own right, and thus also deserve future exploration in the development of Bayesian additive modeling.

1.4 A Note About Generalized AMs

Generalized additive models (GAMs) follow from additive models, as generalized linear models follow from linear models. That is, the linear predictor now predicts some known smooth monotonic function of the expected value of the response, and the response may follow any exponential family distribution, or simply have a known mean-variance relationship, permitting the use of some quasilikelihood approach [40]. In other words, GAMs assume that given x_i, the distribution of y_i belongs to an exponential dispersion family, i.e. y_i has the conditional density

$$p(y_i \mid x_i) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{\phi}\right) c(y_i, \theta_i), \quad i = 1, \dots, n, \tag{21}$$

where $b(\cdot)$, $c(\cdot)$, θ_i, and ϕ determine the respective distributions [25]. A list of the most common exponential family distributions and their parameters can be seen in Fahrmeir and Tutz [12]. The mean $\mu_i = \mathbb{E}(y_i \mid x_i)$ is linked to a semiparametric additive predictor η_i by $\mu_i = h(\eta_i)$, where h is a known response function, and the inverse $g = h^{-1}$ of h is called the link function. Then, a GAM will be given by

$$\eta_i = \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip}), \quad i = 1, \ldots, n, \tag{22}$$

where again, $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $f_1, \ldots, f_p$ are unknown smooth functions of the covariates [25]. Now, it is tempting to suppose that all that is needed, to fit a GAM, is to fit (22) like how one would fit an AM, and then apply the inverse g to obtain inferences on the y_i's. Unfortunately, this is not the case [40]. Indeed, the AM is estimated by penalized least squares, but to fit the GAM, theory dictates a need for penalized likelihood maximization. In practice, this will be achieved by penalized iteratively reweighted least squares (PIRLS) [40]. Thus, while we only focus on AMs, many of the ideas discussed in this paper generalize to the fitting of GAMs as well, though the key nuance we just mentioned must be considered.

1.5 Overview of the Project

The rest of the paper is organized as follows. In Section 2, we detail the basics of frequentist AM theory. To do this, we'll introduce piecewise linear functions and natural cubic splines as perhaps the simplest archetypes for basis functions and the method of cross validation as a frequentist approach to smoothing. Then, we look at the two main algorithms used for the fitting of AMs, namely the backfitting algorithm and the local scoring algorithm. In Section 3, we begin our coverage of Bayesian approaches, starting with Gaussian process priors for AM functions. Here, we follow [37] to suggest methods for constructing a basis with GPs, and also study various techniques for high-dimensional inference (when computation with this GP method would usually be slow and expensive). In Section 4, we will look at the local approach to specifying a prior for AM functions utilizing splines and difference priors. Here, we also look at the high-dimensional case, as well as discuss various nuances of this particular approach such as fixing identifiability issues and making priors periodic and symmetric. We also introduce the concept of spatially varying smoothness. In Section 5, we provide the main bulk of our new contributions by applying all the methods discussed so far to data. Specifically, we will implement a spline model from Section 2, as well as special cases for the models introduced in Section 3 and Section 4, along with benchmark models, run on synthetic data, the mcycle dataset [36], and a dataset measuring air quality in Beijing [8, 44]. All the code for this paper can be found in the repository below:

https://github.com/zadchin/BayesianAM

2 Frequentist AM Theory

To start, we can extend the simplified AM model in (3) to one which contains two explanatory variables, i.e. we'll set $p = 2$ now. Given datapoints $\{((x_i, v_i), y_i)\}_{i=1}^n$, where the x_i's and v_i's are one-dimensional, our model shall take on the form

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \epsilon_i, \quad i = 1, \dots, n, \tag{23}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently. There are two points to note about this model. First, the assumption of additive effects is a fairly strong one: $f_1(x) + f_2(v)$ turns out to be a quite restrictive special case of the general smooth function $f(x, v)$ of two variables. Second, the fact that the model now contains more than 1 function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant. After all, any constant could be simultaneously added to f_1 and subtracted from f_2, without changing the model prediction. So, identifiability constraints have to be imposed on the model before one proceeds to fit the model. However, provided that the identifiability issue is addressed, the additive model can be represented using penalized regression splines and estimated by penalized least squares, in the same way for the simple univariate model in (3). Indeed, let

$$f_1(x) = \sum_{j=1}^{k_1} b_j(x)\delta_j, \tag{24}$$

$$f_2(v) = \sum_{j=1}^{k_2} \mathcal{B}_j(v)\gamma_j, \tag{25}$$

where the δ_j, γ_j are unknown coefficients and the $b_j(x), \mathcal{B}_j(v)$ are basis functions for $f_1(x), f_2(v)$, respectively. Then, we will obtain $\mathrm{f}_1 = X_1\delta$ and $\mathrm{f}_2 = X_2\gamma$, where

$$\mathrm{f}_1 = (f_1(x_1), \dots, f_1(x_n))^t, \tag{26}$$

$$\mathrm{f}_2 = (f_2(v_1), \dots, f_2(v_n))^t, \tag{27}$$

$$\delta = (\delta_1, \dots, \delta_{k_1})^t, \tag{28}$$

$$\gamma = (\gamma_1, \dots, \gamma_{k_2})^t, \tag{29}$$

$$(X_1)_{ij} = b_j(x_i), \quad i = 1, \dots, n, \quad j = 1, \dots, k_1, \tag{30}$$

$$(X_2)_{ij} = \mathcal{B}_j(v_i), \quad i = 1, \dots, n, \quad j = 1, \dots, k_2. \tag{31}$$

Moreover, a penalty of the form in (10) can be associated to both f_1 and f_2. These will be derived explicitly for two specific basis setups in Subsections 2.1 and 2.2.

But for now, suppose that f_1 can be associated the penalty $\delta^t \bar{S}_1 \delta$, and that f_2 can be associated the penalty $\gamma^t \bar{S}_2 \gamma$. Then, we'll deal with the identifiability problem. For estimation purposes, almost any linear constraint that removed the problem could be used, however most choices lead to uselessly wide confidence intervals for the constrained functions [40]. The best constraints from this viewpoint turn out to be sum-to-zero constraints, such as $\mathbb{1}^t \mathrm{f}_1 = 0$, or written in full explicit form,

$$\sum_{i=1}^{n} f_1(x_i) = 0. \tag{32}$$

Here, $\mathbb{1}$ is simply an n-vector of 1's. This constraint still allows f_1 to have exactly the same shape as before the constraint, with exactly the same penalty value. The constraint's only effect is to shift f_1 vertically, so that its mean value is zero.

Now, note that using this constraint is equivalent to requiring $\mathbb{1}^t X_1 \delta = 0$ for all δ, which in turn is equivalent to $\mathbb{1}^t X_1 = 0$. To achieve this, the column mean can be subtracted from each column of X_1: we define a column centered matrix

$$\tilde{X}_1 = X_1 - \mathbb{1}\mathbb{1}^t X_1 / n, \tag{33}$$

and then make $\tilde{\mathrm{f}}_1 = \tilde{X}_1 \delta$. This constraint imposes no more than a shift in the level of f_1. After all, if we let c denote the scalar $\mathbb{1}^t X_1 \delta / n$, we obtain the computation

$$\tilde{\mathrm{f}}_1 = \tilde{X}_1 \delta = X_1 \delta - \mathbb{1}\mathbb{1}^t X_1 \delta / n = X_1 \delta - \mathbb{1} c = \mathrm{f}_1 - c. \tag{34}$$

However, this constraint also introduces another problem: the column centering reduces the rank of $\tilde{X}_1$ to $k_1 - 1$, so only $k_1 - 1$ elements of the k_1-vector δ could be uniquely estimated. One simple additional identifiability constraint deals with this problem: a single element of δ is set to zero, and the corresponding column of $\tilde{X}_1$ and the D_1 matrix that is associated to $\bar{S}_1$ (see Subsection 2.1, say) is deleted. This recipe is applicable to any basis which includes the constant function in its span, and has a penalty that is zero for constant functions; for bases that explicitly include a constant function, it is not sufficient to set any coefficient to zero, and the coefficient for the constant is the one to constrain. Now, the column centered rank reduced basis will automatically satisfy the identifiability constraint. Hence, from now on, we can drop all the tildes, and it is assumed that relevant matrices and vectors are their constrained versions. Having done this, we can proceed to reexpress (23), with $X = (\mathbb{1}, X_1, X_2)$ and $\beta^t = (\alpha, \delta^t, \gamma^t)$, in the familiar linear form

$$y = X\beta + \epsilon, \tag{35}$$

where (6) and (8) define y and ϵ. For notational convenience, it is also very useful to express the penalties $\delta^t \bar{S}_1 \delta$ and $\gamma^t \bar{S}_2 \gamma$ as quadratic forms in the full coefficient vector β. This is easily accomplished by padding out $\bar{S}_1$ and $\bar{S}_2$ with zeros, giving

$$\beta^t S_1 \beta = (\alpha, \delta^t, \gamma^t) \begin{pmatrix} 0 & 0 & 0 \\ 0 & \bar{S}_1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \delta \\ \gamma \end{pmatrix} = \delta^t \bar{S}_1 \delta, \tag{36}$$

$$\beta^t S_2 \beta = (\alpha, \delta^t, \gamma^t) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \bar{S}_2 \end{pmatrix} \begin{pmatrix} \alpha \\ \delta \\ \gamma \end{pmatrix} = \gamma^t \bar{S}_2 \gamma. \tag{37}$$

Now, the coefficient estimates of the model (23) are obtained by the minimization of the penalized least squares objective given, analogously to (10), by the form

$$||y - X\beta||^2 + \lambda_1 \beta^t S_1 \beta + \lambda_2 \beta^t S_2 \beta, \tag{38}$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of ensuring that f_1 and f_2 are smooth, relative to the objective of closely fitting the response data. If λ_1 and λ_2 are given, we have the following result.

Theorem 2.1. *The formal expression for the minimizer of* (38), $\hat{\beta}$, *is given below:*

$$\hat{\beta} = (X^t X + \lambda_1 S_1 + \lambda_2 S_2)^{-1} X^t y. \tag{39}$$

Proof. This expression can be realized by setting partial derivatives to zero. Set

$$f_{\text{obj}}(\beta) = ||y - X\beta||^2 + \lambda_1 \beta^t S_1 \beta + \lambda_2 \beta^t S_2 \beta. \tag{40}$$

Then, since $\beta^t S_1 \beta$ and $\beta^t S_2 \beta$ are quadratic forms, the relevant partial derivative is

$$\frac{\partial f_{\text{obj}}(\beta)}{\partial \beta} = -2X^t(y - X\beta) + 2\lambda_1 S_1 \beta + 2\lambda_2 S_2 \beta. \tag{41}$$

Setting this to zero, cancelling the constant term 2, and rearranging the resulting equation in terms of β, we get, from this simplification, the desired expression. □

It is clear that the analysis in this section, done for the special case $p = 2$, can be easily generalized for any arbitrary p: for the sake of conciseness, we preferred the $p = 2$ case in our derivations so far. For a complete fitting of such an AM, we still have two hurdles to resolve. Namely, we have to deal with actual bases and show how the penalty matrices $\bar{S}_1$ and $\bar{S}_2$ are constructed from them, and we will also have to determine how to choose the smoothing parameters λ_1 and λ_2.

The next two subsections address this first issue with two natural basis choices, while the following subsection introduces cross validation as a way to solve the second issue. In the final two subsections, we pivot completely from this current approach using penalized regression splines and penalized least squares, instead exploring the backfitting algorithm and local scoring algorithm as ways to have numerical approximations for the functions f_j themselves, thus succeeding again in the fitting of additive models and generalized additive models, respectively.

2.1 Piecewise Linear Functions

First, we specialize to the problem of estimating a single smooth function f of a single variable. For instance, the simplest thing to try is to approximate our f via polynomial functions. The crux of this is Lagrange's interpolation theorem [4].

Theorem 2.2. *Given n datapoints $\{(x_i, y_i)\}_{i=1}^n$, where the x_i's are distinct, there is a polynomial P with real coefficients which satisfies $P(x_i) = y_i$ for each $i = 1, \dots, n$. Moreover, if the condition $\deg(P) < n$ is imposed, such a polynomial P is unique.*

Proof. To prove existence, consider $g(x) = (x - x_1) \cdots (x - x_n)$, and then define

$$P_i(x) = \frac{g(x)}{(x - x_i)g'(x_i)}, \quad i = 1, \dots, n. \tag{42}$$

By construction, for all $i = 1, \dots, n$, $P_i(x_i) = 1$ and $P_i(x_j) = 0$ for all $j \neq i$. Whence,

$$P(x) = \sum_{i=1}^{n} y_i P_i(x) \tag{43}$$

is a polynomial with real coefficients satisfying $P(x_i) = y_i$ for all $i = 1, \dots, n$. It is a sum of polynomials of degree $n - 1$, so its degree is less than n. Now, to prove uniqueness under the condition $\deg(P) < n$, assume that Q and R are polynomials with the desired properties. Now, $Q - R$ is a polynomial of degree less than n that vanishes on $x_1, \dots, x_n$ hence the zero polynomial. Whence, $Q = R$, as desired. □

However, using polynomials as basis functions for an AM is inherently problematic. Taylor's theorem implies that polynomial bases will be useful for situations in which interest focuses on properties for f in the vicinity of a single specified point, but when the questions of interest relate to f over its whole domain, this basis will result in a prediction that oscillates wildly in places, to accommodate the twin requirements to interpolate the data and to have every derivative with respect to x continuous. If we relax the requirement for continuity of derivatives, and simply use a piecewise linear interpolant, a much better approximation is obtained. All this is illustrated in the three panels of Figure 2 displayed below.

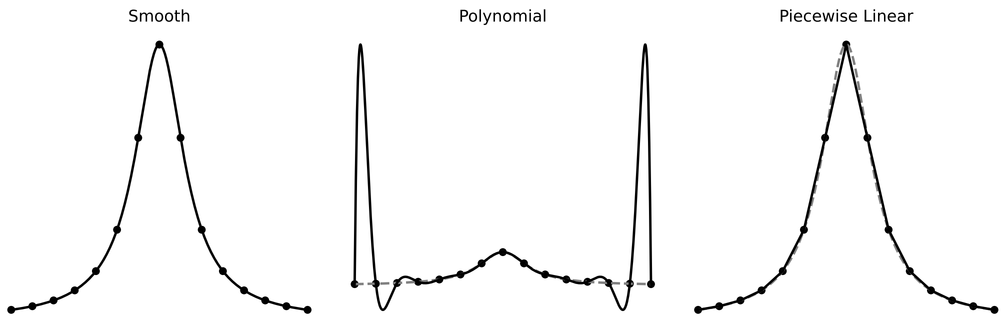


Figure 2: The left panel displays the smooth function $1/(1+x^2)$ and also some points (in black dots) on this function. The middle panel shows an attempt to reconstruct (in black) the function (dashed curve) by polynomial interpolation of the black dots. The rightmost panel shows the equivalent piecewise linear interpolant. The condition that polynomials should interpolate the data with all its derivatives continuous yields wilder oscillations.

Thus, it makes sense to use bases that are good at approximating known functions in order to represent unknown functions. Similarly, bases that perform well for interpolating exact observations of a function are also a good starting point with regard to the task of smoothing noisy observations of a function. We will see in Subsection 2.2 that piecewise linear bases can be improved upon by spline bases having continuity of just a few derivatives, but the piecewise linear case provides a rather good illustration, and therefore we will stick with it for this subsection.

A basis for piecewise linear functions of a univariate variable x is determined entirely by the locations of the function's derivative discontinuities; that is, the locations at which the linear pieces join up. These are called knots: let them be

$$\{x_j^* \mid j = 1, \dots, k\}. \tag{44}$$

Suppose additionally that $x_j^* > x_{j-1}^*$. Then, the piecewise linear basis is given by

$$b_1(x) = \begin{cases} (x_2^* - x)/(x_2^* - x_1^*) & x < x_2^*, \\ 0 & \text{otherwise,} \end{cases} \tag{45}$$

for $b_1(x)$, the piecewise definition that is shown below for each $j = 2, \ldots, k-1$:

$$b_j(x) = \begin{cases} (x - x^*_{j-1})/(x^*_j - x^*_{j-1}) & x^*_{j-1} < x \leq x^*_j, \\ (x^*_{j+1} - x)/(x^*_{j+1} - x^*_j) & x^*_j < x < x^*_{j+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{46}$$

and the piecewise definition (that is similar to that for $b_1(x)$) for the term $b_k(x)$:

$$b_k(x) = \begin{cases} (x - x^*_{k-1})/(x^*_k - x^*_{k-1}) & x > x^*_{k-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{47}$$

Therefore, $b_j(x)$ is zero everywhere, except over the interval between the knots immediately to either side of x^*_j: $b_j(x)$ increases linearly from 0 at x^*_{j-1} to 1 at x^*_j, and then decreases linearly to 0 at x^*_{j+1}. Because of their shape, the b_j are often known as "tent functions." Fitting data with such a basis is displayed in Figure 3.

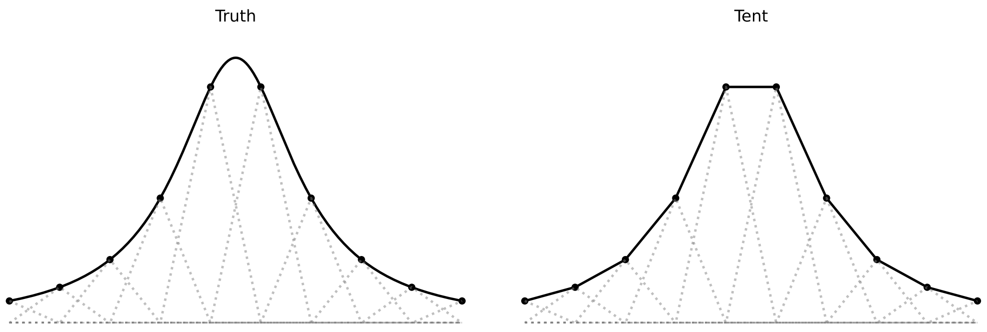


Figure 3: Fitting points on a smooth curve with a combination of tent functions.

Note: an equivalent way of defining $b_j(x)$ is as the linear interpolant of the data

$$\{(x^*_i, \delta_{ij}) \mid i = 1, \ldots, k\}, \tag{48}$$

where δ_{ij} is the Kronecker delta function that is 1 if $i = j$ and 0 otherwise; that is, the indicator function of the event $i = j$. This formulation makes for very easy implementation of the basis in a scientific programming language like Python.

Now, recall that depending on the number of knots chosen for a given function, our model can make smoother (many knots) or rougher (few knots) predictions. Moreover, recall that a common approach to controlling the smoothness of the resulting predictor is to heuristically select "more than enough" knots, and then use an additional penalty term that regularizes towards smoothness [40]. Hence, in the case of piecewise linear functions, we can add on a term that will measure the "wiggliness" of the function using squared second differences of the function at the knots (which shall crudely approximate the integrated squared derivative of the second order penalty used in cubic spline smoothing, see Subsection 2.2). Thus, the penalty term for a given component function f of an AM is of the form

$$\lambda \sum_{j=2}^{k-1} [f(x_{j-1}^*) - 2f(x_j^*) + f(x_{j+1}^*)]^2. \tag{49}$$

For the piecewise linear basis, we can write the sum above as a quadratic form.

Theorem 2.3. *There exist matrices D and S such that the following equality holds:*

$$\sum_{j=2}^{k-1} [f(x_{j-1}^*) - 2f(x_j^*) + f(x_{j+1}^*)]^2 = \beta^t S \beta, \;\; S = D^t D, \tag{50}$$

where β is the coefficient vector for f when expressed in the piecewise linear basis.

Proof. First, it is clear to see that for the basis of tent functions, the coefficients of f are simply the function values at the knots, i.e. $\beta_j = f(x_j^*)$. Moreover, we have

$$\begin{pmatrix} \beta_1 - 2\beta_2 + \beta_3 \\ \beta_2 - 2\beta_3 + \beta_4 \\ \beta_3 - 2\beta_4 + \beta_5 \\ \cdot \\ \cdot \end{pmatrix} = \begin{pmatrix} 1 & -2 & 1 & 0 & \cdot & \cdot & \cdot \\ 0 & 1 & -2 & 1 & 0 & \cdot & \cdot \\ 0 & 0 & 1 & -2 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \cdot \\ \cdot \end{pmatrix}, \tag{51}$$

and writing the right side above as $D\beta$, we get, by the definition of the $(k-2) \times k$-matrix D, that the penalty, expressed as a sum over β_{j-1}, β_j, and β_{j+1}, becomes

$$\sum_{j=2}^{k-1} (\beta_{j-1} - 2\beta_j + \beta_{j+1})^2 = \beta^t D^t D \beta = \beta^t S \beta, \tag{52}$$

where S is the rank deficient matrix defined by $S = D^t D$. Thus, we are done. □

With this, we finally have enough tools to fully specify a two-component additive model of the form (23). Specifically, we can endow f_1 with a piecewise linear basis using (48), and then construct D_1 and $\bar{S}_1$ following the procedure in Theorem 2.3, and we can do the same for f_2. Observe that we have assumed even knot spacing for our analysis so far. This is usually what is chosen in practice, as uneven knot spacing would usually require some reweighing of the penalty terms [40]. Hence, the first initial issue posed before Subsection 2.1 has been addressed. We look at a more refined method of tackling this issue next, delving into spline theory.

2.2 Natural Cubic Splines

The piecewise linear smoother of the previous subsection is a perfectly reasonable way of representing the smooth functions in additive models, but some substantial improvement is still possible. In particular, if we choose to represent our smooths in terms of spline bases, it is possible to get greatly reduced approximation error for our functions when the dimension of the smoothing basis is fixed. To see why this is the case, we shall spend some time on the theoretical properties that make splines so appealing for penalized regression. We shall express these theoretical ideas by considering specifically the properties of cubic splines, first in the context of interpolation, and then in the context of smoothing. Consider the set of points

$$\{(x_i, y_i) \mid i = 1, \ldots, n\}, \quad x_i < x_{i+1}. \tag{53}$$

Then, the natural cubic spline, $g(x)$, interpolating these points, is a function that is made up of sections of cubic polynomial, one for each $[x_i, x_{i+1}]$, which are then joined together such that the whole spline is continuous to the second derivative and $g(x_i) = y_i$, $g''(x_1) = g''(x_n) = 0$ are both satisfied. Such a function g exists as, for each piecewise cubic polynomial, it suffices to interpolate between endpoints and match the first and second derivatives between this piece with the adjacent pieces. This yields $4(n-1)+2$ equations to solve, and the boundary conditions give us 2 equations for a total of $4n$ equations in $4n$ unknowns (each cubic is four unknowns). This results in unique identification. Now, Green and Silverman [15] show that in the context of interpolation, natural cubic splines are, in fact, "best."

Theorem 2.4. *Of all functions which are continuous on $[x_1, x_n]$ having absolutely continuous first derivatives and interpolating $\{(x_i, y_i)\}_{i=1}^n$, $g(x)$ is the smoothest one in the sense that this natural cubic spline is the minimizer of the following integral:*

$$J(f) = \int_{x_1}^{x_n} f''(x)^2 \, dx. \tag{54}$$

Proof. Let $f(x)$ be some interpolant of $\{(x_i, y_i)\}_{i=1}^n$, other than $g(x)$, and let $h(x) = f(x) - g(x)$. We would like an expression for $J(f)$ in terms of $J(g)$. Thus, expand

$$\int_{x_1}^{x_n} f''(x)^2 \, dx = \int_{x_1}^{x_n} [g''(x) + h''(x)]^2 \, dx \tag{55}$$

$$= \int_{x_1}^{x_n} g''(x)^2 \, dx + 2 \int_{x_1}^{x_n} g''(x)h''(x) \, dx + \int_{x_1}^{x_n} h''(x)^2 \, dx. \tag{56}$$

Then, integrating the second term in the last line above by parts, we shall obtain

$$\int_{x_1}^{x_n} g''(x)h''(x) \, dx = g''(x_n)h'(x_n) - g''(x_1)h'(x_1) - \int_{x_1}^{x_n} g'''(x)h'(x) \, dx \tag{57}$$

$$= -\int_{x_1}^{x_n} g'''(x)h'(x) \, dx \tag{58}$$

$$= -\sum_{i=1}^{n-1} g'''(x_i^+) \int_{x_i}^{x_{i+1}} h'(x) \, dx \tag{59}$$

$$= -\sum_{i=1}^{n-1} g'''(x_i^+)[h(x_{i+1}) - h(x_i)] \tag{60}$$

$$= 0. \tag{61}$$

Here, (57) and (58) follow from the fact that $g''(x_1) = g''(x_n) = 0$, and (59) follows from the fact that $g(x)$ is made up of sections of cubic polynomial, so $g'''(x)$ turns out to be constant over any open interval (x_i, x_{i+1}); x_i^+ denotes an element of such an interval. Finally, (61) follows from the fact that both $f(x)$ and $g(x)$ are in fact interpolants, hence equal at x_i, implying that $h(x_i) = 0$. Therefore, we infer that

$$J(f) = \int_{x_1}^{x_n} g''(x)^2 \, dx + \int_{x_1}^{x_n} h''(x)^2 \, dx \geq J(g), \tag{62}$$

with equality if and only if $h''(x) = 0$ for $x_1 < x < x_n$. But, $h(x_1) = h(x_n) = 0$, so equality occurs in (62) if and only if $h''(x)$ vanishes on $[x_1, x_n]$. So put differently, any interpolant that is not identical to $g(x)$ shall have a strictly larger integrated squared second derivative, and this is precisely what we wanted to exhibit. □

Besides this result, it is shown in Chapter 5 of de Boor [7] that the cubic spline interpolant is optimal in various other respects. For instance, they exhibit that if a cubic spline, g, is used to approximate a function, $\tilde{f}$, by interpolating the set of points $\{(x_i, \tilde{f}(x_i) \mid i = 1, \dots, n\}$ and matching $\tilde{f}'(x_1)$ and $\tilde{f}'(x_n)$, then if $\tilde{f}(x)$ has four continuous derivatives, the following displayed "sharp" inequality holds:

$$\max |\tilde{f} - g| \leq \frac{5}{384} \max |x_{i+1} - x_i|^4 \max |\tilde{f}''''|. \tag{63}$$

By sharp, we mean that the constant 5/384 cannot be improved upon. Thus, with the properties of spline interpolants we have just studied, it is reasonable to then suggest that splines ought to give rise to a good basis for representing functions in an AM. We will see that Theorem 2.4 justifies creating cubic smoothing splines.

In statistics, y_i is usually measured with noise, hence it makes more sense to smooth the data $\{(x_i, y_i)\}_{i=1}^n$ instead of interpolating this data. Thus, rather than setting $g(x_i) = y_i$, we will instaed treat the $g(x_i)$'s as n free parameters of a cubic spline, and then estimate these parameters in order to minimize the expression

$$\sum_{i=1}^n [y_i - g(x_i)]^2 + \lambda \int g''(x)^2 \, dx, \tag{64}$$

where λ is a tunable parameter used to control the relative weight to be given to the conflicting goals of matching the data and producing a smooth g. The function $g(x)$ that results is called a smoothing spline. We now have the following result:

Theorem 2.5. *Of all functions f that are continuous on $[x_1, x_n]$ and have absolutely continuous first derivatives, $g(x)$ is the function that minimizes the expression below:*

$$\sum_{i=1}^n [y_i - f(x_i)]^2 + \lambda \int f''(x)^2 \, dx. \tag{65}$$

Proof. Suppose that some other function $f^*(x)$ minimized (65). Then, we are able to interpolate $\{(x_i, f^*(x_i))\}_{i=1}^n$ using a cubic spline, $g(x)$, so that $g(x)$ and $f^*(x)$ shall have the same sum of squares term in (65). However, by Theorem 2.4, $g(x)$ will have a lower integrated squared second derivative unless $f^* = g$. Therefore, $g(x)$ will give rise to a lower value of (65) than $f^*(x)$ unless $f^* = g$, as needed. □

Observe that we could have repeated the exact same proof if we change the sum of squares residual in (65) to any other measure of fit dependent on f only from $f(x_1), \dots, f(x_n)$. In particular, we might instead choose a loglikelihood measure. Thus, rather than being chosen in advance, the cubic spline basis arises naturally from the specification of the smoothing objective (65), and in this sense, we infer that model fit and smoothness are precisely defined in a basis independent way.

In what remains, we will provide an explicit construction of a cubic penalized regression spline. This involves constructing some spline basis (and the associated penalties) for a much smaller dataset than the one to be analyzed, and then using that basis (plus penalties) to model the original dataset. The covariate values from the smaller dataset should be arranged to appropriately cover the distribution of covariate values in the original dataset. Here, there is a question of how many basis functions to use, but no answer to this is generally possible without knowing the true functions that we are trying to estimate [40]. Hence, assume, once again, that we are trying to estimate a smooth function f of a single variable, but now with a cubic spline basis that has k knots $x_1, \dots, x_k$. Then, define the expressions

$$h_j = x_{j+1} - x_j, \tag{66}$$
$$a_j^-(x) = (x_{j+1} - x)/h_j, \tag{67}$$
$$a_j^+(x) = (x - x_j)/h_j, \tag{68}$$
$$c_j^-(x) = [(x_{j+1} - x)^3/h_j - h_j(x_{j+1} - x)]/6, \tag{69}$$
$$c_j^+(x) = [(x - x_j)^3/h_j - h_j(x - x_j)]/6, \tag{70}$$
$$D_{i,i} = 1/h_i, \tag{71}$$
$$D_{i,i+1} = -1/h_i - 1/h_{i+1}, \tag{72}$$
$$D_{i,i+2} = 1/h_{i+1}, \tag{73}$$
$$B_{i,i} = (h_i + h_{i+1})/3, \tag{74}$$
$$B_{i,i+1} = h_{i+1}/6, \tag{75}$$
$$B_{i+1,i} = h_{i+1}/6, \tag{76}$$

where $j = 1, \dots, k$ and $i = 1, \dots, k-2$ for D, $i = 1, \dots, k-3$ for B. These formulas define the components of a basis for f and the nonzero entries in two matrices, D and B. Additionally, let $\beta_j = f(x_j)$ and $\delta_j = f''(x_j)$. Then, we could express

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\delta_j + c_j^+(x)\delta_{j+1}, \quad x_j \le x \le x_{j+1}, \tag{77}$$

where the basis functions $a_j^-, a_j^+, c_j^-, c_j^+$ are defined in (67), (68), (69), (70). We have that the cubic regression spline defined by (77) has value β_j and second derivative δ_j at knot x_j, and that value and second derivative are continuous across the knots. Moreover, having the spline continuous at the second derivative implies that:

Theorem 2.6. *The conditions that the spline in* (77) *must be continuous to second derivative, at the x_j, and should have zero second derivative at x_1 and x_k, implies*

$$B\delta^- = D\beta, \tag{78}$$

where $\delta^- = (\delta_2, \ldots, \delta_{k-1})^t$, $\delta_1 = \delta_k = 0$, *and* B *and* D *are defined within* (71) *to* (76).

Proof. At knot position x_{j+1}, we require that the derivative at x_{j+1} of the section of cubic to the left of x_{j+1} matches the derivative at x_{j+1} of the section of cubic to the right of x_{j+1}. Writing this condition out in full yields the following equation:

$$\begin{aligned} -\frac{\beta_j}{h_j} + \frac{\beta_{j+1}}{h_j} + \delta_j \frac{h_j}{6} + \delta_{j+1}\frac{3h_j}{6} - \delta_{j+1}\frac{h_j}{6} &= \\ -\frac{\beta_{j+1}}{h_{j+1}} + \frac{\beta_{j+2}}{h_{j+1}} - \delta_{j+1}\frac{3h_{j+1}}{6} + \delta_{j+1}\frac{h_{j+1}}{6} - \delta_{j+2}\frac{h_{j+1}}{6}&. \end{aligned} \tag{79}$$

Now, simple rearrangement of the equation above yields the following equation:

$$\frac{1}{h_j}\beta_j - \left(\frac{1}{h_j} + \frac{1}{h_{j+1}}\right)\beta_{j+1} + \frac{1}{h_{j+1}}\beta_{j+2} = \frac{h_j}{6}\delta_j + \left(\frac{h_j}{3} + \frac{h_{j+1}}{3}\right)\delta_{j+1} + \frac{h_{j+1}}{6}\delta_{j+2}. \tag{80}$$

With the restriction $\delta_1 = \delta_k = 0$, repeating (80) for $j = 1, \ldots, k-2$ yields (78). □

Moving ahead, defining $F^- = B^{-1}D$, we have that $\delta = F\beta$, where F is defined as

$$F = \begin{pmatrix} 0 \\ F^- \\ 0 \end{pmatrix}. \tag{81}$$

Therefore, the spline can be rewritten entirely in terms of β as we display below:

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)F_j\beta + c_j^+(x)F_{j+1}\beta, \;\; x_j \le x \le x_{j+1}, \tag{82}$$

which can be rewritten, by isolating each β_i, in the familiar form of (4) as follows:

$$f(x) = \sum_{i=1}^{k} b_i(x)\beta_i, \tag{83}$$

where the new basis functions $b_i(x)$ are defined implicitly. Furthermore, we have

$$\int_{x_1}^{x_k} f''(x)^2 \, dx = \beta^t D^t B^{-1} D\beta, \tag{84}$$

as shown by Lancaster and Salkauskas [24], so $S = D^t B^{-1} D$ is the penalty matrix for this basis. Thus, just like in the end of Subsection 2.1, we once again are able to specify a two-component additive model of the form (23). This time, we could endow f_1 with a cubic spline basis of the form in (83), and then construct $\bar{S}_1$ as in (84). This time, however, the approach in (34) does not quite work because $\bar{S}_1$ can no longer be expressed as $D_1^t D_1$ for some diagonal matrix D_1 in general. But in this case, we can instead use the general QR approach described in Section 5.4.1 of Wood [40]. We can then repeat this whole process for f_2. Thus, we have found a basis for smooth functions in an AM that are far more useful than the piecewise linear basis. An illustration of this cubic spline basis is shown below in Figure 4.

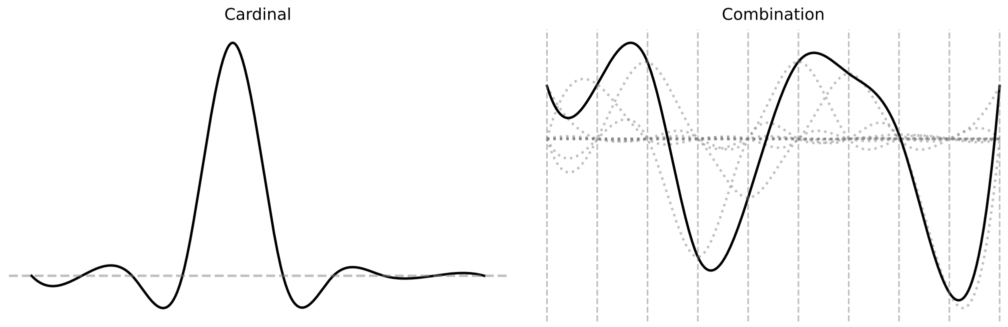


Figure 4: The left panel illustrates one basis function for a cubic regression spline of the type discussed in (77): this basis function takes the value one at one knot of the spline, and zero at all other knots (such basis functions are sometimes denoted as "cardinal basis functions"). The right panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions of a cubic regression spline, each multiplied by their associated coefficient: these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve, and additionally, the vertical thin lines display the knot locations which were specified.

2.3 Smoothing via Cross Validation

We return to the two-component model in (23), with the purpose of answering the second question issued before Subsection 2.1. The hat matrix of model (35) is

$$A = X(X^tX + \lambda_1 S_1 + \lambda_2 S_2)^{-1}X^t, \tag{85}$$

and now we concern ourselves with the question of estimating $\lambda = (\lambda_1, \lambda_2)$. Now, an appealing approach, when the variance of the data is known and constant, is to try to ensure that the expected mean square error is minimized; that is, that

$$M = \mathbb{E}(\|\mathbb{E}(y) - X\hat{\beta}\|^2/n) = \mathbb{E}(\|y - Ay\|^2)/n - \sigma^2 + 2\,\mathrm{tr}(A)\sigma^2/n \tag{86}$$

is minimized. This leads to the unbiased risk estimator (UBRE) introduced in 1979 by Craven and Wahba [6]. The UBRE is given by the following expression below:

$$\mathcal{V}_u = \|y - Ay\|^2/n - \sigma^2 + 2\,\mathrm{tr}(A)\sigma^2/n. \tag{87}$$

Observe that $\mathcal{V}_u$ depends on the smoothing parameters through A. Now, if σ^2 is known, then estimating λ by minimizing $\mathcal{V}_u$ works well, but problems arise for the case where σ^2 has to be estimated (see Section 6.2.1 in Wood [40]). Hence, an alternative is to base smoothing parameter estimation on mean square prediction error; that is, on the average squared error in predicting some new observation $\tilde{y}$ using the fitted model. This expected mean square prediction error is simply [40]

$$P = \sigma^2 + M. \tag{88}$$

The direct dependence on σ^2 suggests that basing criteria on P ensures that said criteria are more resistant to over-smoothing, which would inflate the σ^2 estimate, than are criteria based on M alone. The most natural way to estimate P is to use cross validation: by omitting a datum, y_i, from the model fitting process, it shall become independent of the model fitted to the remaining data. Hence, the error squared in predicting y_i is readily estimated, and by omitting all data in turn, we arrive at the ordinary cross validation estimate of P that is defined in Wood [40]:

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_i^{[-i]})^2, \tag{89}$$

where $\mu_i^{[-i]}$ denotes the prediction of $\mathbb{E}(y_i)$ obtained from the model fitted from all data except y_i. Fortunately, it is not necessary to calculate $\mathcal{V}_o$ by performing n model fits to obtain the n terms in the sum above. This is due to the following:

Theorem 2.7. *The ordinary cross validation (OCV) estimate, $\mathcal{V}_o$, can be written as*

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}, \tag{90}$$

where $\hat{\mu}_i$ is from the fit to the full y. Whence, $\mathcal{V}_o$ can be calculated from a single fit.

Proof. Consider the penalized least squares objective which, in principle, has to be minimized to find the ith term in the OCV score. Clearly, adding zero to this will leave the estimates that minimize it completely unchanged. Thus, by adding

$$(\hat{\mu}_i^{[-i]} - \hat{\mu}_i^{[-i]})^2 \tag{91}$$

to this objective and setting $\bar{y}^{[i]}$ and $\bar{\mu}^{[i]}$ to be zero vectors except for y_i and $\hat{\mu}_i^{[-i]}$ as their ith elements, we find that this objective can now be rewritten in the form

$$\sum_{j=1}^{n}(y_j^* - \hat{\mu}_j^{[-i]})^2 + \text{Penalties}, \quad y^* = y - \bar{y}^{[i]} + \bar{\mu}^{[i]}. \tag{92}$$

Here, the penalties do not depend on which observations are included in the sum of squares term. Now, considering the ith prediction, we can then deduce that

$$\hat{\mu}_i^{[-i]} = A_i y^* = A_i y - A_{ii} y_i + A_{ii}\hat{\mu}_i^{[-i]} = \hat{\mu}_i - A_{ii} y_i + A_{ii}\hat{\mu}_i^{[-i]} \tag{93}$$

since minimizing (92) obviously results in ith prediction $\hat{\mu}_i^{[-i]}$ and influence matrix A, which is just the influence matrix for the model fitted to all the data (since (92) has the structure of the fitting objective for the model of the whole data). Finally, subtracting y_i from both sides of (93) and performing some rearrangement yields

$$y_i - \hat{\mu}_i^{[-i]} = \frac{y_i - \hat{\mu}_i}{1 - A_{ii}}, \tag{94}$$

and with this, it is hence clear that the OCV score, $\mathcal{V}_o$, is of the desired form. □

Following the exposition in Wood [40], we note that OCV is a reasonable way of estimating smoothing parameters, but it suffers from a potential drawback: it has a slightly disturbing lack of invariance. To see this, consider our additive model fitting objective (38). Given smoothing parameters, all inferences about β made on the basis of minimizing this objective are identical to the inferences that are made by using the alternate objective, with Q some orthogonal matrix, given by

$$||Qy - QX\beta||^2 + \lambda_1\beta^t S_1\beta + \lambda_2\beta^t S_2\beta. \tag{95}$$

However, the two objectives generally give rise to different OCV scores! Such a problem arises because, despite parameter estimates, effective degrees of freedom, and expected prediction error being invariant to rotation of $y - X\beta$ by orthogonal matrices Q, the elements A_{ii} are not invariant. To fix this, it is possible to choose Q to make all the A_{ii} equal [40]. Doing so, we get the generalized cross validation

$$\mathcal{V}_g = \frac{n||y - \hat{\mu}||^2}{[n - \text{tr}(A)]^2} \tag{96}$$

score, i.e. the GCV score. Here, $\hat{\mu}$ is the estimate of $\mathbb{E}(y)$ from the fit. Note that since the expected prediction error is unaffected by the rotation, and GCV is just

OCV on the rotated problem, thus GCV must be as valid an estimate of prediction error as OCV, but of course, GCV has the nice property of being invariant. So far, we have introduced three cross validation scores, $\mathcal{V}_u$, $\mathcal{V}_o$, and $\mathcal{V}_g$, that we could use to estimate λ. In the case where σ^2 is unknown (so that using $\mathcal{V}_u$ is not ideal), the main practical problem with OCV or GCV is that these scores are not nearly sensitive enough to overfit. Thus, using them, we will tend to do worse than we could with actual prediction. To remedy this, Wood [40] suggests choosing the smoothing parameters in order to get the closest match between predicted values for the same point when it is excluded and when it is included in the fit. Namely,

$$c_1 = \frac{1}{n}\sum_{i=1}^{n}(\hat{\mu}_i - \hat{\mu}_i^{[-i]})^2 \tag{97}$$

would be the quantity we would want to study. Similarly to Theorem 2.7, we get

$$c_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{\mu}_i)^2 A_{ii}^2}{(1 - A_{ii})^2}, \tag{98}$$

and if we approximate the A_{ii} in the equation by their mean, as in GCV, we have

$$c_2 = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{\mu}_i)^2 \operatorname{tr}(A)^2}{[n - \operatorname{tr}(A)]^2}. \tag{99}$$

On its own, the above is not very useful, as there is nothing anchoring c_1 to the data, so we can combine it with the GCV score. A natural method to do this is

$$c_3 = \mathcal{V}_g + \sqrt{c_2 \mathcal{V}_g} = \frac{n\sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2}{[n - \operatorname{tr}(A)]^2}\left(1 + \frac{\operatorname{tr}(A)}{n} + \frac{\operatorname{tr}(A)^2}{n^2}\right). \tag{100}$$

From this, some manipulations and approximations (see Wood [40] or Kim and Gu [22]) will then lead us to the double cross validation (DCV) score defined as

$$\mathcal{V}_d = \frac{n\|y - \hat{\mu}\|^2}{[n - 1.5\operatorname{tr}(A)]^2}. \tag{101}$$

2.4 The Backfitting Algorithm

Now, pivoting completely from the penalized regression approach that has been our main focus thus far, we will introduce the backfitting algorithm, which turns out to be another popular technique for estimating the functions in an AM directly without the use of basis functions or smoothing penalties. This algorithm is some special case of the Gauss-Seidel procedure for solving linear systems [35]. First, we introduce scatterplot smoothers, and then delve into the backfitting algorithm for AMs, which we will then extend to the local scoring algorithm for GAMs in the following subsection. Our model setup here will be the same as in (2); that is,

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), \quad i = 1, \dots, n, \tag{102}$$

where $x_i = (x_{i1}, \dots, x_{ip})^t$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently for the data $\{(x_i, y_i)\}_{i=1}^n$. Then, the backfitting algorithm fits our additive model with the following steps.

Algorithm 1: Backfitting Algorithm for Additive Models

```
1.1  initialize $\beta_0 \leftarrow \bar{y}$
1.2  initialize $f_j \leftarrow 0$ for $j = 1, \dots, p$
1.3  repeat
1.4      for $j = 1, \dots, p$ do
1.5          $e_j \leftarrow y - \beta_0 - \sum_{k \neq j} f_k$
1.6          $f_j \leftarrow \mathcal{S}_j[e_j]$
1.7          $f_j \leftarrow f_j - \sum_{i=1}^n f_j(x_{ij})/n$
1.8      $\delta \leftarrow \sum_{i=1}^n [y_i - \beta_0 - \sum_{j=1}^p f_j(x_{ij})]^2$
1.9  until $\delta \leq \epsilon$
1.10 return $f_j$ for $j = 1, \dots, p$
```

Many explanations are in order, following the exposition in Section 9.1.1 of Hastie, Tibshirani, and Friedman [20]. First, as we have mentioned multiple times before, the constant β_0 is not identifiable, since we can add or subtract any constants to each of the functions f_j and adjust β_0 accordingly. The standard convention is

$$\sum_{i=1}^{n} f_j(x_{ij}) = 0, \quad j = 1, \dots, p, \tag{103}$$

so that the functions average zero over the data. In this case, it could be checked that $\beta_0 = \bar{y}$. Now, Algorithm 1 takes in the data, a grid of points, and a threshold value ϵ. It sets $\beta_0 = \bar{y}$, and this value never changes, and also initializes every one of the f_j's to zero. Then, it applies the cubic smoothing spline from (64), denoted as $\mathcal{S}_j$ above, to the residual $e_j = y - \beta_0 - \sum_{k \neq j} f_k$, as a function of x_{ij}, to obtain some new estimate f_j. This is done for each predictor in turn, using the current estimates of the functions when computing residuals. This process is continued until the estimates f_j stabilize, in the sense that the residual sum of squares is

$$\delta = \sum_{i=1}^{n} \left[y_i - \beta_0 - \sum_{i=1}^{p} f_j(x_{ij}) \right]^2 \leq \epsilon. \tag{104}$$

Then, the algorithm returns the latest estimates f_j, evaluated on, say, the grid of points in the input. In principle, the second step in the loop in Algorithm 1 is not needed, as the smoothing spline fit to a mean-zero response has mean zero [20]. However, in practice, machine rounding can cause slippage, and thus this minor adjustment is advised. Moreover, we can use other smoothing operators $\mathcal{S}_j$ too:

- other univariate regression smoothers such as local polynomial regression and kernel methods (with Gaussian kernels or nearest neighbor smoothers);
- linear regression operators yielding polynomial fits, piecewise constant fits, and series and Fourier fits (some fits here are discussed in Subsection 2.1);
- more complicated operators such as surface smoothers for second or higher-order interactions or periodic smoothers for seasonal effects for the model.

In the case of infinite samples, where this algorithm is applied for distributions, and the scatterplot smoothers are replaced with conditional expectations, it turns out to be guaranteed that the residual sum of squares does not increase at every step, and therefore must converge. In the case of finite samples, it was shown by Breiman and Friedman [3] that the following convergence properties will hold:

(a) Algorithm 1 converges for a restrictive and impractical set of smoothers.

(b) Algorithm 1 is mean square consistent for a slightly less restrictive class of smoothers in the following sense: if $(f_j)_n$ is the estimate of f_j applied to a finite sample with size n, and $(f_j)_\infty$ is the estimate of f_j applied to infinite samples (the distribution), then as $n \longrightarrow \infty$, the convergence below holds:

$$\mathbb{E}[(f_j)_n(x) - (f_j)_\infty(x)]^2 \longrightarrow 0. \tag{105}$$

The backfitting algorithm is popular for many reasons, one of which being that it gives high flexibility in terms of the choices of smooth components for the AM, allowing for these smooth components to come from a wide range of smoothing and modeling methods, such as regression trees [40]. However, while this natural flexibility certainly makes backfitting a very appealing method, one of the main drawbacks with backfitting is the difficulty in estimating smoothness. This turns out to be the case because both main methods to do this, i.e. cross validation for the partial residuals and directly estimating the hat matrix of the whole model, are quite computationally expensive tasks. Other disadvantages include the fact that the choice of when to stop the algorithm is arbitrary and it is hard to figure out, a priori, how long reaching a specific convergence threshold will take. Also, the model actually depends on the order in which the predictor variables are fit.

2.5 The Local Scoring Algorithm

Note that the backfitting algorithm can be extended to incorporate weights such as $\{q_i\}_{i=1}^n$ by transforming the data to have predictors $\{q_i x_i\}_{i=1}^n$, and then applying the backfitting algorithm to this transformed data. Thus, we use the expression

$$\text{Backfit}(\{(x_i, y_i), q_i\}_{i=1}^n)_j \tag{106}$$

to denote the jth function ($j = 1, \ldots, p$) in the output of Algorithm 1 with respect to the data $\{(x_i, y_i)\}_{i=1}^n$ and the weights $\{q_i\}_{i=1}^n$. Then, the local scoring algorithm, first introduced by Hastie and Tibshirani in 1987 [19], is defined below as follows:

Algorithm 2: Local Scoring Algorithm for Generalized Additive Models

```
2.1  initialize $\beta_0 \leftarrow g(\bar{y})$
2.2  initialize $f_j \leftarrow 0$ for $j = 1, \ldots, p$
2.3  repeat
2.4      for $i = 1, \ldots, n$ do
2.5          $\eta_i \leftarrow \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip})$
2.6          $\mu_i \leftarrow h(\eta_i)$
2.7          $z_i \leftarrow \eta_i + (y_i - \mu_i)(\partial\eta_i/\partial\mu_i)$
2.8          $v_i \leftarrow \mathbb{V}(y_i)$ given $\mu_i$
2.9          $q_i \leftarrow (\partial\mu_i/\partial\eta_i)^2/v_i$
2.10     $f_j \leftarrow \text{Backfit}(\{(x_i, z_i), q_i\}_{i=1}^n)_j$ for $j = 1, \ldots, p$
2.11     $\delta \leftarrow \sum_{i=1}^n \text{Dev}(y_i, \mu_i)/n$
2.12 until $\delta \leq \epsilon$
2.13 return $f_j$ for $j = 1, \ldots, p$
```

We will now explain what this algorithm does in words. Recall that a generalized AM of the form in (22) has a link function g with inverse $h = g^{-1}$. Now, Algorithm 2 takes in data, a grid of points, and a threshold value ϵ. It sets $\beta_0 = g(\bar{y})$ as well as initializes all the f_j's to zero. Then, for each $i = 1, \dots, n$, it gets an initial estimate

$$\eta_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), \tag{107}$$

sets $\mu_i = h(\eta_i)$, and calculates the working target variable z_i, which is defined by

$$z_i = \eta_i + (y_i - \mu_i)\frac{\partial \eta_i}{\partial \mu_i}. \tag{108}$$

Then, it sets v_i to be the variance of y_i at mean μ_i and calculates the weights q_i:

$$q_i = \left(\frac{\partial \mu_i}{\partial \eta_i}\right)^2 \frac{1}{v_i}. \tag{109}$$

Finally, it fits $f_j = \text{Backfit}(\{(x_i, z_i), q_i\}_{i=1}^{n})_j$ for each $j = 1, \dots, p$ and iterates this whole procedure until the deviance, defined, say, in McCullagh and Nelder [29],

$$\delta = \frac{1}{n}\sum_{i=1}^{n} \text{Dev}(y_i, \mu_i) \leq \epsilon. \tag{110}$$

Then, the algorithm returns the latest estimates f_j, evaluated on, say, the grid of points in the input. While the backfitting algorithm works for the fitting of AMs, it turns out that it is insufficient for the fitting of GAMs [19], which is why the local scoring algorithm is needed. We end with some observations and properties:

(a) In the Gaussian case, or when the link function, g, is the identity function, the local scoring algorithm is equivalent to the AM backfitting algorithm.

(b) The advantages and disadvantages of the backfitting algorithm (Algorithm 1) discussed in Subsection 2.4 of course apply to the local scoring algorithm.

(c) Both the backfitting and local scoring algorithms have limitations for large data-mining applications as they fit all predictors, which is not feasible [20].

This concludes our exposition on the frequentist approach to AMs, where broadly, we follow the textbooks by Wood [40] and Hastie, Tibshirani, and Friedman [20].

3 Gaussian Process Priors

In this section, and the next, we address the remaining question from our analysis of the penalized regression framework, namely how the basis functions for some additive model can be constructed in an inherently Bayesian way. The piecewise linear and cubic spline bases constructed in the previous section may work well in practice, but there was not an explicitly Bayesian motivation for why we chose those basis functions, and not others, in the first place. Here, we will follow the recent advancements by Solonen and Staboulis [37] to introduce function "priors."

In broad terms, Solonen and Staboulis imposed a Gaussian process prior for each function in the additive model, and for every one of these functions, in turn, the eigenfunctions of the Gaussian process can be used as a basis for the function. This hence results in the penalized regression form that we want, and from here, the penalized regression methods from the previous section shall fit the model.

There are many reasons why we would want to have a Gaussian process prior for our AM functions [37]. Firstly, as we have seen in Statistics 220, there exists a wide array and literature of different kernel functions available. The resulting ability we would have to choose between these numerous kernel functions makes our model rather flexible. Moreover, with the theory of Gaussian processes within our grasp, it is clear what kind of functions the prior yields, plus sampling from the prior is a straightforward process. Furthermore, it is easy to restrict, beforehand, the range of allowed function values, and because each functions is given its own mean value, there are no additional identifiability issues beyond the one that is given in the previous section. Also, periodicity and symmetry are easy to encode.

3.1 Notation for This Section

For Section 3 and Section 4, we would like to stick close to the notation that we have introduced before this, but at the same time make it more general to handle the additive model in a slightly more general form. Namely, our data will now be

$$\{(x_i, y_i)\}_{i=1}^n, \quad x_i = (x_{i1}, \dots, x_{ip})^t, \tag{111}$$

where each x_{ij} is now allowed to be a vector. Then, our additive model shall be

$$y_i = f_1(x_{i1}, \beta_1) + \cdots + f_p(x_{ip}, \beta_p) + \epsilon_i, \quad i = 1, \dots, n, \tag{112}$$

where the f_j are smooth functions, x_{ij} are the input observations for the function f_j, β_j are the vectors of parameters for the function f_j, and the ϵ_i are measurement errors. Note the key differences between this new formulation and the old one:

(a) We no longer have a constant term β_0, for simplicity.

(b) We explicitly encode the parameters β_j on which f_j depends on.

(c) The x_{ij}'s are now vectors, so each f_j could be multivariable.

(d) A priori, we do not assume a Normal distribution for the ϵ_i's.

This model, when compared to our previous specification, is, in some sense, both more general and more restrictive at the same time. However, either specification still sticks within the family of additive models, and in particular, does not stray into the realm of the generalized additive model. Of course, most ideas which we introduce here will work for the case of the generalized additive model, keeping in mind the caveat about model fitting using PIRLS mentioned in Subsection 1.4.

On top of this, we shall consider one more restriction on the model specified in (112), simply for notational convenience. Namely, we will require that the f_j's are such that the full model is linear with respect to the parameters $\beta = (\beta_1, \dots, \beta_p)^t$ (but can be nonlinear with respect to their respective inputs x_{ij}). Then, we further impose Normal prior distributions on the unknown parameter values β, which we will want to estimate. In other words, we want the full model for the estimation of β to always be expressible in the following compact form that is shown below:

$$y = X\beta + \epsilon, \tag{113}$$

$$B\beta \sim \mathcal{N}(\mu_{\text{pr}}, \Gamma_{\text{pr}}), \tag{114}$$

where $y = (y_1, \dots, y_n)^t$, $\epsilon = (\epsilon_1, \dots, \epsilon_n)^t$, and X and B are matrices. If we further suppose that the measurement errors are Normal, i.e. $\epsilon \sim \mathcal{N}(0, \Gamma_{\text{obs}})$, we can then realize this model as a linear Normal system, and by Normal-Normal conjugacy,

$$\beta \mid y \sim \mathcal{N}(\mu_{\text{pos}}, \Gamma_{\text{pos}}), \tag{115}$$

$$\Gamma_{\text{pos}}^{-1} = X^t \Gamma_{\text{obs}}^{-1} X + B^t \Gamma_{\text{pr}}^{-1} B, \tag{116}$$

$$\mu_{\text{pos}} = \Gamma_{\text{pos}} (X^t \Gamma_{\text{obs}}^{-1} y + B^t \Gamma_{\text{pr}}^{-1} \mu_{\text{pr}}). \tag{117}$$

An initial worry one might possess about the above approach is that the quantities μ_{pos} and Γ_{pos} may be hard to compute, as these computations involve multiplying matrices as well as inverting matrices, which are both computationally expensive operations. Nevertheless, Solonen and Staboulis [37] remark that the matrices X and B are often sparse, and plus, the noise covariances are usually assumed to be diagonal matrices, so the above system can be solved in an efficient way by way of using methods in sparse linear algebra, even if the data and parameters are of a large dimension. They also mention that the posterior precision matrix $\Gamma_{\mathrm{pos}}^{-1}$ is usually sparse, so it can then be used, for instance, to sample from the posterior distribution $\beta \mid y$ in an efficient way. This formulation will be sufficiently general to describe all the models we introduce in this section and the next. When we go into specifics, we shall then also describe how to write the model in the form we described above. As an easy special case, in the standard linear model (1), where

$$f_j(x_{ij}, \beta_j) = \beta_j x_{ij}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, p, \tag{118}$$

we simply obtain $X = (z_1, \ldots, z_p)$, where $z_j = (x_{1j}, \ldots, x_{nj})^t$ is the typical design matrix [37]. Also, the matrix B that is multiplied to the left of β in the prior for our specification is included to account for when we do not want to penalize the unknown function values directly, but instead want to penalize another quantity related to these function values. As an example, in Section 4, we would want to penalize the differences between consecutive unknown function values in some spatial grid [37]. Here, B is usually not of full rank, and therefore not invertible. This makes the prior improper, but this is not a problem as long as the posterior distribution can be made proper. This indeed happens in the example where we set priors on the differences between consecutive function values but then do not penalize the function values directly. Next, we shall show how to write each f_i in terms of a basis. For $i = 1, \ldots, p$ and each index j for the jth component of f_i,

$$(f_i)_j = \sum_{k=1}^{K_i} \alpha_{ik} b_{ik}(x_{ij}), \tag{119}$$

where now the unknown parameters β_i related to the function f_i are the weights α_{ik}. Then, the design matrix X can be written in the following "stacked" form:

$$X = \begin{pmatrix} \overbrace{b_{11}(x_{11}) \ \cdots \ b_{1K_1}(x_{11})}^{f_1} & \cdots & \overbrace{b_{p1}(x_{p1}) \ \cdots \ b_{pK_p}(x_{p1})}^{f_p} \\ b_{11}(x_{12}) \ \cdots \ b_{1K_1}(x_{12}) & \cdots & b_{p1}(x_{p2}) \ \cdots \ b_{pK_p}(x_{p2}) \\ \vdots & \vdots & \vdots \\ b_{11}(x_{1n}) \ \cdots \ b_{1K_1}(x_{1n}) & \cdots & b_{p1}(x_{pn}) \ \cdots \ b_{pK_p}(x_{pn}) \end{pmatrix}. \tag{120}$$

At times, we will also add a translation to Equation (119) to obtain the modified

$$(f_i)_j = m_i(x_{ij}) + \sum_{k=1}^{K_i} \alpha_{ik} b_{ik}(x_{ij}), \tag{121}$$

where m_i is the translation we will associate to the smooth function f_i. Here, each m_i should be a known function of the input data, and we can basically interpret this as a sort of "mean function." Of course, if we implement this addition, the X design matrix stays the same, but to be able to have our model be of the form in (113), it is necessary to subtract the translations induced by the m_i from the data.

3.2 Gaussian Processes and Kernels

Before using Gaussian processes to construct the basis functions b_{ik} that we have introduced above, we'll use this subsection to briefly review Gaussian processes (and the notation we use for this), as well as introduce some useful kernels that we plan to focus on for the rest of this section. To start, [31] gives the definition:

Definition 3.1. A Gaussian process is a collection of random variables such that any finite number of these possess a Multivariate Normal distribution.

Such a Gaussian process is completely specified by its mean and covariance function, $m(x)$ and $k(x, x')$, and we write $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. These are just:

$$m(x) = \mathbb{E}[f(x)], \tag{122}$$
$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{123}$$

In Statistics 220, we learned that any function $k(x, x')$ that satisfies the inequality

$$\iint g(x)g(x')k(x, x')\, dxdx' \geq 0 \tag{124}$$

for all square-integrable functions g, i.e. Mercer's condition, is some valid kernel function. However, choosing the kernel determines almost all the generalization properties of a Gaussian process model, and indeed, it has been found historically that some kernels perform much better than others. Because we have to specify a kernel function for our Gaussian process prior later, we have decided to list a range of choices of kernel functions first, as well as explain various properties of these kernel functions, following the exposition in [9]. Firstly, we will introduce

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right), \tag{125}$$

the squared exponential kernel (also known as the radial basis function kernel or the Gaussian kernel). This has become the de-facto default kernel for a Gaussian process model mainly because it has some rather elegant properties. Namely, this kernel is universal, in the sense that it can approximate an arbitrary continuous target function uniformly on any compact subset of the input space (see [30] for a more precise definition of universal). Moreover, Duvenaud [9] argues that this kernel could be integrated against most functions, every function in its prior has infinitely many derivatives, and it has only two parameters. For this last aspect:

(a) The lengthscale ℓ determines the length of the "wiggles" in the function; in general, one cannot extrapolate more than ℓ units away from the data.

(b) The output variance σ^2 will determine the average distance of the function away from its mean; this is just the generic scale factor of the kernel.

Next, we look at a three-parameter kernel, namely the rational quadratic kernel:

$$k_{\mathrm{RQ}}(x, x') = \sigma^2 \left(1 + \frac{\|x - x'\|^2}{2\alpha\ell^2}\right)^{-\alpha}. \tag{126}$$

This kernel is equivalent to summing together many squared exponential kernels with different lengthscales, and whence, Gaussian process priors with this kernel are suitable for predicting functions which vary smoothly across different lengthscales. The third new parameter α determines the relative weighting of the large-scale and small-scale variations, and also, as $\alpha \longrightarrow \infty$, this rational quadratic kernel is identical to the squared exponential kernel [9]. Duvenaud describes the pitfalls of the squared exponential and rational quadratic kernels as follows: if the function happens to have a discontinuity or is discontinuous in the first few derivatives (like the absolute value function), then either the lengthscale will be extremely short and the posterior mean would zero everywhere, or the posterior mean will have "ringing" effects [9]. Moreover, even if there does not exist any hard discontinuities, the lengthscale would usually be determined by the smallest "wiggle" in the function. Thus, there will be failure to extrapolate within smooth regions if there is even a small nonsmooth region in the data. Next, we introduce

$$k_{\mathrm{PER}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/\phi)}{\ell^2}\right), \tag{127}$$

the periodic kernel. This kernel allows one to model functions that repeat themselves exactly, i.e. periodic functions. Such functions naturally show up as angles or time periods. The only new parameter here not yet seen is the period ϕ, which determines the distance between repetitions of the function. Next, sometimes it is needed to model functions which possess symmetry; that is, functions f satisfying $f(x) = f(-x)$. The symmetric kernel, which will do this, could be constructed from some existing kernel function $k(x, x')$ by the following construction below:

$$k_{\text{SYM}}(x, x') = k(x, x') + k(-x, x'). \tag{128}$$

We mention one last kernel: the Ornstein-Uhlenbeck kernel which is defined by

$$k_{\text{OU}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|}{\ell}\right), \tag{129}$$

which is used in place of the squared exponential or rational quadratic kernels when one believes that these latter kernels are too smooth (notice that the main difference of the Ornstein-Uhlenbeck kernel above is that the squared Euclidean distance function is replaced by an absolute distance function). To conclude, we note that kernel functions can be both multiplied and added together. Multiplying two kernels together is the standard way to combine two kernels, and it can be viewed of as an "AND" operation. Namely, the resulting multiplied kernel would have a high value only if both of the two base kernels have a high value. On the other hand, adding two kernels together is less standard (though notice that we have did it to define the symmetric kernel!), and it can be thought of as an "OR" operation. Basically, this means that the resulting added kernel would possess a high value if either of the two base kernels have a high value. We will utilize the notations and kernels defined in this subsection, as well as the idea to multiply and add multiple kernel functions together, freely in the upcoming exposition.

3.3 Global Gaussian Process Bases

We are now ready to construct basis functions for our additive model (112) using Gaussian process priors. Namely, we will set priors on each function f_i as follows:

$$f_i(x, \beta) \sim \mathcal{GP}(m_i(x), k_i(x, x')), \tag{130}$$

where we specify the model parameters β below. To actually perform computations with these functions, it is standard practice to discretize f_i on a fixed grid of input points, when we get $\mathrm{f}_i \sim \mathcal{N}(\mu_{f_i}, \Sigma_{f_i})$, where μ_{f_i} and Σ_{f_i} are defined by

$$\mu_{f_i} = (m_i(a_1), \ldots, m_i(a_L)), \tag{131}$$
$$(\Sigma_{f_i})_{jk} = k_i(a_j, a_k), \tag{132}$$

where $a_1, \dots, a_L$ form the grid of input points we specify. With this, the key idea introduced by Solonen and Staboulis [37] is to compute the eigenvalue decomposition for the covariance matrix Σ_{f_i}, and then employ this to identify f_i in the basis form given in (121). Namely, we shall be able to decompose the matrix Σ_{f_i}:

$$\Sigma_{f_i} = \sum_{r=1}^{L} \lambda_r q_r q_r^t = PP^t, \tag{133}$$

where λ_r is the rth eigenvalue of Σ_{f_i} with corresponding eigenvector q_r, and the rth column of the matrix P, p_r, is the rth eigenvector scaled by the its eigenvalue:

$$p_r = \sqrt{\lambda_r} q_r, \quad r = 1, \dots, L. \tag{134}$$

Now, with the representation definition of a Multivariate Normal random variable (see the Statistics 210 textbook by Blitzstein and Morris [2] for this definition),

$$\mathrm{f}_i = \mu_{f_i} + P\beta, \quad \beta \sim \mathcal{N}(0, I_L), \tag{135}$$

which can be recognized as the basis form given in (121). Explicitly, the parameters for our model, β, are the weights for the basis vectors (with the convenient i.i.d. Normal prior), the translation is given by μ_{f_i}, and the basis vectors would be encoded by P [37]. Hence, we have functions that are defined on a grid of input points, and if we want to evaluate these functions at observed locations, all that needs to be done is to interpolate to these locations [37]. From this, Solonen and Staboulis remark that in infinite-dimensional terms (that is, without picking the explicit grid of points for evaluation) what we would like to do given a function $f_i(x, \beta) \sim \mathcal{GP}(m_i(x), k_i(x, x'))$ is to find its eigenfunctions, and use these as the basis vectors for our additive model functions. The approach that we mentioned, namely (133), (134), and (135), is simply a discretization-interpolation trick which can be used as a way to approximate these eigenfunctions in practice. However, the textbook by Rasmussen and Williams [31] mention other approaches which can also be taken to obtain these desired eigenfunctions. Another approximation approach in this textbook is the Nystrom method (in Chapter 8.3.2), and further, in some special cases where we have sufficiently nice kernels, there are analytically available eigenfunctions (in Chapter 4.3). We address a key drawback next.

With the approach outlined in this subsection, note that the number of basis vectors obtained (i.e. the basis dimension) is precisely the number L of grid points chosen for representing the f_i following a Gaussian process as the vector f_i. Thus, to have a sufficiently fine grid so that the function f_i looks continuous, we would end up with an enormously high basis dimension! Of course, this is undesirable as it would be computationally expensive (or in most cases, impossible) to store the resulting matrix P, and plus, such a model is very prone to overfitting anyway. Fortunately, there is a solution to this problem, namely the well-known "truncated SVD" method which is frequently used in inverse problems where one typically encounters high-dimensional estimation problems (see [28] for a reference). Now, in our context, this translates to a straightforward way for making the dimension of the basis significantly smaller by way of removing the basis vectors which do not contribute much to the covariance. Specifically, out of the L eigenvalues

$$\lambda_1, \ldots, \lambda_L, \tag{136}$$

we can select the first K eigenvalues $\lambda_1, \ldots, \lambda_K$ such that, for instance, we obtain

$$(\lambda_1 + \cdots + \lambda_K)/(\lambda_1 + \cdots + \lambda_L) > 0.9999. \tag{137}$$

Then, we can consider the corresponding eigenvalues $q_1, \ldots, q_K$, and their scaled

$$p_1 = \sqrt{\lambda_1} q_1, \ldots, p_K = \sqrt{\lambda_K} q_K \tag{138}$$

counterparts. The vectors $p_1, \ldots, p_K$ can then be used as the columns for a matrix P_K, and with this, we can approximate $\Sigma_{f_i} \approx P_K(P_K)^t$. By this construction, the truncated covariance will contain 99.99% of the original covariance; we can use

$$\mathrm{f}_i = \mu_{f_i} + P_k\beta, \quad \beta \sim \mathcal{N}(0, I_K), \tag{139}$$

as the parameterization corresponding to (135), and thus estimate only K weights instead of the original L. In most cases, $K \ll L$, i.e. K will be much smaller than L, so this method reduces the dimension of our problem dramatically. One thing Solonen and Staboulis notes is that the success of this dimensionality reduction technique depends on how smooth the selected Gaussian process prior is [37]. In other words, the smoother we assume the unknown function $f_i(x, \beta)$ to be, then the lower the dimension shall be for the resulting estimation when we utilize the truncated SVD method of (137), (138), and (139). This approach gets particularly useful when the number of independent input variables (i.e. the dimension of x_{ij}) is larger than one. We reason through why this is the case in the next subsection.

3.4 Considerations in High Dimensions

Another potential issue arises with this Gaussian process approach, namely that evaluating functions on a grid of points becomes harder and harder as the dimension of the input space increases. In more precise language, we can say that the number of grid points grows as an exponential function with respect to the total number of variables in the input space. As an example, if the input space turns out to be three-dimensional for one of the functions in our AM and we use fifty grid points in each direction, we would need $50^3 = 125000$ points in total. First off, even storing the high-dimensional covariance matrices that arise is impossible for most computing systems, and what's even worse, one usually cannot employ methods from sparse matrix algebra: most of the typically used kernel functions, including the ones given in Subsection 3.2, result in dense covariance matrices.

The solution suggested by Solonen and Staboulis [37] is to restrict the model to only consider separable kernels for the Gaussian process priors on AM functions and then to apply the dimensionality reduction approach introduced at the end of Subsection 3.3. To start, we have the following definition for the term separable.

Definition 3.2. A kernel function is said to be separable if it can be expressed as a product of kernel functions whose input vectors are of a lower dimension.

To be concrete in the exposition ahead, consider an arbitrary two-dimensional function in our AM, so that its prior is $f(x, \beta) \sim \mathcal{GP}(m(x), k(x, x'))$, where the relevant entries are two-dimensional. Then, if $k(x, x')$ is separable, we can express

$$k(x, x') = k_1(x_1, x_1')k_2(x_2, x_2'), \quad x = (x_1, x_2), \quad x' = (x_1', x_2'), \tag{140}$$

where k_1 and k_2 are kernel functions with one-dimensional inputs. Then, we will discretize the first component using m_1 grid points and the second component using m_2 grid points. We can calculate the one-dimensional covariance matrices

$$\Sigma_1 \in \mathbb{R}^{m_1 \times m_1} \quad \text{and} \quad \Sigma_2 \in \mathbb{R}^{m_2 \times m_2} \tag{141}$$

that are associated to the kernels k_1 and k_2, respectively, and then, the question is: how does one obtain the covariance matrix Σ associated to k using Σ_1 and Σ_2 alone? To answer this question, we will introduce a new operation, the Kronecker product, and explain some of its properties. We follow Horn and Johnson [21].

Definition 3.3. Let A be an $m \times n$ matrix and B a $p \times q$ matrix (the notation used here is independent of notation used in this paper). Then, the Kronecker product of A and B, denoted $A \otimes B$, is the $pm \times qn$ block matrix that is displayed below:

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{pmatrix}. \tag{142}$$

More explicitly, if we suppose that $\star$ and $\diamond$ denote truncating integer division and remainder, respectively, we can write the elements of $A \otimes B$ directly in terms of the elements of A and the elements of B according to the following formula:

$$(A \otimes B)_{ij} = A_{i\star p, j\star q} B_{i\diamond p, j\diamond q}, \tag{143}$$

assuming that as a convention, we number the matrix elements starting from 0.

The Kronecker product can be looked at intrinsically in terms of abstract linear algebra as well: if A and B represent the linear maps $V_1 \longrightarrow W_1$ and $V_2 \longrightarrow W_2$ of vector spaces, respectively, then $A \otimes B$ in fact represents the induced linear map

$$V_1 \otimes V_2 \longrightarrow W_1 \otimes W_2 \tag{144}$$

of tensor product spaces. From the definition of the Kronecker product, it could then be checked, albeit tediously so, that the covariance matrix in the joint space is simply the Kronecker product of the two individual covariance matrices, i.e.

$$\Sigma = \Sigma_1 \otimes \Sigma_2 \in \mathbb{R}^{m_1 m_2 \times m_1 m_2}, \tag{145}$$

under a suitable ordering of the variables. To further our analysis and compute the basis elements of the joint space, we would need to find the eigendecomposition of Σ. This cannot be done in a direct way (recall that we cannot store Σ!), but it can be done indirectly using the eigendecompositions of Σ_1 and Σ_2. In this regard, Horn and Johnson [21] derive two very useful properties of Kronecker products.

Theorem 3.4. *The Kronecker matrix product satisfies the following two properties.*

(a) *If A, B, C, and D are matrices of the correct sizes so that the matrix products AC and BD are well-defined, then $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ holds.*

(b) *Transposition is distributive over the Kronecker product: $(A \otimes B)^t = A^t \otimes B^t$.*

With this, we see that if $\Sigma_1 = Q_1 \Lambda_1 Q_1^t$ and $\Sigma_2 = Q_2 \Lambda_2 Q_2^t$ are eigendecompositions for Σ_1 and Σ_2, respectively, then $\Sigma = \Sigma_1 \otimes \Sigma_2$ has the below eigendecomposition:

$$\Sigma = (Q_1 \Lambda_1 Q_1^t) \otimes (Q_2 \Lambda_2 Q_2^t) = (Q_1 \otimes Q_2)(\Lambda_1 \otimes \Lambda_2)(Q_1 \otimes Q_2)^t \tag{146}$$

using (a) and (b) of Theorem 3.4. This equation yields following the interpretation: the eigenvectors of Σ are just Kronecker products of the individual eigenvectors of Σ_1 and Σ_2, while the eigenvalues of Σ are just products of the eigenvalues for Σ_1

and Σ_2 (after a suitable ordering of vectors and values). Moreover, it is clear that for dimensions higher than two, the method outlined above works in exactly the same way, except that we will now have to take (possibly many) more Kronecker products! When the truncated SVD method is now applied to each input variable before taking Kronecker products, what results is a vastly reduced dimensionality for the overall joint space of estimated unknowns; this is precisely what we want.

3.5 A Small Selection of Examples

We end with some examples illustrating these techniques, displayed in Figure 5.

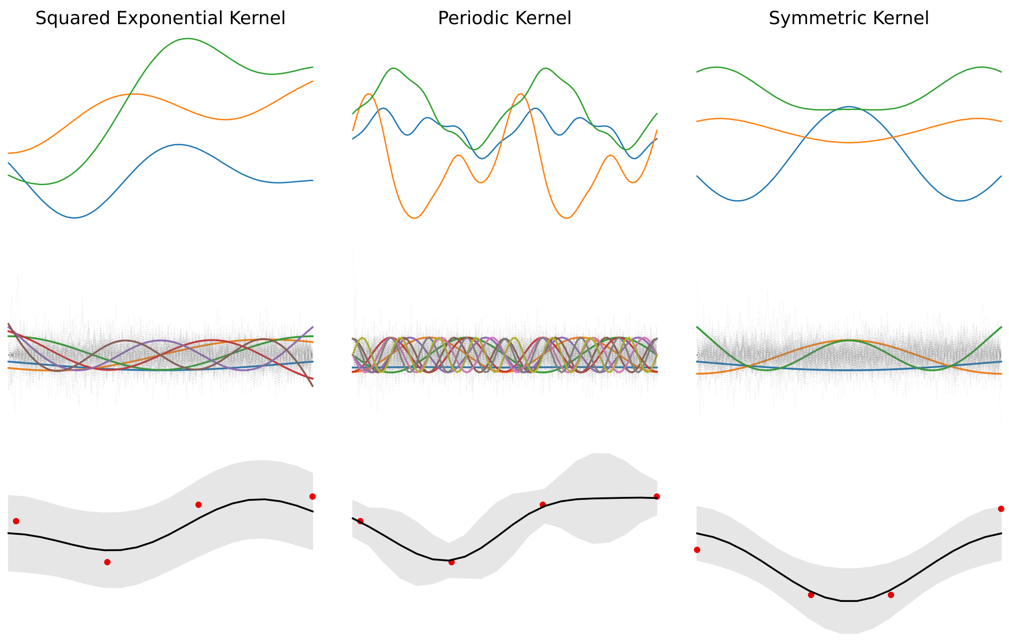


Figure 5: Random samples (top row), basis vectors (middle row), and simple fitting examples (bottom row), for $k_{\mathrm{SE}}(x, x')$ (first column), $k_{\mathrm{PER}}(x, x')$ (second column), $k_{\mathrm{SYM}}(x, x')$ with respect to $k_{\mathrm{SE}}(x, x')$ (third column). The dashed lines in the second row plots show the basis functions that are dropped out from the estimation. The final row compares the mean and ± two standard deviations that are calculated from the reduced basis vectors.

We see that samples and basis vectors from the periodic and symmetric kernels are, as expected, periodic and symmetric, respectively. We also infer that with a strongly smoothing kernel, only a reasonably small number of basis vectors will remain after we drop basis vectors using the truncated SVD method, as expected.

4 Splines and Difference Priors

Having looked at how Gaussian process priors can give rise to a natural basis for the functions in an additive model, we now have a global approach to the issue of finding naturally Bayesian-occuring basis constructions. However, as we have mentioned in Section 1, the sometimes we would also want to have at hand some local approach, as it is usually the case that local approaches describe sharp local features much better. We have then also seen in Section 2 that spline bases have many special properties that make them the ideal interpolators and smoothers.

Therefore, we would like to find a spline basis for the functions in our additive model that is naturally linked to some prior, thus connecting our model further to the Bayesian framework of statistics. In the literature (see Subsection 1.3), we decided to explore further an approach which uses a B-spline basis coupled with a difference prior applied directly to the parameters to control function wiggliness.

Thus, we will follow the textbook by Wood [40] to first introduce the B-spline basis and the difference priors associated to it (this is Section 5.3.3 in Wood). Then, we will generalize this approach to higher dimensions following the general linear framework introduced in Subsection 3.1 using the ideas that were recently given by Solonen and Staboulis [37]. To begin, Wood remarks that the B-spline basis is appealing because the functions in this basis are local in the strictest sense: each basis function will only be nonzero over the intervals between the $m+3$ adjacent knots, where $m+1$ is the order of this basis (so $m = 2$ for a cubic spline). Whence, to define a B-spline basis with k dimensions, we first define $k+m+2$ knots,

$$x_1 < x_2 < \cdots < x_{k+m+1} < x_{k+m+2}, \tag{147}$$

where the interval for spline evaluation is $[x_{m+2}, x_{k+1}]$, and the first and last $m+1$ knot locations are essentially arbitrary. Then, an $(m+1)$-th order spline will be

$$f(x) = \sum_{i=1}^{k} B_i^m \beta_i, \tag{148}$$

where the B-spline basis functions are defined recursively in the following way:

$$B_i^{-1}(x) = \begin{cases} 1 & x_i \leq x < x_{i+1}, \\ 0 & \text{otherwise}. \end{cases} \tag{149}$$

$$B_i^m(x) = \frac{x - x_i}{x_{i+m+1} - x_i} B_i^{m-1}(x) + \frac{x_{i+m+2} - x}{x_{i+m+2} - x_{i+1}} B_{i+1}^{m-1}(x), \tag{150}$$

for $i = 1, \dots, k$. As is observed within de Boor [7], these B-splines were introduced as a very stable basis for large scale spline interpolation, but for most statistical inference with low rank penalized regression splines, it turns out that many poor numerical methods have to be used before the enhanced stability of the basis will become noticeable [40]. It turns out that the really interesting statistical work to arise from B-splines comes from the work of Eilers and Marx [11] who employed B-splines along with difference priors. This setup is a low rank smoother using a B-spline basis, usually defined on evenly spaced knots, along with a difference prior applied directly to the parameters β_i to control function wiggliness. Here, the best way to know how this setup works is by way of an example: suppose we decide to penalize the squared difference between adjacent β_i values, so that

$$\mathcal{P} = \sum_{i=1}^{k-1} (\beta_{i+1} - \beta_i)^2 = \beta^t P^t P \beta \tag{151}$$

is the penalty, where the matrix P is defined so that the following equalities hold:

$$P = \begin{pmatrix} -1 & 1 & 0 & \cdot & \cdot \\ 0 & -1 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \quad P\beta = \begin{pmatrix} \beta_2 - \beta_1 \\ \beta_3 - \beta_2 \\ \cdot \\ \cdot \end{pmatrix}, \quad \mathcal{P} = \beta^t \begin{pmatrix} 1 & -1 & 0 & \cdot & \cdot \\ -1 & 2 & -1 & \cdot & \cdot \\ 0 & -1 & 2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \beta. \tag{152}$$

Next, we see how these B-splines fit into the framework by Solonen and Staboulis.

4.1 Difference Priors in One Dimension

To start, we note that although the estimation problem for spline-based additive models can be rather high-dimensional, in the sense that we usually select a more fine than necessary grid of knot locations and then use a smoothness penalty to pick the correct smoothness level, this high dimensionality is usually not a huge issue. After all, this difficulty is counteracted by the fact that spline basis functions are nonzero only close to the knot location in question, so sparse linear algebra can be used in the relevant matrix computations [37]. For the one-dimensional case, suppose we restrict to using first order B-splines as the building blocks for our additive model (so in the preceding exposition, we set $m = 0$). Here, Solonen and Staboulis note that this is equivalent to treating the function values as the direct unknowns, and then performing linear interpolation to the regions which are in between the selected input points [37]. In the framework (113), we would have the matrix X as the sparse linear interpolation matrix that sends the function values (at the chosen input grid) to the observation locations. An example is

$$X = \begin{pmatrix} 0 & \cdots & w_1 & 1-w_1 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & w_2 & 1-w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \tag{153}$$

where w_i is the distance of the input observation to the first input grid value that it exceeds. In other words, if the input grid is $x_g = (x_{g,1}, \dots, x_{g,L})$, then X is just

$$X_{ij} = \begin{cases} x_i - x_{g,j} & x_{g,j} \le x_i \le x_{g,j+1}, \\ 1 - (x_i - x_{g,j}) & x_{g,j-1} \le x_i \le x_{g,j}, \\ 0 & \text{otherwise.} \end{cases} \tag{154}$$

Now, in contrast to focusing on second order smoothness penalties as we did in Section 2 or directly restrict the function values with Normal priors, we shall use the approach we introduced at the start of Section 4 and penalize the changes in the values of the function (i.e. the derivatives). Then, if we approximate, in a grid, the derivatives of the functions, we will get sparse difference matrices as we did in (151) and (152). Examples for the first three orders of difference matrices for a one-dimensional input grid of length $L = 5$ is shown (as D_1, D_2, and D_3) below:

$$D_1 = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}, \tag{155}$$

$$D_2 = \begin{pmatrix} -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \end{pmatrix}, \quad D_3 = \begin{pmatrix} -1 & 3 & -3 & 1 & 0 \\ 0 & -1 & 3 & -3 & 1 \end{pmatrix}. \tag{156}$$

Here, we purposefully choose to ignore the so-called "boundary conditions" for the difference operators here, and only choose to penalize the differences, not the function values. As alluded to after (118), when these difference matrices are to be used as our B matrices in (113) and (114), we will get improper Normal priors for $B\beta$. However, this is not an issue as long as we can collect enough data, when the posterior distribution will surely be proper and our analysis could proceed.

We can observe the phenomenon that different orders for the difference priors will behave in rather different ways. In general, while all choices of order shall give similar fits for the data range, the extrapolation behavior would be different. For instance, Figure 6 displays some example using first, second, and third order differences for a simple function approximation task. Beyond the data range, the first order prior extrapolates to a constant, the second order prior extrapolates to a linear function, and the third order prior extrapolates to a quadratic function.

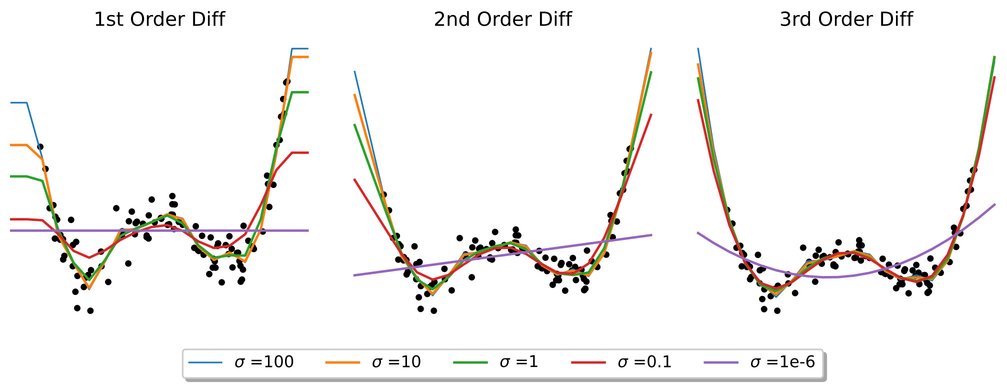


Figure 6: Fitting data generated around the true function $f(x) = 3x^4 - 6x^2 + 2$ using 3 different orders for the difference priors and 5 different prior variances (see the legend).

Choosing the right order for the difference priors is clearly a task which depends on the application at hand, and moreover, the optimal fitting is sometimes gotten by a mixture of various orders of difference priors. For example, we can use the mixture of first and second order difference priors if we know a one-dimensional function is expected to extrapolate linearly in one direction but saturate to some constant in the other direction. We look at high-dimensional considerations next.

4.2 Difference Priors in Higher Dimensions

To now extend our spline bases from Subsection 4.1 to higher dimensions, we shall first follow Section 5.6.1 of Wood's textbook [40] to extend spline bases to some higher dimension using tensor product bases. The approach here is to start with smooths of a single variable, represented using any basis which has an associated quadratic penalty measuring "wiggliness" of the smooth. Whence, from marginal smooths, a construction using tensor products is used to build up smooths with several variables. We will introduce this concept using the construction for some smooth function of three covariates, x, z, and v, but this trivially generalizes to any number of covariates. Suppose we have low rank bases for representing the smooth marginal functions f_x, f_z, and f_v of each of the covariates. Put differently, assume we can express $f_x(x)$, $f_z(z)$, and $f_v(v)$ with the following sums below:

$$f_x(x) = \sum_{i=1}^{I} \alpha_i a_i(x), \tag{157}$$

$$f_z(z) = \sum_{\ell=1}^{L} \delta_\ell d_\ell(z), \tag{158}$$

$$f_v(v) = \sum_{k=1}^{K} \beta_k b_k(v), \tag{159}$$

where the α_i, δ_ℓ, and β_k are parameters, and the $a_i(x)$, $d_\ell(z)$, and $b_k(v)$ are known basis functions. (The notation used here will be isolated to just this part of Subsection 4.2 to avoid conflict with the same notation used elsewhere.) With this, we look at how f_x can be converted into a smooth function of x and z. We'll need f_x to vary smoothly with z, and this can be achieved by allowing its parameters, α_i, to vary smoothly with z. Using the basis for smooth functions of z, we obtain

$$\alpha_i(z) = \sum_{\ell=1}^{L} \delta_{i\ell} d_\ell(z), \tag{160}$$

$$f_{xz}(x, z) = \sum_{i=1}^{I} \sum_{\ell=1}^{L} \delta_{i\ell} d_\ell(z) a_i(x). \tag{161}$$

Proceeding in the same way, we'll extend f_{xz} to a smooth function of x, z, and v:

$$\delta_{i\ell}(v) = \sum_{k=1}^{K} \beta_{i\ell k} b_k(v), \tag{162}$$

$$f_{xzv}(x, z, v) = \sum_{i=1}^{I} \sum_{\ell=1}^{L} \sum_{k=1}^{K} \beta_{i\ell k} b_k(v) d_\ell(z) a_i(x). \tag{163}$$

Furthermore, for any set of observations of x, z, and v, we can relate the model matrix, X, evaluating the tensor product smooth at these observations, and the model matrices X_x, X_z, and X_v, evaluating the marginal smooths at the same set of observations. Then, given an appropriate ordering of the $\beta_{i\ell k}$ into a vector β,

$$X = X_x \odot X_z \odot X_v \tag{164}$$

follows directly from the definition of the row-wise Kronecker product operation, denoted $\odot$, which we define below (this is different to the Kronecker product $\otimes$).

Definition 4.1. Let A be an $n \times m$ matrix and C a $n \times p$ matrix. Let C_i denote the ith row of C. Then, the row-wise Kronecker product, denoted $A \odot C$, is given by

$$A \odot C = \begin{pmatrix} A_{11}C_1 & \cdots & A_{1n}C_1 \\ \vdots & \ddots & \vdots \\ A_{n1}C_n & \cdots & A_{nm}C_n \end{pmatrix}. \tag{165}$$

This is a $n \times mp$ matrix, which is often also known as the Khatri-Rao product [40].

Wood [40] notes that this whole construction so far can be continued for as many covariates as needed, the result is independent of the order in which the covariates are treated, and the covariates themselves can be vector covariates in this method.

Returning to the first order splines and difference priors from Subsection 4.1, we see that we can use the technique of tensor product bases to extend our linear interpolation model to multiple dimensions in a straightforward way. Namely:

(a) We decide on a suitable ordering of the many variables that we possess.

(b) We vectorize the high-dimensional function values following this ordering.

(c) We construct the interpolation matrix by finding the relevant differences.

As an example, the first row of a two-dimensional interpolation matrix that uses column ordering would have four nonzero values and assume the following form:

$$X_{1\cdot} = (0, \ldots, 0, w_{11}, w_{12}, 0, \ldots, 0, w_{13}, w_{14}, 0, \ldots, 0), \tag{166}$$

where the w_{1i}'s are simply the distances to the corners of the "square pixel" where the first observation falls in, and the constraint $\sum_i w_{1i} = 1$ is satisfied [37]. Here, observe that each row in an L-dimensional interpolation matrix would have 2^L nonzero elements. The difference priors for these interpolation matrices (for some fixed order) can also be extended to multiple dimensions in the same way. Here, for constructing the priors, Solonen and Staboulis [37] remark that it might make more sense to define the priors separately to the direction of each dimension (or even specify some different order for the differences for different dimensions).

As a concrete example, suppose that we attempt to fit data randomly generated around some true function $z = f(x, y)$, say the true function that is shown below:

$$z = f(x, y) = \frac{1}{2}x + \frac{4(y - 1/2)^2}{1 + 2x}. \tag{167}$$

This function is quadratic in the second dimension y, with curvature varying as a function of x, and behaves rather linearly in the first dimension x. If we fit a function with second order difference prior in the first dimension and third order difference prior in the second dimension, we obtain a fit that extrapolates nicely along both dimensions. This resulting fit is illustrated below within Figure 7.

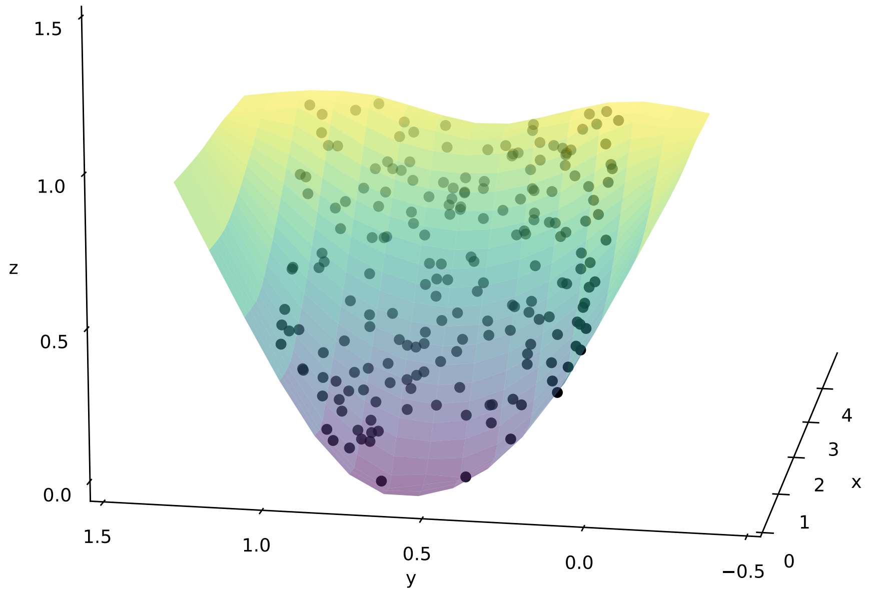


Figure 7: Fitting data generated around the true function $f(x,y) = 0.5x + 4(y-0.5)^2/(1+2x)$ using a second order difference prior for x and a third order difference prior for y.

Solonen and Staboulis comment that this approach of using difference priors for approximating unknown functions is a widely studied topic in the field of inverse problems. As an example of this application, Haario, Laine, Lehtinen, Saksman, and Tamminen [17] discuss difference priors in the context of atmospheric sensing, and they also demonstrate how to make these difference priors discretization independent. One can also refer to Bardsley [1], Rue and Held [33], and also the references within these sources for discussion about Gaussian Markov random fields (GMRFs) and their connections to difference priors within inverse problems.

4.3 Spatially Varying Smoothness

One benefit of this local basis approach is that there is another level of flexibility that is not used in other AM approaches. Specifically, it is possible to tune prior variances of the difference priors as a way to achieve what Solonen and Staboulis call “spatially varying smoothness.” By this term, they mean that when estimating a function which has nonconstant smoothness, specifying a prior variance that is suitable for one region of the function might work poorly for different regions of the function. The solution is to “spatially vary” the smoothness by assigning different prior variances for different parts of the function. This phenomenon can be seen clearly, for example, in the three plots of Figure 8 that are shown below.

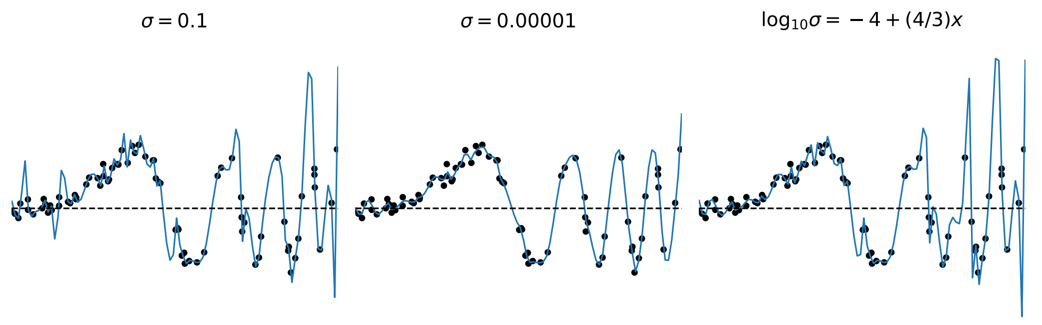


Figure 8: Fitting data generated around the true function $f(x) = \sin(x^3)$ using 3 different ways of defining the prior variance. For the first two plots, the prior variance is fixed at a constant value, while in the third/last plot, the prior variance is made a function of x.

In the figure above, the goal was to fit points around the true function given by

$$f(x) = \sin(x^3), \tag{168}$$

which clearly has nonconstant smoothness. We see that increasing the variance of the prior via $\log_{10}\sigma = -4 + (4/3)x$ as a function of x yielded a better fit to the data compared with keeping the prior variance fixed, as expected. Of course, one thing of note is that we have chosen the behavior of the prior variance manually such that the fit looks good, and automatically estimating how the prior variance should vary with each dimension is a highly tricky thing to do, and is covered in more detail in Section 7 of Solonen and Staboulis [37]. Moreover, one particular case where spatially varying smoothness will dramatically improve the resulting fit of the additive model is when there is a sudden jump in the function. This type of behavior can be induced by setting the prior variance to be high in the vicinity of the discontinuity. For an example, consider the function f with sudden jump

$$f(x) = \sin(x) + x\mathbb{1}(x > 2) \tag{169}$$

at $x = 2$. Here, $\mathbb{1}(A)$ is simply the indicator function that equals 1 if A occurs and equals 0 otherwise. We can fit two additive models with B-spline bases as well as difference priors, the first one with constant smoothness and the second one with varying smoothness. The results of this fit are displayed below in Figure 9.

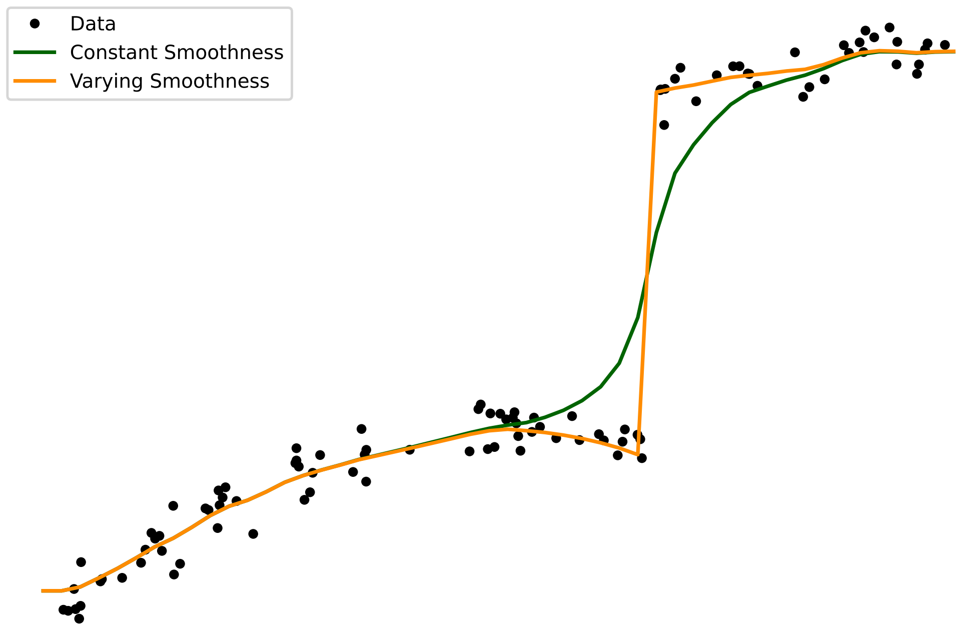


Figure 9: Fitting data generated around the true function $f(x) = \sin(x) + x\mathbb{1}(x > 2)$ in 2 different ways: fixing the prior variance vs. increasing the prior variance around $x = 2$.

4.4 Fixing Identifiability Issues

One might notice that we did not mention the identifiability issue that continues to resurface when there are multiplie additive components in the same model, i.e. when we can add a constant value to the first function and subtract this value from the second and still end up with the same predictions and penalties. Indeed, the Gaussian process approach to additive modeling does not suffer this problem: this prior penalizes the function values directly, and each function converges to their own specified mean values [37]. But, with the local basis function approach, where only the derivatives are penalized, this identifiability issue is very present. To resolve this issue, we would proceed as we have suggested many times before, following the textbook by Wood [40], to add some constraint to the estimation problem so that the smooth components should sum to zero, and then add have some intercept which picks up the correct level. This works, and has the additional benefit of making sure that the confidence bands for the functions stay narrow.

4.5 Periodic and Symmetric Priors

Finally, we mention how to require periodicity and symmetry with our local basis approach with difference priors. Doing so was rather straightforward using the Gaussian process prior: we simply needed to choose an appropriate kernel with the desired periodic/symmetric properties. However, with the approach outlined in this section, we would need to include additional rows in the difference prior matrices to obtain such a behavior. We address each of these properties below.

To have periodicity in our prediction, all we need to do is introduce an extra row within the prior system which stores the difference between the first and last function values [37]. To additionally match derivatives as well, for the one-dimensional case, it suffices to add the following rows to the prior system B:

$$D_{\mathrm{PER}} = \begin{pmatrix} -1 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & -1 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix}. \tag{170}$$

Note that the "strength" of the periodicity assumption is controllable via tuning the relevant prior variances, and custom periods for these "periodicity priors" can also be gotten by calculating the differences between the correct function values. Doing this is more manually tedious than with the Gaussian prior approach, for which the period is directly a parameter in the kernel function, but on the other hand, difference priors offer more flexibility because it is not needed to specify a restrictive kernel function in the first place as is required with Gaussian processes.

To have symmetry in our prediction, we would need to penalize the "correct differences" computed on both sides of the desired axis of symmetry. For example, with a six-dimensional input grid of points, symmetry about the direct middle of the domain can be imposed by setting the prior system matrix B as the following:

$$B = \begin{pmatrix} 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{171}$$

As we can see, enforcing either property on our function is difficult in this local basis function with difference prior setup: a notable weakness of this method.

5 Applications

Having studied the theory of additive modeling in depth, both from a frequentist and a Bayesian perspective, we now turn our attention to applying special cases of the models we have introduced in various empirical investigations. The prime goal of this data analysis, beyond understanding and experiencing how such AMs are implemented in practice, is to find out how the different variations of the same additive modeling approach perform when benchmarked against the same type of data, as well as to figure out which properties of the data yield these differences in model performance. So, this analysis deepens our understanding of AMs further.

For the purposes of this goal, we have curated three different datasets to test AM performance on, as well as considered nine model specifications out from the theoretical material presented in Sections 2, 3, and 4. To measure performance after model fitting, we have appealed to four different statistics, namely the mean squared error (MSE), the root-mean-square error (RMSE), the mean absolute error (MAE), and the coefficient of determination (R^2). This last statistic may, in some intuitive sense, be more informative than the first three statistics in our regression analysis evaluations since it is a percentage whereas the other measures possess arbitrary ranges, but we include all four statistics for completeness. Thus, in what follows, we will introduce the nine models we consider (in Subsection 5.1), and for each of the datasets (in Subsections 5.2, 5.3, and 5.4), we shall exhibit a form of visualization for the data, and then move forward with presenting the results from our model fits as well as plotting these fits (except in Subsection 5.4 where there were too many datapoints for a plot to give any sensible insights). Finally, we conclude with our insights and interpretation of the obtained results. All the code for this section, as well as for the rest of the paper, is in the repository below:

https://github.com/zadchin/BayesianAM

5.1 Models Under Consideration

To start, we will list the nine models that we will test (on our three datasets) and give a brief description of the model, referencing previous equations and sections.

(1) LinearModel: as a simple benchmark model, we shall test a linear model of the form in Equation (1), running the ordinary least squares method. Since all three datasets are nonlinear, we expect this model to perform poorly in comparison to the nonlinear, and whence more flexible, models in our list.

(2) FrequentistAM: we will consider a model using B-splines introduced at the start of Section 4 (that recall is a more general case of the cubic splines we introduced in Subsection 2.2) under the penalized regression framework of Section 2 using second derivative smoothing as our penalty. This model is to represent the frequentist approach to additive modeling from Section 2.

(3) SquaredExpGP: this shall be a Gaussian process prior model of the form in (112), (113), (114), and (130) introduced from Section 3 and Subsection 3.3, using the squared exponential kernel function $k_{\mathrm{SE}}(x, x')$ of (125) introduced in Subsection 3.2. We use the eigendecomposition method of (136) to (139).

(4) RationalQuadGP: this shall be a Gaussian process prior model of the form in (112), (113), (114), and (130) introduced from Section 3 and Subsection 3.3, using the rational quadratic kernel function $k_{\mathrm{RQ}}(x, x')$ of (126) introduced in Subsection 3.2. We use the eigendecomposition method of (136) to (139).

(5) OrnsteinUhlGP: this shall be a Gaussian process prior model of the form in (112), (113), (114), and (130) introduced from Section 3 and Subsection 3.3, with the Ornstein-Uhlenbeck kernel function $k_{\mathrm{OU}}(x, x')$ of (129) introduced in Subsection 3.2. We use the eigendecomposition method of (136) to (139).

(6) SmoothDP: we use a model using B-splines and one-dimensional difference priors, as outlined in Section 4 and Subsection 4.1. This model still has the form given in (112), (113), and (114), with the matrix B chosen to be of the form given in (155) or (156), where the order of the difference matrix shall be 2 for a more flexible extrapolation behavior (see, for instance, Figure 6). Here, we only have the smoothness prior as our choice of difference prior.

(7) PeriodicDP: we use a model using B-splines and one-dimensional difference priors, as outlined in Section 4 and Subsection 4.1. This model still has the form given in (112), (113), and (114), with the matrix B chosen to be of the form given in (155) or (156), where the order of the difference matrix shall be 2 for a more flexible extrapolation behavior (see, for instance, Figure 6). Here, we will incorporate some periodicity into the difference prior, but in different ways. For the first two datasets, we consider both a smooth and a periodic component, while in the last dataset, we use only a periodic prior.

(8) MixtureDP: we use a model using B-splines and one-dimensional difference priors, as outlined in Section 4 and Subsection 4.1. This model still has the form given in (112), (113), and (114), with the matrix B chosen to be of the

form given in (155) or (156), where the order of the difference matrix shall be 2 for a more flexible extrapolation behavior (see, for instance, Figure 6). Here, we will incorporate a mixture of properties into the difference prior depending on the dataset. For the first two datasets, we consider a smooth plus periodic plus symmetric mixture, while in the last dataset, we only use a smooth and periodic mixture as symmety is not a reasonable assumption.

(9) RandomForest: finally, as a more sophisticated but less interpretable model for benchmarking purposes, we will apply a random forest regression (with cross validation as necessary) to all our datasets. The theory for such models can be found in Chapter 15 of [20], and we use this Python implementation.

Whenever possible, we'll try to tune the hyperparameters of these models using the method of Bayesian optimization (see Garnett's textbook [13] for a reference), or if the dimension is sufficiently small, we will use a simple grid search instead.

5.2 Fitting Data From Synthetic Functions

Our dataset here is 50 datapoints sampled with scaled Normal noise around the true function $y = -\sin^3(x) + \cos^3(x)$, for $x \in [-\pi, \pi]$. We view this in Figure 10.

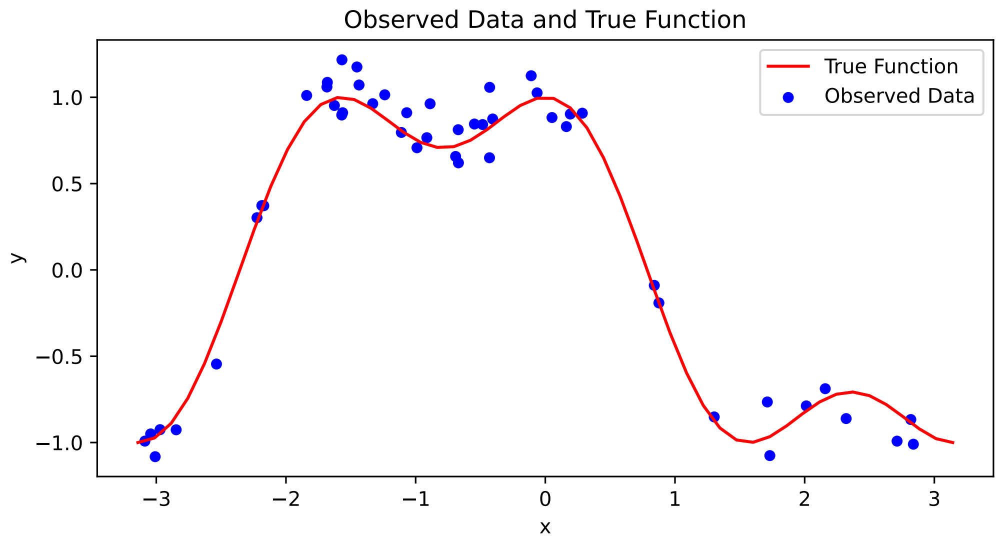


Figure 10: Fifty datapoints sampled around the true function $y = -\sin^3(x) + \cos^3(x)$ in the domain $x \in [-\pi, \pi]$ with scaled $0.1\mathcal{N}(0, 1)$ Normal noise for y values given x values.

Of course, LinearModel gives a rather poor fit to the data, as shown in Figure 11.

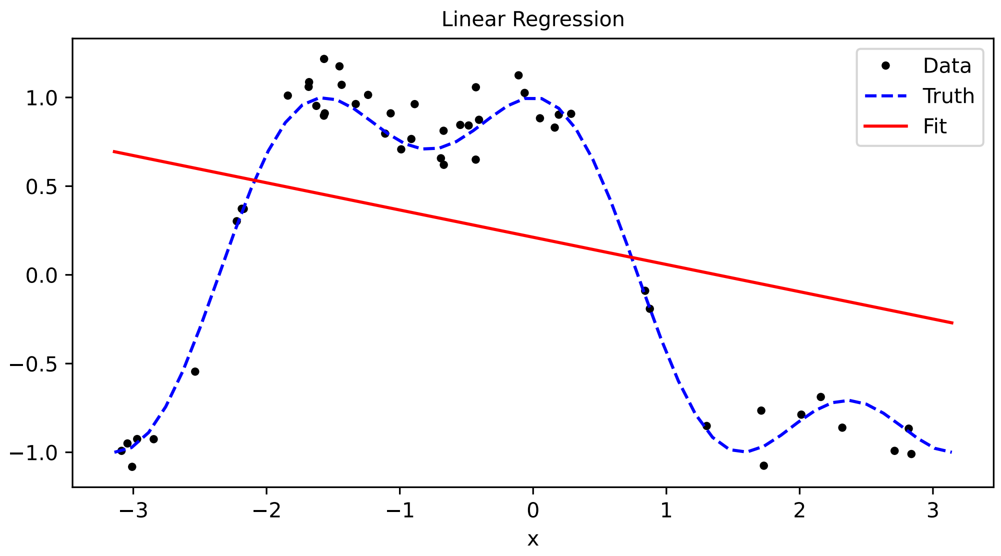


Figure 11: The fit produced by LinearModel on our synthetic dataset.

On the other hand, FrequentistAM produced a better fit, as is seen in Figure 12.

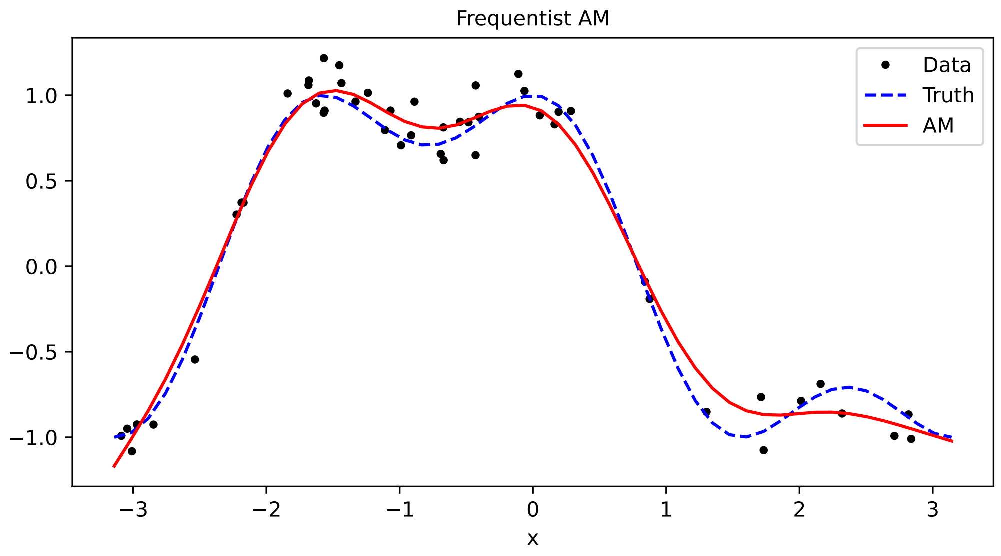


Figure 12: The fit produced by FrequentistAM on our synthetic dataset.

Fits for SquaredExpGP, RationalQuadGP, and OrnsteinUhlGP are in Figure 13. These are plotted side-to-side in the three graphs that we display right below this.

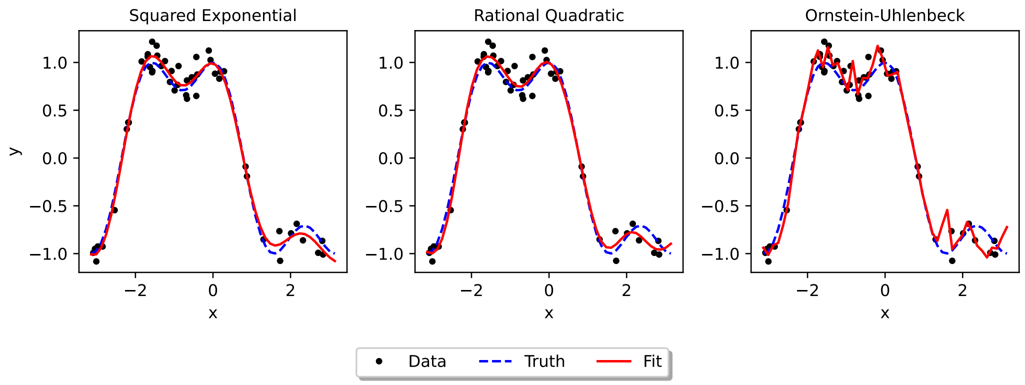


Figure 13: Fits by SquaredExpGP, RationalQuadGP, and OrnsteinUhlGP on our dataset.

Notice that these fits get progressively more “jagged” going from left to right: the smoothness of the kernels used decreases in this direction. Moving on, the fitting of the models SmoothDP, PeriodicDP, and MixtureDP are displayed in Figure 14.

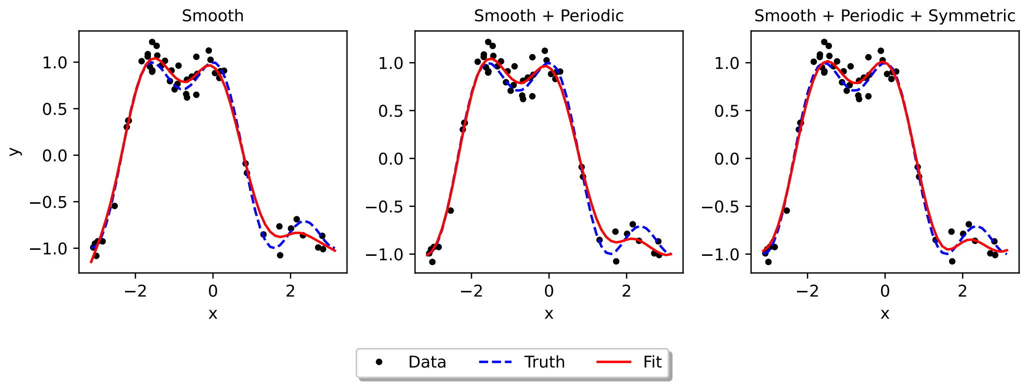


Figure 14: Fits by SmoothDP, PeriodicDP, and MixtureDP on our dataset.

We see that there are very few differences between the above three fits, with the exception that the second fit is periodic (so that the values and derivatives at the end of the domain match), and that the third fit is symmetric about a line roughly at $x = -0.8$. This visually improves the fit since the trigonometric functions, sine and cosine, which make up the true function are intrinsically both periodic and symmetric. Finally, the fit which resulted from RandomForest is shown in Figure 15. This fit is clearly of a rather jagged nature, suggesting a possible overfitting.

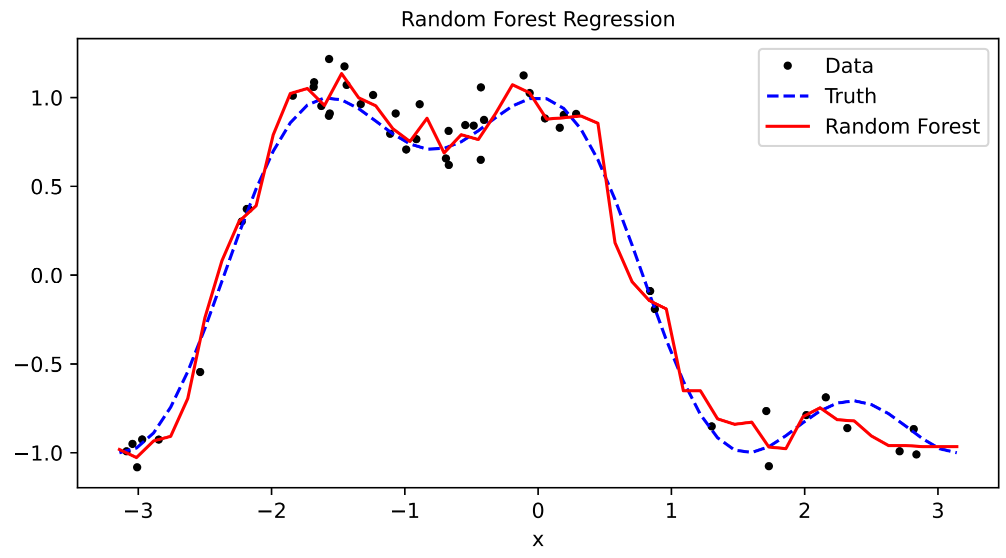


Figure 15: The fit produced by RandomForest on our synthetic dataset.

The numerical results obtained from all of these fits is shown in the table below.

Model	MSE	RMSE	MAE	R^2
LinearModel	0.56602	0.75235	0.66527	0.10453
FrequentistAM	0.00902	0.09496	0.07896	0.98573
SquaredExpGP	0.00301	0.05483	0.04816	0.99524
RationalQuadGP	0.00401	0.06329	0.05389	0.99366
OrnsteinUhlGP	0.01922	0.13864	0.09894	0.96959
SmoothDP	0.00674	0.08209	0.06725	0.98934
PeriodicDP	0.00690	0.08304	0.06522	0.98909
MixtureDP	0.00431	0.06568	0.05368	0.99318
RandomForest	0.01208	0.10993	0.09154	0.98088

As we expected, the linear regression, LinearModel, performs the worst out of all of the tested models since the data is nonlinear. Meanwhile, the best performing model is SquaredExpGP, followed by RationalQuadGP. This intuitively makes a lot of sense since the true function $y = -\sin^3(x) + \cos^3(x)$ is smooth (that is, it's differentiable to all orders), and thus best approximated by the models that will assume the most about how smooth the function is. Interestingly, we infer that RandomForest performs rather poorly, as does FrequentistAM. This may indeed be attributed to the generality of these methods, which make them perform worse than the other models we used which were more suited to infinitely differentiable functions: the difference prior models (SmoothDP, PeriodicDP, and MixtureDP) turned out to perform really well, just third, lagging behind the top two models.

5.3 Fitting Data From the mcycle Dataset

Next, we study mcycle, a dataset introduced by Silverman [36], where we examine the relationship between time and the acceleration of one's head in a simulated motorcycle accident. This is often used to test the robustness and efficacy of crash helmets. A plot of the 133 datapoints within this dataset is shown in Figure 16.

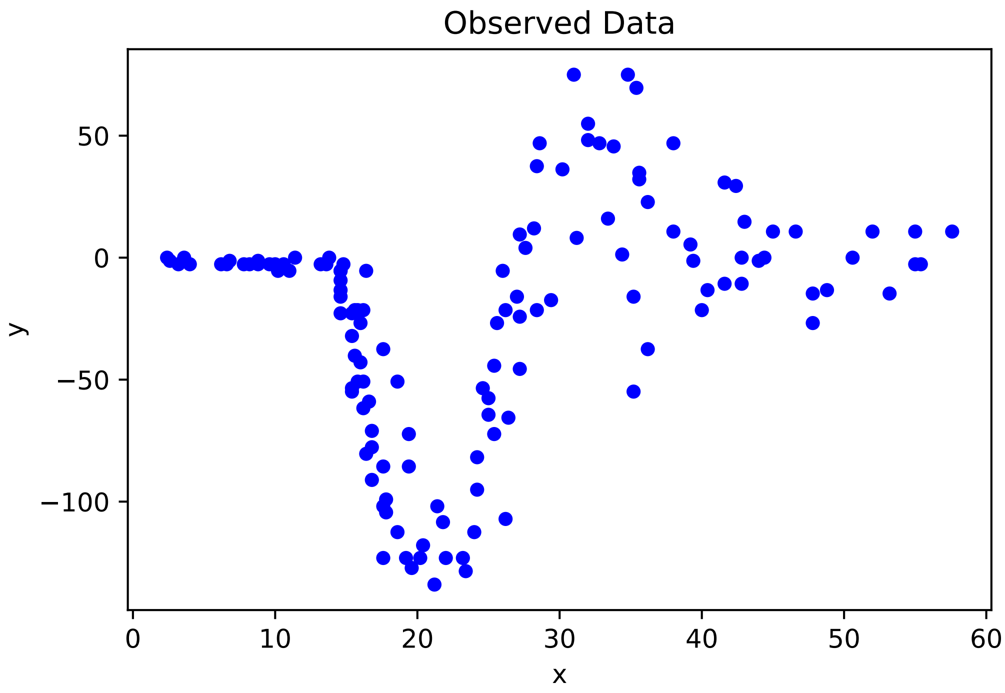


Figure 16: All 133 datapoints from the mcycle dataset plotted.

The response variable is accel, the acceleration of the head measured in the unit

G-force, while the explanatory covariate is times, measured in milliseconds from the time of impact. The fit produced by LinearModel is again bad; see Figure 17.

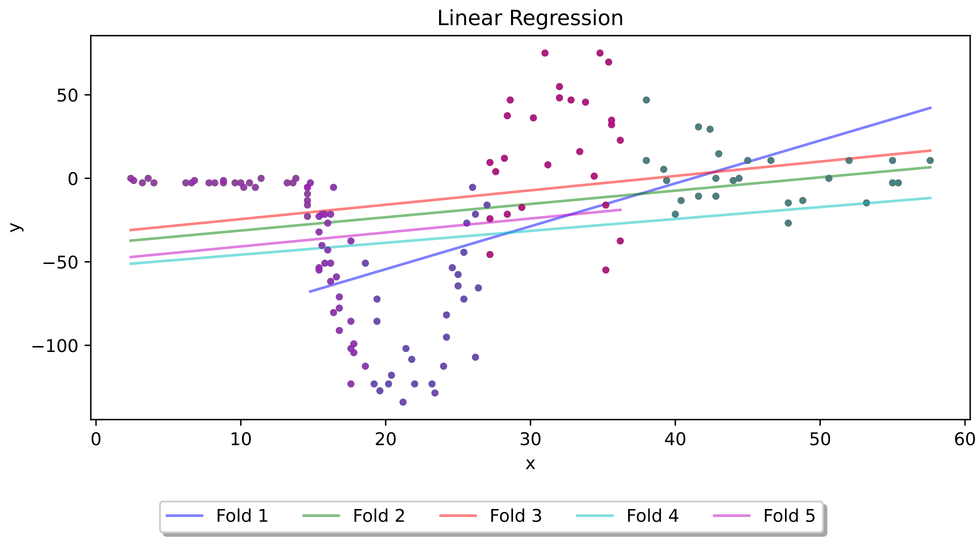


Figure 17: The fit produced by LinearModel on the mcycle dataset.

Meanwhile, FrequentistAM again produces a nicer fit, as we can see in Figure 18.

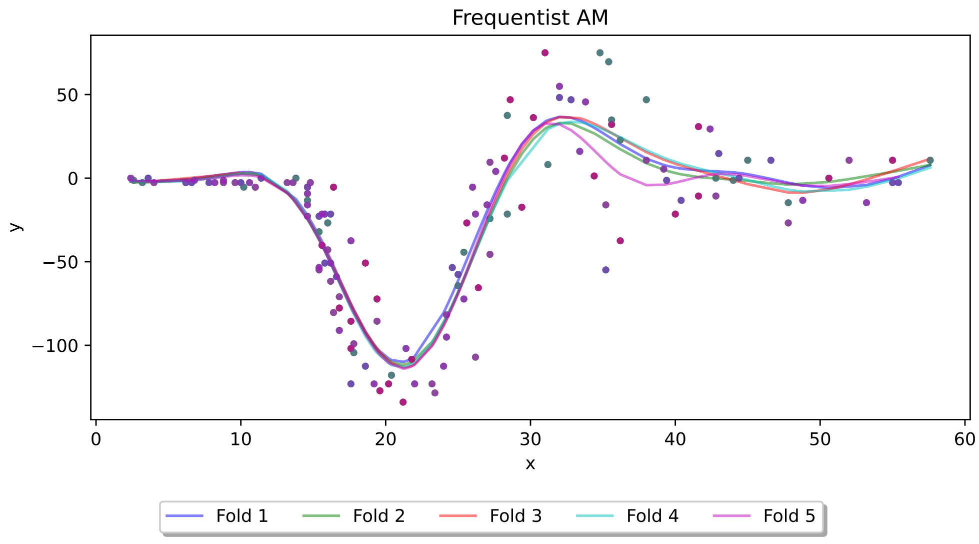


Figure 18: The fit produced by FrequentistAM on the mcycle dataset.

All this while, observe that we have used 5-fold cross validation to produce a less biased or less optimistic estimate, and thus counteract overfitting. Moving ahead, fits for SquaredExpGP, RationalQuadGP, and OrnsteinUhlGP are in Figure 19.

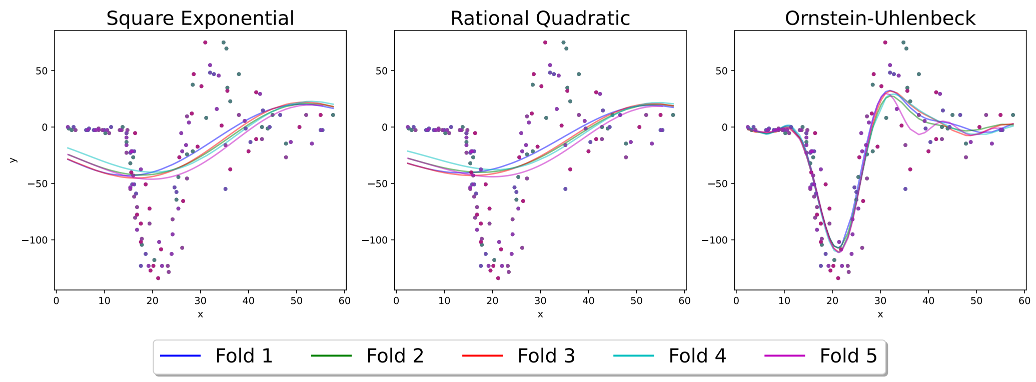


Figure 19: Fits by SquaredExpGP, RationalQuadGP, and OrnsteinUhlGP on the dataset.

Observe that this time, the Ornstein-Uhlenbeck kernel results in a way better fit than the other two more smooth kernels, the squared exponential plus rational quadratic kernels, which this time resulted in a rather bad fit of the data. Indeed, this could be because the mcycle dataset possesses some quite “sharp” features in its overall shape, which ruins the Gaussian process fits when the kernel is too smooth. The fitting of SmoothDP, PeriodicDP, and MixtureDP are in Figure 20.

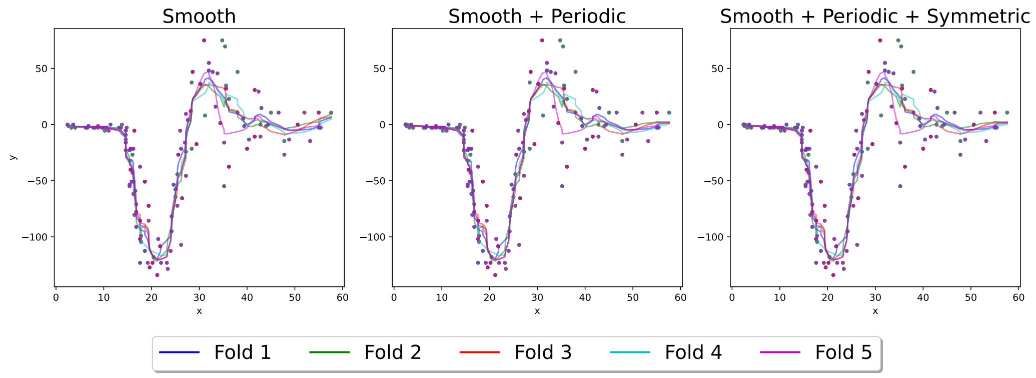


Figure 20: Fits by SmoothDP, PeriodicDP, and MixtureDP on the dataset.

These fits are considerably more jagged than their counterparts in Subsection 5.2, because indeed the true function underlying the data from this dataset should be far less smooth. This is justified by the fit of our random forest regressor, which recall is RandomForest, being far more jagged than before, as seen in Figure 21.

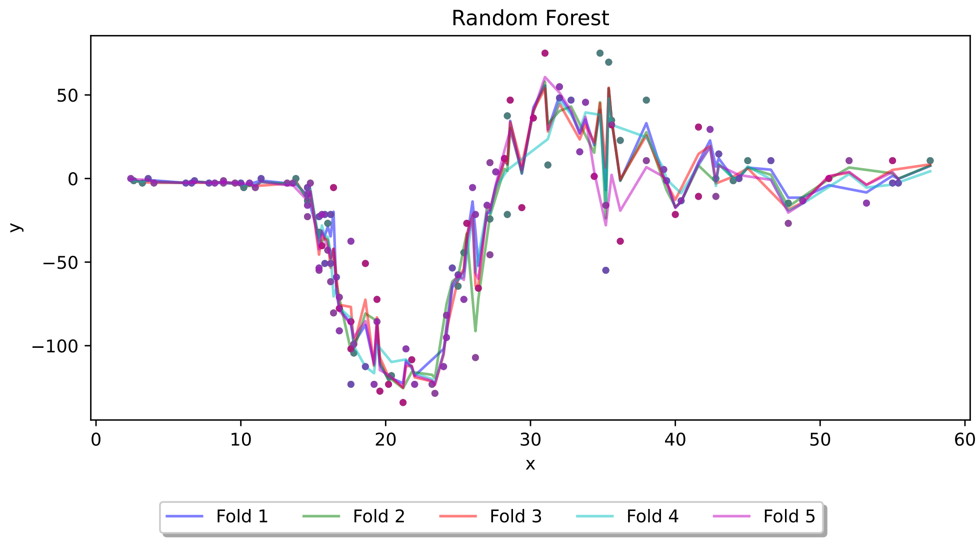


Figure 21: The fit produced by RandomForest on the mcycle dataset.

The numerical results obtained from all of these fits is shown in the table below.

Model	MSE	RMSE	MAE	R^2
LinearModel	1962.3170	44.05878	35.97921	0.10951
FrequentistAM	476.69544	21.81958	16.57895	0.79347
SquaredExpGP	1762.6589	41.96837	34.64343	0.23659
RationalQuadGP	1826.0479	42.71584	35.01356	0.20919
OrnsteinUhlGP	487.54859	22.06440	16.87112	0.78877
SmoothDP	379.33748	19.45626	13.71573	0.83592
PeriodicDP	379.99529	19.47301	13.78276	0.83563
MixtureDP	379.99529	19.47301	13.78276	0.83563
RandomForest	205.31152	14.32060	9.777730	0.91103

As we expected again, the linear regression, LinearModel, is the worst-performing model out of all the models we tested. Of course, the mcycle dataset is obviously nonlinear, and the last two subsections thus show a clear need for methods that extend beyond the well-understood linear model. In this case, it turns out that the best-performing model is RandomForest. This is not too surprising since we have used a robust fitting with 5-fold cross validation, and the sophisticated nature of a random forest regressor made it fit the data in this case, where it is not clear if the true function can be approximated by sums of smooth functions at all. Here, the runner-up is the class of difference prior models (that consists of SmoothDP, PeriodicDP, and MixtureDP), whose local flexibility allowed for a decent fitting of the data. Here, the periodic and symmetric properties did not influence much.

In stark opposition to the results in Subsection 5.2, the models which used a Gaussian process prior and a smooth kernel, SquaredExpGP and RationalQuadGP, performed rather poorly, only besting LinearModel. As we hinted to earlier, this is probably the case because the underlying true function modeling acceleration of the head during motorcycle crash accidents is probably not only nonlinear, but also nonsmooth, in the sense that it is not differentiable at some points. Therefore, the Ornstein-Uhlenbeck kernel Gaussian process prior model, i.e. OrnsteinUhlGP, performed much better than the other Gaussian process prior models, partly due to the fact that the $k_{\mathrm{OU}}(x, x')$ kernel does not assume smoothness of the function as much as the other two kernels used for SquaredExpGP and RationalQuadGP.

5.4 Predicting Air Quality in Beijing

Finally, to really highlight and leverage the "additive" structure of additive models, we'll now examine the relationship between various air pollutant measurements and the resulting fine particulate matter ($\mathrm{PM}_{2.5}$) value. To do this, we will consider a dataset measuring air quality in Beijing supplied by the UCI Machine Learning Repository and first introduced in a 2017 paper by Zhang et al. [8, 44]. The dataset includes hourly air pollutants data from 12 nationally-controlled air quality sites. This data is collected by the Beijing Municipal Environmental Monitoring Center, and the meteorologial data in each air quality site is compared with the nearest weather station from the China Meteorological Administration [8]. This data was collected from March 1, 2013 to February 28, 2017. Missing data is denoted NA.

The response variable is the $\mathrm{PM}_{2.5}$ concentration, and in particular, we shall be interested in predicting the $\mathrm{PM}_{2.5}$ value of tomorrow given all the data of today. There are nine explanatory covariates included in this dataset, as is shown below:

- $\mathrm{PM}_{2.5}$: the particulate matter 2.5 value.
- PM_{10}: the particulate matter 10 value.
- SO_2: the sulfur dioxide concentration.

- NO_2: the nitrogen dioxide concentration.
- CO: the carbon monoxide concentration.
- O_3: the ozone concentration.
- PRES: the ambient air pressure.
- RAIN: the rain precipitation value.
- WSPM: the ambient wind speed.

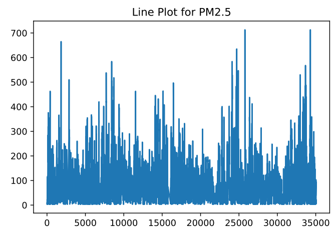
The value we are trying to predict is tomorrow's $PM_{2.5}$, which we will denote by Tomorrow$PM_{2.5}$. The given dataset has 12 subdatasets, which correspond to the 12 air quality collection sites in Beijing. For simplicity, we only consider the data collected at the Aoti Zhongxin site (which is more commonly known as Olympic Park) for the whole duration of data collection. After some data cleaning (that is, after dropping all NA rows and string columns), we then obtain 31815 datapoints.

We begin by performing some exploratory data analysis. The first action we carry out is to produce line plots for each of the covariates and the response; we provide these plots in Figure 22. From these, we see that there is some periodicity among some of the covariates, namely SO_2, CO, O_3, and PRES. We can also plot histograms of the covariates and the response; these are presented in Figure 23.

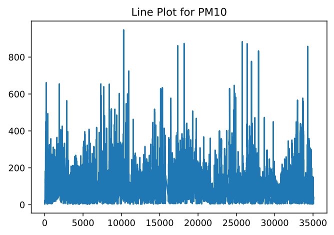
From these histograms, we see that most covariates are right-skewed, including $PM_{2.5}$, PM_{10}, SO_2, NO_2, CO, O_3, RAIN, and WSPM. This suggests that some form of normalization and/or standardization of the data is required. Moreover, we can see that there are some outlier (very high or very low) values as well, so the standardization method we use should be robust towards such outliers (as an example, a z-score standardization is rather unsuitable for data with outliers).

We can also plot a heatmap showing the values, and relative intensity, of the pairwise correlation coefficients between each of the nine covariates and also the response; this is shown in Figure 24. Here, observe that some covariates possess a greater than half correlation coefficient value with the response: this includes $PM_{2.5}$ (0.96), PM_{10} (0.84), NO_2 (0.65), and CO (0.76), but not the other covariates.

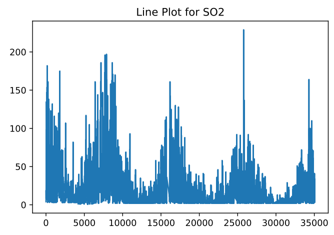
Having seen these plots, we make the decision to lognormalize the data plus perform a robust scaling as can be found here; this preserves outliers on a relative scale and contributes no loss of information. The resulting scaled histograms can be seen in Figure 25. We may clarify this robust scaling process further: for some generic y_i value inside the dataset $y = (y_1, \ldots, y_n)$, which we assume is already



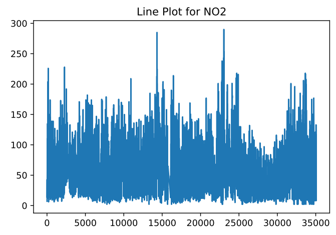
(a) $PM_{2.5}$.



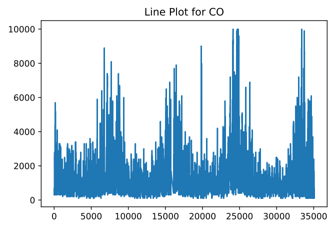
(b) PM_{10}.



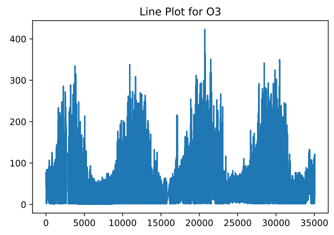
(c) SO_2.



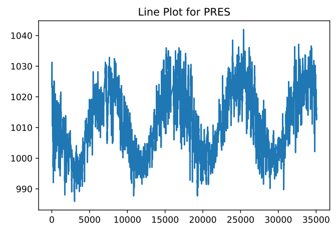
(d) NO_2.



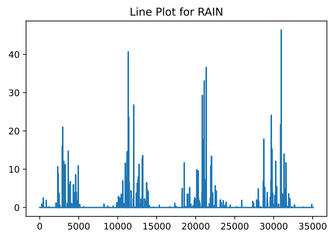
(e) CO.



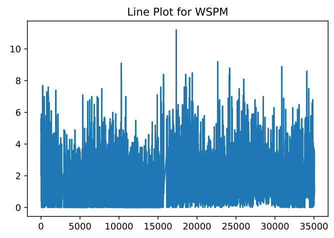
(f) O_3.



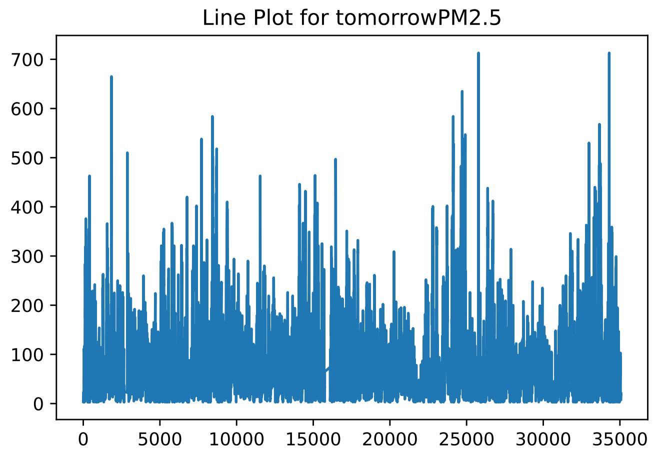
(g) PRES.



(h) RAIN.

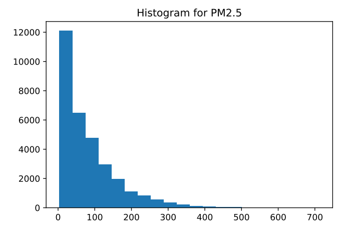


(i) WSPM.

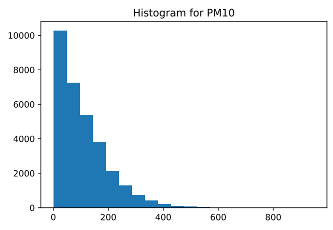


(j) Tomorrow$PM_{2.5}$.

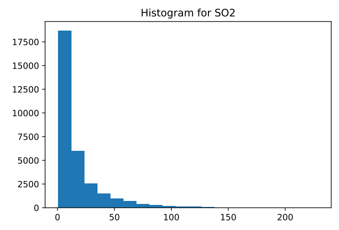
Figure 22: Line plots for each of the nine covariates and the response in our dataset.



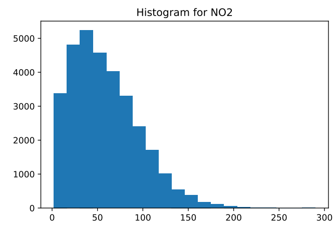
(a) $PM_{2.5}$.



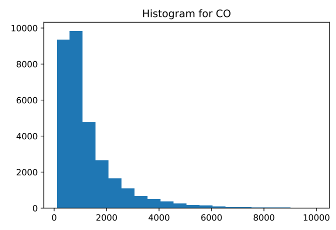
(b) PM_{10}.



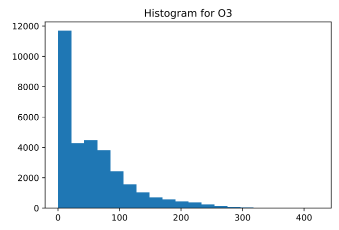
(c) SO_2.



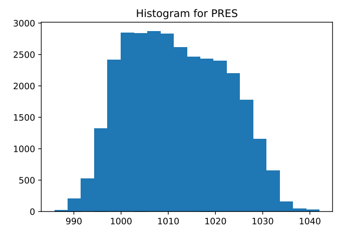
(d) NO_2.



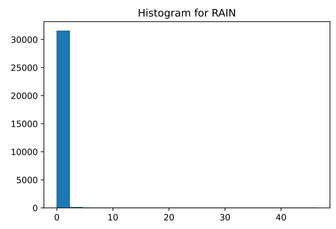
(e) CO.



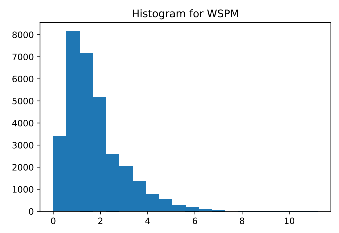
(f) O_3.



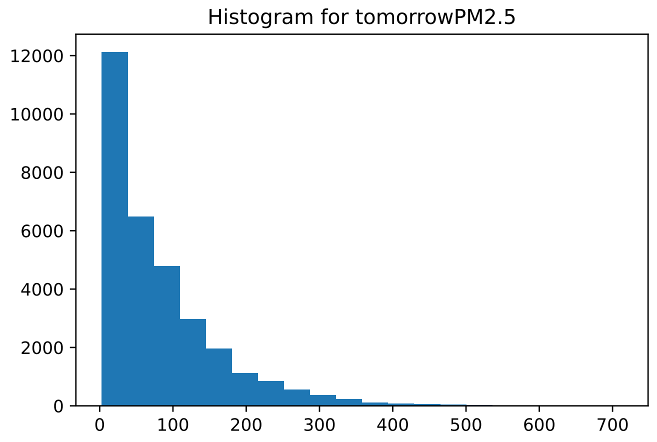
(g) PRES.



(h) RAIN.



(i) WSPM.



(j) $TomorrowPM_{2.5}$.

Figure 23: Histograms for each of the nine covariates and the response in our dataset.

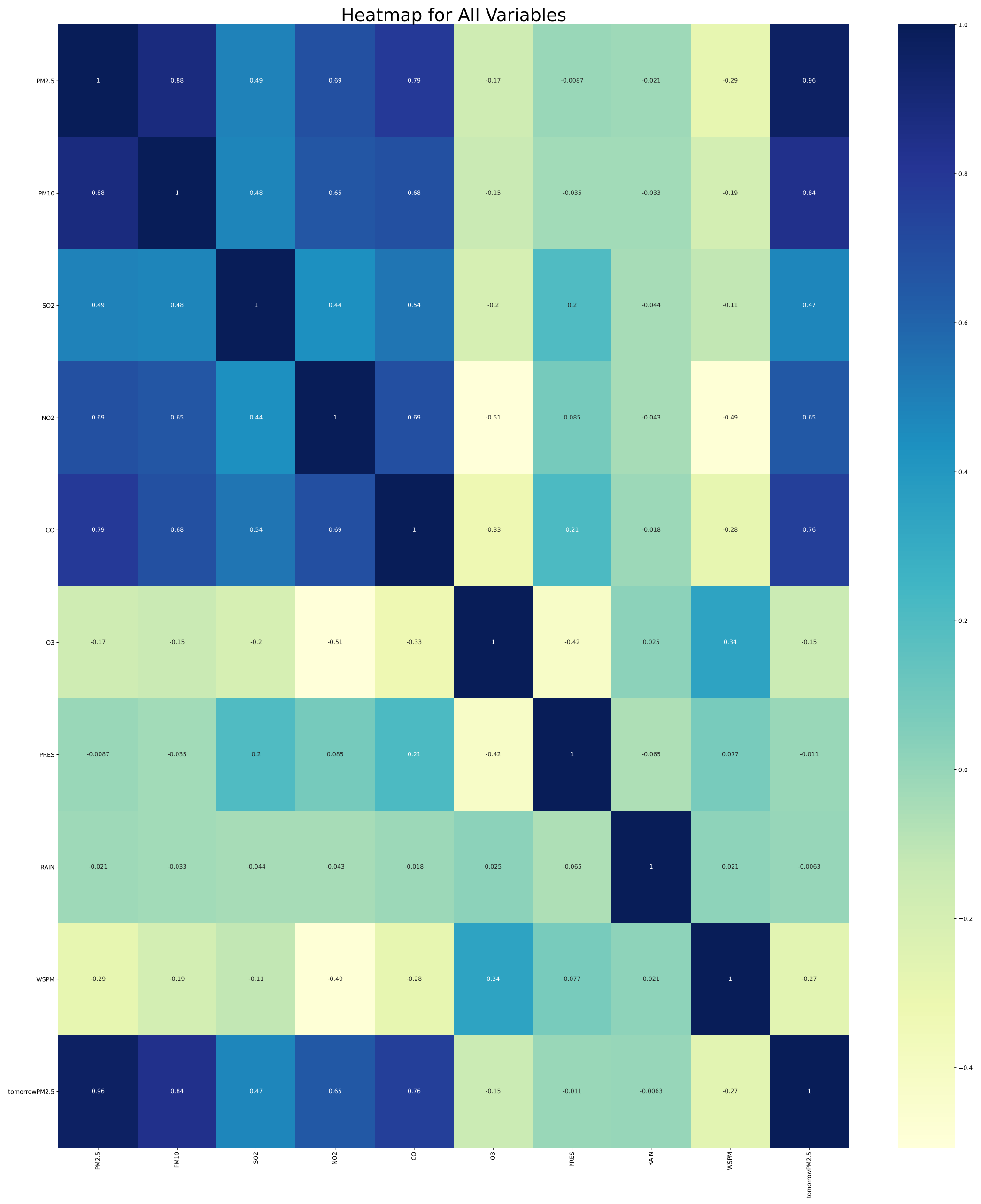
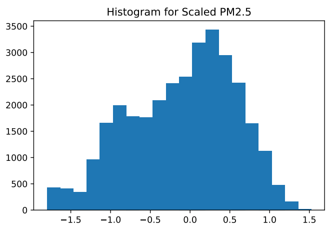
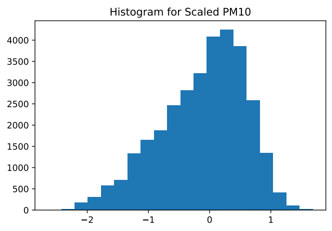


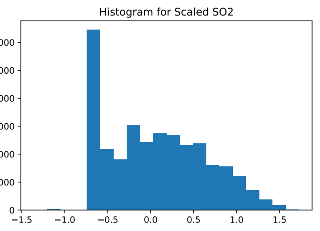
Figure 24: Heatmap showing pairwise correlation coefficients of covariates and response.



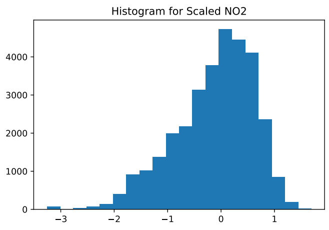
(a) $PM_{2.5}$.



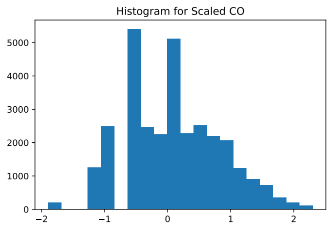
(b) PM_{10}.



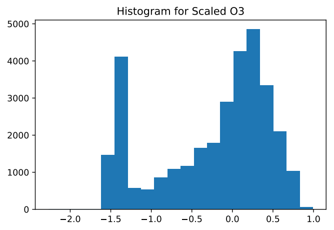
(c) SO_2.



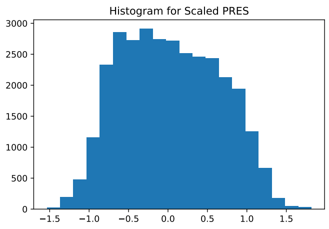
(d) NO_2.



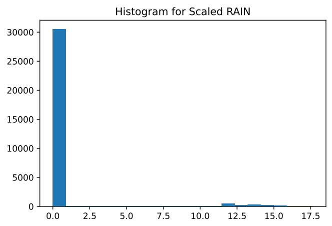
(e) CO.



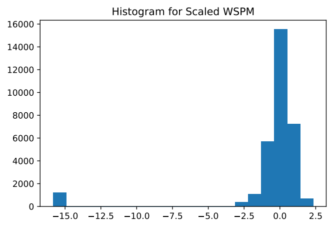
(f) O_3.



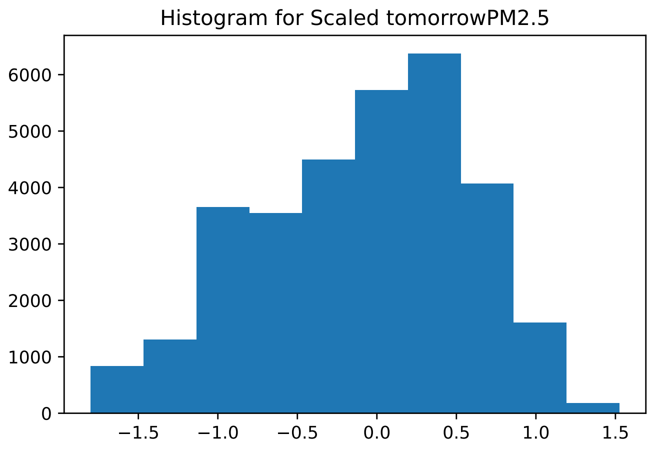
(g) PRES.



(h) RAIN.



(i) WSPM.



(j) Tomorrow$PM_{2.5}$.

Figure 25: Scaled histograms for each of the covariates and the response in our dataset.

lognormalized, we scale using the following formula (where $Q_\alpha(y)$ is our notation used to denote the αth quantile of the data y) that is displayed (for $i = 1, \dots, n$):

$$y_i \mapsto \frac{y_i - Q_{0.50}(y)}{Q_{0.75}(y) - Q_{0.25}(y)}. \tag{172}$$

We also perform a train-test split on the dataset, assigning 30% of the dataset as a test set, resulting in a training set with dimension 22270×10 and a test set with dimension 9545×10. Fitting our models on the train set yields the scores below:

Model	MSE	RMSE	MAE	R^2
LinearModel	0.041791	0.204429	0.124798	0.906273
FrequentistAM	0.039135	0.197827	0.120681	0.912229
SquaredExpGP	0.039246	0.198106	0.121153	0.911982
RationalQuadGP	0.039189	0.197963	0.121193	0.912108
OrnsteinUhlGP	0.039210	0.198014	0.121232	0.912063
SmoothDP	0.038189	0.195420	0.121443	0.914352
PeriodicDP	0.038189	0.195420	0.121439	0.914352
MixtureDP	0.038190	0.195422	0.121457	0.914350
RandomForest	0.031482	0.177432	0.111289	0.929393

On the other hand, fitting our models on the test set yields the following scores:

Model	MSE	RMSE	MAE	R^2
LinearModel	0.043389	0.208299	0.125495	0.903937
FrequentistAM	0.040531	0.201322	0.121427	0.910265
SquaredExpGP	0.040964	0.202397	0.122669	0.909305
RationalQuadGP	0.040988	0.202456	0.122855	0.909252
OrnsteinUhlGP	0.040953	0.202369	0.122749	0.909330
SmoothDP	0.041526	0.203780	0.125011	0.908060
PeriodicDP	0.041588	0.203932	0.125091	0.907924
MixtureDP	0.041505	0.203728	0.124995	0.908108
RandomForest	0.040227	0.200567	0.121238	0.910937

We have the following comments and analysis below for these numerical results:

(a) There is nothing much to say about linear regression besides the fact that LinearModel is the worst-performing model in both the train and test sets.

(b) We optimized RandomForest using Bayesian optimization, and this resulted in this model overfitting in training, having poorer performance in testing.

(c) The three Gaussian process prior models seemed to perform almost equally, with RationalQuadGP best in training but OrnsteinUhlGP best in testing.

(d) For MixtureDP, we handpicked some covariates to be assigned a smooth difference prior and others to be assigned a periodic difference prior. Here, our reasoning for these assignments are from the exploratory data analysis.

(e) Overall, difference prior models perform better than Gaussian process prior models in training, but worse in testing. Hence, this indicates that the local approach may be more prone to overfitting compared to a global approach.

To wrap things up, we see that all our models perform decently, achieving roughly a 0.90 R^2 score, with only minor variations. This could be due to the fact that the dataset in question here is rather large. Moreover, LinearModel does not lag too far behind the other models, which suggests that the data could have been more linear than initially expected. Among the Gaussian process prior models, again OrnsteinUhlGP emerges the winner as in Subsection 5.3, which suggests that the data may not be entirely smooth. This concludes our analysis for this experiment.

5.5 Conclusions and Final Remarks

Throughout this analysis, we see clear evidence for the initial three advantages of AMs posed in Section 1 brought to the fore. With AMs, we can select, a priori, the function type for each covariate based on exploratory data analysis, resulting in more model flexibility. Interpretability and smoothness control also appeared in full display throughout our experiments. The Bayesian methods which we have studied so much in Sections 3 and 4 also perform comparably (and even better in most cases) to their frequentist counterparts, and so deserve strong consideration.

It seems as though, in recent practice, there may be many instances where a particular practitioner of the art of modeling would sacrifice the interpretability of AMs in favor of very nonlinear but highly numerically optimized deep learning methods, when only prediction is concerned. However, this may not be the case forever; in addition to much of the research that is actively going in the study of additive models (and generalized additive models), there exists recent work for the purposes of making AMs parallelizable for modern hardware and reasonably efficient for very large datasets (see [41], for instance). This may further contribute to the growing popularity of AMs (and Bayesian AMs) in the near or far future.

At this end, what to do if, after perusing this project, one wants to study AMs and GAMs even further? For the frequentist method, the textbook by Wood [40] provides much more content that is not covered in this project. Not enough has been said here about the various different splines that one could employ as basis functions for AMs or GAMs. In particular, splines for the estimation of functions of many variables deserves more attention, and on this topic, of particular interest are the so-called "isotropic smoothers," which produce identical predictions of the response under any rotation or reflection of the covariates. Here, many spline archetypes with a rich depth of theory show up, including thin plate splines, the Duchon spline, the interestingly-named soap film smoother, and much more [40].

On the Bayesian side of things, not all has been addressed about our Gaussian process prior models and difference prior models either. Firstly, the linear Normal system of (113) and (114) can be modified (by imposing an inequality constraint) to fit functions which we know beforehand are either monotonically increasing or monotonically decreasing (along some input dimension). Then, this constrained problem can be solved using efficient quadratic programming methods like those in [14]. There is also the problem of hyperparameter estimation when employing the approaches in Section 3 or Section 4. So far, we have manually chosen values for the necessary hyperparameters (like difference prior variances and scale parameters for the kernel), but it is of course imperative to have a way of getting these hyperparameter values automatically. Solonen and Staboulis [37] suggest:

- direct MAP estimation;
- maximizing evidence;
- cross validation;
- posterior predictive score;
- variational inference;

as a few approaches towards hyperparameter estimation for AMs (with details).

Finally, in our experimentations, more could be done as well. In particular, it may be of interest to test other benchmark models, such as the classical Gaussian process regression or a deep learning method, to the three datasets that we have created/procured. A possible future work would be to perform a far more robust meta-analysis comparing the various AM/GAM approaches with each other and other benchmark models, with the aim of evaluating the overall performance of these models with a view towards providing empirical justification for their efficacy that goes far beyond testing on a few specifically chosen datasets alone.

References

[1] J. H. Bardsley. Gaussian markov random field priors for inverse problems, 2013.

[2] J. Blitzstein and C. Morris. Probability for statistical science, 2020.

[3] L. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation: Rejoinder, 1985.

[4] Brilliant.org. Lagrange interpolation, 2023.

[5] H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees, 2010.

[6] P. Craven and G. Wahba. Smoothing noisy data with spline functions, 1979.

[7] C. de Boor. A practical guide to splines, 1978.

[8] D. Dua and C. Graff. UCI Machine learning repository, 2017.

[9] D. Duvenaud. Automatic model construction with gaussian processes, 2014.

[10] D. Duvenaud, H. Nickisch, and C. Rasmussen. Additive gaussian processes, 2011.

[11] P. Eilers and B. Marx. Flexible smoothing with B-splines and penalties, 1996.

[12] L. Fahrmeir and G. Tutz. Multivariate statistical modelling based on generalized linear models, 1994.

[13] R. Garnett. Bayesian optimization, 2023.

[14] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs, 1983.

[15] P. J. Green and B. W. Silverman. Nonparametric regression and generalized linear models, 1994.

[16] O. Gressani and P. Lambert. Laplace approximations for fast bayesian inference in generalized additive models based on P-splines, 2021.

[17] H. Haario, M. Laine, M. Lehtinen, E. Saksman, and J. Tamminen. Markov chain monte carlo methods for high dimensional inversion in remote sensing, 2004.

[18] T. Hastie and R. Tibshirani. Generalized additive models, 1986.

[19] T. Hastie and R. Tibshirani. Generalized additive models: Some applications, 1987.

[20] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning, 2009.

[21] R. Horn and C. Johnson. Topics in matrix analysis, 1991.

[22] Y. J. Kim and C. Gu. Smoothing spline Gaussian regression: more scalable computation via efficient approximation, 2004.

[23] G. Kimeldorf and G. Wahba. A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines, 1970.

[24] P. Lancaster and K. Salkauskas. Curve and surface fitting: An introduction, 1986.

[25] S. Lang and A. Brezger. Bayesian P-splines, 2004.

[26] K. Larsen. GAM: the predictive modeling silver bullet, 2015.

[27] X. Lu, A. Boukouvalas, and J. Hensman. Additive gaussian processes revisited, 2022.

[28] Y. Marzouk and H. Najm. Dimension reduction and polynomial chaos acceleration of Bayesian inference in inverse problems, 2009.

[29] P. McCullagh and J. Nelder. Generalized linear models, 1989.

[30] C. Micchelli, Y. Xu, and H. Zhang. Universal kernels, 2006.

[31] C. E. Rasmussen and C. Williams. Gaussian processes for machine learning, 2006.

[32] K. Ravindra, P. Rattan, S. Mor, and A. Aggarwal. Generalized additive models: building evidence of air pollution, climate change, and human health, 2019.

[33] H. Rue and L. Held. Gaussian markov random fields: Theory and applications, 2005.

[34] H. Rue, S. Martino, and N. Chopin. Bayesian inference for latent gaussian models by using integrated nested laplace approximations, 2009.

[35] T. Sauer. Numerical analysis, 2006.

[36] B. Silverman. Some aspects of the spline smoothing approach to nonparametric regression curve fitting, 1985.

[37] A. Solonen and S. Staboulis. On bayesian generalized additive models, 2023.

[38] P. Taylan, G. Weber, and A. Beck. New approaches to regression by generalized additive models and continuous optimization for modern applications in finance, science, and technology, 2007.

[39] T. Webster, V. Vieira, J. Weinberg, and A. Ann. Methods for mapping population based case control studies: an application using generalized additive models, 2006.

[40] S. Wood. Generalized additive models: an introduction with R, 2006.

[41] S. Wood, Y. Goude, and S. Shaw. Generalized additive models for large data sets, 2015.

[42] X. Yan and X. Su. Linear regression analysis: theory and computing, 2009.

[43] T. Yee and N. Mitchell. Generalized additive models in plant ecology, 1991.

[44] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. Chen. Cautionary tales on air-quality improvement in Beijing, 2017.