Part II

Incorporating structure in Gaussian Process Regression

Chapter 4

Basic Covariance Functions

Résumé

La partie II de la thèse montre comment incorporer les informations *a-priori* dans la construction de modèles GP, en choisissant les différents types de fonctions de covariance. Bien que les formes fonctionnelles de covariance discutées ici soient généralement utilisées dans la communauté d'apprentissage automatique, elles ne sont pas souvent utilisées pour construire des modèles d'ingénierie. La contribution originale du chapitre 4 et du chapitre 5 est l'application de ces noyaux pour construire des modèles d'ingénierie (en mécanique des fluides et en structure). Cette partie est fortement inspirée des travaux de [Duvenaud 2014, Wilson 2014, Lloyd 2014, Durrande 2001].

La section 4.2 définit quelques propriétés importantes des fonctions de covariance. La section 4.3 détaille quelques propriétés de noyaux non-stationnaires et discute dans quelle mesure la Régression Linéaire Bayésienne peut être efficacement vue comme une régression GP avec la fonction de covariance linéaire. La section 4.4 décrit les noyaux stationnaires, utilisant le *théorème de Bochner* nous pouvons représenter un noyau stationnaire par sa transformée de Fourier, qui augmente l'interprétation des fonctions constitutive de la famille. Pour chaque fonction de covariance, nous essayons de donner une visualisation de la forme des fonctions constitutives.

La contribution principale de ce chapitre consiste à démontrer comment utiliser la régression GP pour détecter automatiquement les paramètres modaux de la dynamique structurelle. À notre connaissance, une telle méthode n'a pas été utilisée dans la littérature existante pour identifier les paramètres modaux. Par l'utilisation des noyaux de 'Spectral Mixture' nous démontrons comment construire des modèles pour des expériences dynamiques structurelles et identifier automatiquement des paramètres dynamiques comme la fréquence modale (section 4.5) [Chiplunkar 2017b]. Il s'agit

d'une étape très préliminaire de l'application de noyaux de 'Spectral Mixture' pour l'identification de systèmes, et des problèmes comme l'identification de la forme du mode ou le taux d'amortissement demeurent dans cet algorithme. Nous souhaitons approfondir cette application dans le futur.

4.1 Introduction

If we assume the mean function as equal to zero, then a GP prior can be completely parametrized by its covariance function. Hence the problem of learning in a GP regression is exactly the problem of finding suitable properties of the covariance function [Rasmussen 2005]. The covariance function consists of two parts: a functional form, (which specifies the shape of functions in the hypothesis space) and a set of hyper-parameters (which define the probability of a function in the hypothesis space). In section 2.4 we have seen how to automatically choose hyper-parameters, part II of this thesis will detail how to choose the functional form.

Part II of the thesis shows how to incorporate prior information of patterns into building GP models, by choosing different types of covariance functions. Chapter 4 shows a few basic kernel types, while chapter 5 shows how to combine these basic kernels together and incorporate more complex patterns. This part is heavily inspired from prior works of [Duvenaud 2014, Wilson 2014, Lloyd 2014, Durrande 2001].

Contribution

Although the functional forms of covariance discussed here are commonly used in the machine learning community, unfortunately they are not often used to build engineering models. The original contribution of chapter 4 and chapter 5 is application of these kernels to build meaningful engineering models[1]. In chapter 4 we build a GP model to automatically identify structural dynamics parameters (section 4.5) [Chiplunkar 2017b], while in chapter 5 we use a combination of basic kernels to identify the onset of non-linear behaviour in physical systems. For example we identify the initiation of flow separation in NACA 0012 airfoil and initiation of plasticity in AL6061 alloy using a statistical criteria for automatic detection of non-linearity (section 5.2.4) [Chiplunkar 2016b]. We finally build a GP model to predict the position of aerodynamic shock in the transonic regime (section 5.3.5) [Chiplunkar 2017a].

The current chapter unfolds as follows; section 4.2 details a few important properties of covariance functions. Section 4.3 gives some insights into non-stationary covariance functions. Section 4.4 describes stationary kernels and their Fourier domain. We then leverage this knowledge to automatically identify the dynamic behaviour of structural experiments (section 4.5).

[1] both in structural and fluid mechanics

4.2 Properties

A kernel is a function that maps any pair of inputs ($\boldsymbol{x_1} \in \mathcal{X}$ and $\boldsymbol{x_2} \in \mathcal{X}$) into a scalar $\mathbb{R}$, the inputs can be scalars, vectors, categorical variables [Villegas García 2013] or even images. The covariance function of a GP is a special type of kernel, which specifies covariance of a pair of random functions $f(\boldsymbol{x_1})$ and $f(\boldsymbol{x_2})$ situated at points $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (mostly written as a function of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$).

Measure of distance

Most of the learning algorithms work on distance measures, i.e. if two points are closer then their observations (y) will also tend to be similar. Covariance functions specify this measure of distance in a GP Regression. If two points have a high value of covariance, then they are assumed to be close, and hence will have similar value of outputs (y). Therefore, by defining a covariance function we encode which type of input points will be termed as close, this effectively encodes biases into our family of functions. Biases based on smoothness (section 4.4.1), linearity (section 4.3.1), differentiability (section 4.4.2), etc. can be easily encoded using simple covariance functions.

A covariance function between $f(\boldsymbol{x_1})$ and $f(\boldsymbol{x_2})$ can be written as equation 4.1.

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = cov(f(\boldsymbol{x_1}), f(\boldsymbol{x_2})) \tag{4.1}$$

A covariance function $k(\boldsymbol{x_1}, \boldsymbol{x_2})$ is always symmetric, since:

$$\begin{aligned} k(\boldsymbol{x_1}, \boldsymbol{x_2}) &= cov(f(\boldsymbol{x_1}), f(\boldsymbol{x_2})) && (4.2)\\ &= \mathbf{E}[f(\boldsymbol{x_1}) - m(\boldsymbol{x_1}), f(\boldsymbol{x_2}) - m(\boldsymbol{x_2})] && (4.3)\\ &= cov(f(\boldsymbol{x_2}), f(\boldsymbol{x_1})) && (4.4)\\ &= k(\boldsymbol{x_2}, \boldsymbol{x_1}) && (4.5) \end{aligned}$$

$k(\boldsymbol{x_1}, \boldsymbol{x_2})$ corresponds to a covariance function if it is a symmetric Positive Semi Definite (PSD) function [Mercer 1909, Loeve 1978, Durrande 2001]. Consider the variance of a new random vector $T = \sum \alpha_i f(\boldsymbol{x_i})$:

$$\begin{aligned} var(T) &= cov\left(\sum_i \alpha_i f(\boldsymbol{x_i}), \sum_j \alpha_j f(\boldsymbol{x_j})\right) = \sum_i \sum_j \alpha_i \alpha_j cov(f(\boldsymbol{x_i}), f(\boldsymbol{x_j}))\\ &= \sum_i \sum_j \alpha_i \alpha_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \end{aligned} \tag{4.6}$$

Since a variance is always non-negative, hence:

$$\sum_i \sum_j \alpha_i \alpha_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0 \tag{4.7}$$

According to the Mercer's theorem [Mercer 1909] equation 4.7 is a sufficient condition to prove that $k(\boldsymbol{x_i}, \boldsymbol{x_j})$ is a PSD function. The positive definite requirement means that the covariance kernel corresponds to an inner product in some basis space [Bishop 2006]. It is generally difficult to prove if a function is PSD, hence creating new covariance functions is a tough task. Fortunately there already exist a wide variety of covariance functions, this chapter will describe a few basic covariances.

4.3 Non-stationary kernels

Earlier in section 2.2 we have seen the SE kernel which is an example of stationary kernel. Stationary kernels are covariance functions which are purely a function of $\boldsymbol{d} = \boldsymbol{x_i} - \boldsymbol{x_j}$. In this section we list some non-stationary kernels and their properties.

4.3.1 Linear Kernel

The Bayesian linear regression described during section 1.4 can also be seen as a form of GP Regression but with a Linear covariance function. In the Bayesian linear regression we first assume a functional form of the function, in terms of its basis functions ($f(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \boldsymbol{w}$). We then assume a prior distribution of parameters, this is equivalent to assuming a prior distribution over functions. For a function and its prior as defined by equation 4.8.

$$f(\boldsymbol{x_i}) = \phi(\boldsymbol{x_i})^T \boldsymbol{w} \qquad \Pr[\boldsymbol{w}] = \mathcal{N}(0, \boldsymbol{\Sigma_{Prior}}) \tag{4.8}$$

The equivalent prior over functions[2] f can be written as equation 4.9.

$$\Pr[f(\boldsymbol{x})] = GP(0, \phi(\boldsymbol{x})^T \boldsymbol{\Sigma_{Prior}} \phi(\boldsymbol{x'})) \tag{4.9}$$

The above covariance function ($k(\boldsymbol{x_1}, \boldsymbol{x_2}) = \phi(\boldsymbol{x_1})^T \boldsymbol{\Sigma_{Prior}} \phi(\boldsymbol{x_2})^T$) describes a family of functions which are linear combinations of the basis functions ($\phi(\boldsymbol{x})$) [Bishop 2006]. The matrix $\boldsymbol{\Sigma_{Prior}}$, and noise variance σ_n^2, are the hyper-parameters of this GP prior while the $\phi(\boldsymbol{x})$ represents its functional form. Hence a linear basis ($\phi(\boldsymbol{x}) = \{1, \boldsymbol{x}\}^T$) describes a family of linear functions (equation 4.10), while a polynomial basis ($\phi(\boldsymbol{x}) = \{1, \boldsymbol{x}, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^P\}^T$) encodes a family of P^{th} order polynomial basis functions.

$$k_{Lin}(\boldsymbol{x_1}, \boldsymbol{x_2}) = w_0 + w_1 \boldsymbol{x_1}^T \boldsymbol{x_2} \tag{4.10}$$

[2] By the affine property (appendix A.3) of Multivariate Gaussian random variables we have that:

$$X = \mathcal{N}(\mu, \Sigma) \implies AX = \mathcal{N}(A\mu, A\Sigma A')$$

The above equation is the covariance function for a Linear kernel, where the w_0 is the hyper-parameter for the intercept while w_1 is the hyper-parameter for slope. If we assume an independent noise ϵ on the observations, then based on the discussion on noisy GPs (section 2.3.2) the GP prior becomes:

$$\Pr[y(\boldsymbol{x})] = GP(0, w_0 + w_1 \boldsymbol{x_1}^T \boldsymbol{x_2} + \sigma_n^2 \delta_{x_1 x_2}) \tag{4.11}$$

Hence, the intercept (w_0), the slope (w_1) and the noise (σ_n) are the hyper-parameters of the above prior. The above equation is equivalent to performing Bayesian linear regression as discussed in section 1.4. The hyper-parameters can be chosen using marginal likelihood and posterior prediction can be performed based on the discussion of section 2.4.

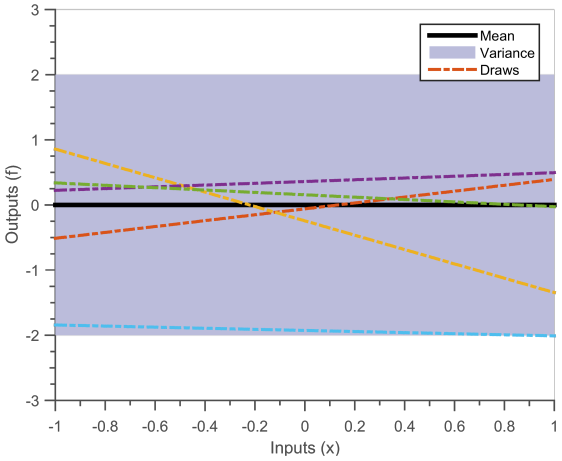
Revisiting Bayesian linear regression Let us revisit the experiment performed in section 1.4 but this time using GP regression and a linear kernel. The toy data-set $\mathcal{D}_1 = \{\boldsymbol{X} = [-0.5, 0.33, 0.66], \boldsymbol{y} = [0, 0.5, 0.5]\}$ will be used again. The prior distribution on parameters $\boldsymbol{\Sigma_{Prior}} = \left(\begin{smallmatrix} w_0 & 0 \\ 0 & w_1 \end{smallmatrix}\right)$ with $w_0 = 1, w_1 = 1$ and prior on noise $\sigma_n = 0.1$ will be the same as used in the earlier experiment.

Figure 4.1(a) shows draws from a GP prior with mean zero and Linear kernel as defined in the above paragraph. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior. Random functions drawn from a linear GP are linear. Figure 4.1(b) show draws from a GP posterior with mean zero and Linear kernel as defined in above paragraph and conditioned on the first data-point $X = -0.5, Y = 0$. The posterior mean passes from the data-point, random functions drawn from a linear GP are linear.

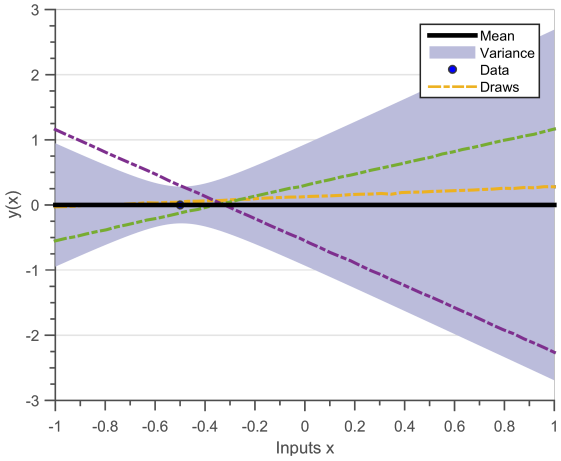
Figure 4.2(a) shows the contours of marginal likelihood with respect to intercept (w_0) and slope (w_1). The marginal likelihood is maximum for $w_0 = 0.2576, w_1 = 0.4584$, this is same as the posterior mean predicted in equation 1.8. This means that the data $\mathcal{D}_1$ has the highest possibility of coming from a data-set defined by a prior having these hyper-parameters. Figure 4.2(b) shows the posterior for same data-set but for the hyper-parameters where marginal likelihood is maximum.

4.3.2 Neural Network Kernel

It can be shown that a Neural Network with infinitely many hidden units and an error activation function ($erf(z)$) tends to a GP with a Neural Network kernel [Neal 2012] (equation

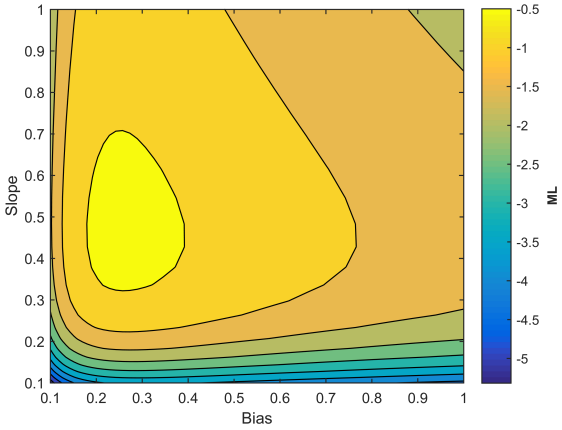


(a) Draws from a GP prior with mean zero and Linear kernel ($k_{Lin}(\boldsymbol{x_1}, \boldsymbol{x_2}) = \phi(\boldsymbol{x_1})^T \boldsymbol{\Sigma_{Prior}} \phi(\boldsymbol{x_2})^T + \sigma_n^2 \delta_{x_1 x_2}$) with $w_0 = 1, w_1 = 1$ and $\sigma_n = 0.1$. Random functions drawn from a linear GP are linear.

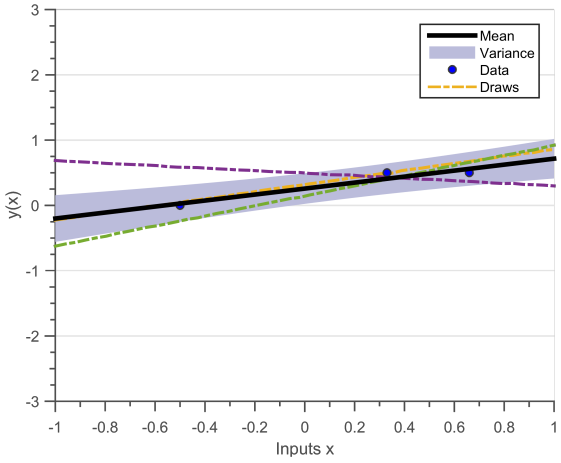


(b) Draws from a GP posterior with mean zero and Linear kernel (figure 4.1(a)) conditioned on the data $x = -0.5, y = 0$. The posterior mean passes from the data-point, random functions drawn from a linear GP are linear

Figure 4.1: Prior and posterior from a GP prior of linear kernel. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior.



(a) Marginal likelihood contours for varying bias and slope parameter. The noise hyper-parameter is ($\sigma_1 = [0.1]$). Marginal likelihood is maximum for $w_0 = 0.2576, w_1 = 0.4584$.



(b) Draws from a GP posterior, conditioned on the data-set $\mathcal{D}_1$ with mean zero and Linear kernel with hyper-parameters that maximize the marginal likelihood.

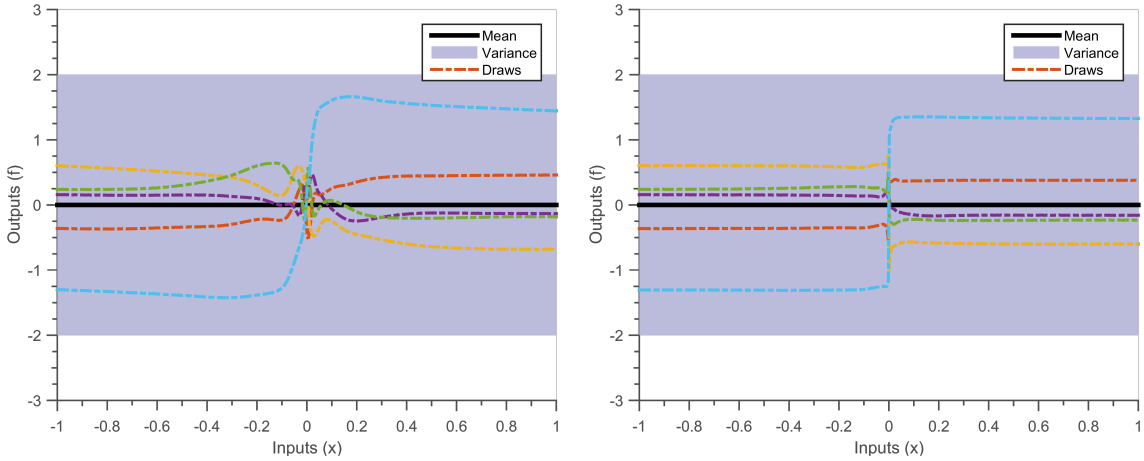
Figure 4.2: Maximizing Marginal Likelihood Linear kernel

4.12).

$$K_{NN}(x_1, x_2, \theta) = \theta_1^2 \frac{2}{\pi} sin^{-1}\left(\frac{2x_1\theta_2 x_2}{\sqrt{(1+2x_1^T\theta_2 x_1)(1+2x_2^T\theta_2 x_2)}}\right) \tag{4.12}$$

The hyper-parameters ($\theta = [\theta_1, \theta_2]$) are; amplitude θ_1 which defines average distance from mean and the length scale θ_2 which define the smoothness of functions. GPs with this covariance function define a space of superimposed sigmoidal functions. Figure 4.3 shows random draws from a Neural Network kernel but with varying value of hyper-parameters. Figure 4.3(b) has a higher value of smoothness hyper-parameter (θ_2) than figure 4.3(a), which makes the constituent functions have stronger slope. Hence, we can use this kernel to approximate the presence of discontinuity in our family of functions.

Hyper-parameters



(a) Draws from a GP prior with mean zero and Neural Network kernel (equation 4.12) with $\theta = [1, 10]$.

(b) Draws from a GP prior with mean zero and Neural Network kernel (equation 4.12) with $\theta = [1, 100]$. Higher value of θ_2 signifies increasing slope

Figure 4.3: Draws from Neural Network kernels having different hyper-parameters. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior.

4.3.3 Constant and Noise kernel

A constant covariance function (equation 4.13) defines a constant function. Here, $\sigma_{constant}$ defines the possible amplitude of the constant function.

$$k_{constant} = \sigma^2_{constant} \tag{4.13}$$

$$k_{noise}(x_1, x_2) = \sigma^2_{noise}\delta_{x_1 x_2} \tag{4.14}$$

On the other hand, equation 4.14 defines a white noise kernel, the σ_{noise} defines the amplitude of noise. $\delta_{x_1 x_2}$ is a Kronecker delta function which is 1 if $x_1 = x_2$ and zero otherwise, this means that observations at two inputs are independent of each other.

4.4 Stationary kernels

Covariance functions which are purely a function of distance $d = (x_1 - x_2)$ are called as stationary functions, Whereas covariance functions which are functions of absolute value of distance $d = |x_1 - x_2|$ are called as isotropic covariance functions. Stationary covariance functions remain unchanged if the points x_1, x_2 are translated. Hence a family of functions defined by stationary kernels will have similar local features throughout the input domain.

Bochner's theorem

The *Bochner's theorem* defines a relationship between a stationary covariance function and its Fourier transform. It states that the Fourier transform of a stationary covariance function exists and is a positive finite measure. If $k(d)$ is a stationary covariance function (equation 4.15) then $S(s)$ is its Fourier transform (equation 4.16), also called the power spectrum or spectral density [Bochner 1959, Stein 1999, Cox 1977]. A positive finite measure in this context means that $S(s)$ is non-negative for all frequencies s, and integral in equation 4.15 is finite.

$$k(d) = \int S(s) e^{2\pi i s^T d} ds \tag{4.15}$$

$$S(s) = \int k(d) e^{-2\pi i s^T s} dd \tag{4.16}$$

Power spectrum

The power spectrum is a more interpretable method of understanding constituent functions in a hypothesis space. A covariance function probabilistically defines a hypothesis space, the power spectrum corresponding to this covariance function tells us the power of the frequencies in this hypothesis space. If a power spectrum has more power at lower frequencies, then the constituent functions in its hypothesis space will be more smooth. Whereas, if the power spectrum has more power at higher frequencies, then the constituent functions in its hypothesis space will be less smooth [Wilson 2014].

4.4.1 Squared Exponential Kernel

We have already encountered the SE kernel in section 2.2.3. It is one of the most widely used kernel because it defines a hypothesis space of infinitely differentiable (infinitely smooth) functions. The SE kernel is Gaussian in shape (equation 4.17), which makes its Fourier transform also Gaussian in shape (equation 4.18).

$$k_{SE}(d,\theta) = \theta^2_{amplitude} exp\left[-\frac{d^2}{2\theta^2_{lengthScale}}\right] \tag{4.17}$$

$$S_{SE}(s,\theta) = \theta^2_{amplitude} exp\left[-2\pi\theta^2_{lengthScale}s^2\right] \tag{4.18}$$

Hyper-parameters

The two hyper-parameters of the SE kernel are the amplitude hyper-parameter ($\theta_{amplitude}$), which defines the amplitude of functions and the length-scale hyper-parameter ($\theta_{lengthScale}$), which defines the smoothness of functions.

Interpretation

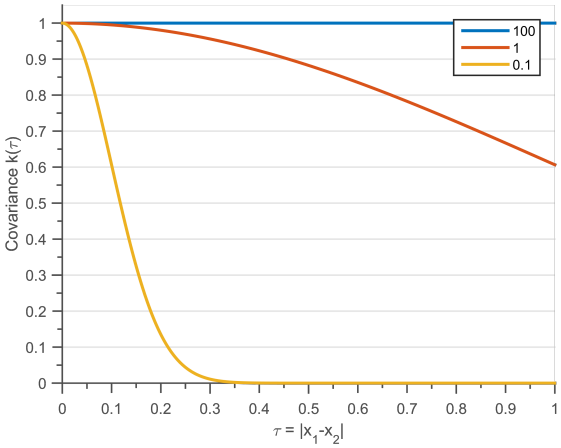
As discussed earlier, the covariance function (k_{SE}) defines the measure of similarity between two input points, if two points have high value of covariance then their output values (y) will be similar. If we increase $\theta_{lengthScale}$ then k_{SE} will also increase, for the same value of d and $\theta_{amplitude}$. This means that, the points for same value of d will become more similar and hence the constituent functions in the hypothesis space will become more smooth. As length-scale increases the constituent functions tend to become smoother. Another way to interpret the effect of length-scale is by observing the power spectrum (S_{SE}). If we increase $\theta_{lengthScale}$ then S_{SE} will start decreasing, for the same values of s and $\theta_{amplitude}$. This means that less power will be allocated to higher frequencies and hence the constituent functions in the hypothesis space will become more smooth (Figure 2.2).

Figure 4.4(a) plots the covariance values of SE kernel (($\theta_{amplitude} = 1$) for various values of length-scales ($\theta_{lengthScale} = [0.1, 1, 100]$)) whereas figure 4.4(b) plots the power spectrum for same kernels. We can see that the power spectrum ($S(s)$) for higher length-scales has less power at higher frequencies, meaning that their corresponding hypothesis spaces will not have faster moving functions.

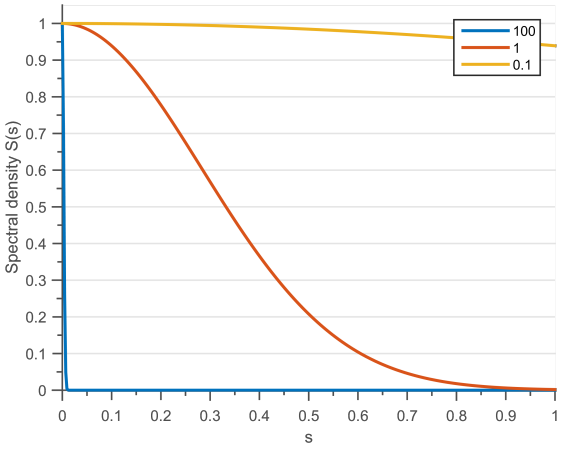
Note, as $\theta_{lengthScale}$ tends to infinity, an SE kernel tends to a constant covariance function (Fourier transform of a constant function is a delta function). Whereas, if $\theta_{lengthScale}$ tends to zero, an SE kernel tends to a white noise kernel (Fourier transform of a delta function is a constant function).

4.4.2 Matérn Kernel

The Matérn kernel is the second most popular kernel after the squared exponential. This kernel is derived using a t-distribution as the power spectrum ($S(s)$) and calculating its



(a) Kernel density for SE kernel with three different length-scales. The hyper-parameters are amplitude ($\theta_{amplitude} = 1$) and length-scale ($\theta_{lengthScale} = [0.1, 1, 100]$). $k(d)$ for $\theta_{lengthScale} = 100$ resembles a constant function.



(b) Power spectrum for SE kernel with three different length-scales. The hyper-parameters are amplitude ($\theta_{amplitude} = 1$) and length-scale ($\theta_{lengthScale} = [0.1, 1, 100]$). $S(s)$ for $\theta_{lengthScale} = 0.1$ resembles a constant function, while for $\theta_{lengthScale} = 100$ $S(s)$ resembles a dirac-delta function.

Figure 4.4: Covariance functions and Power spectrums for SE kernel with three different length-scales

inverse Fourier transform. A t-distribution has more weight on higher-frequencies (when compared to a Gaussian distribution) and hence gives rise to more non-smooth functions.

$$k_{Matern}(\nu, d, \theta) = \theta^2_{amplitude} \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu(d))}}{\theta_{lengthScale}} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu(d))}}{\theta_{lengthScale}} \right) \tag{4.19}$$

Differentiability

k_{Matern} is the covariance function for a Matérn kernel, ν is a positive parameter which signifies the degrees of freedom in the t-distribution of the power spectrum. $\Gamma(\nu)$ is a Gamma function, while K_ν is a modified Bessel function. The Matérn kernel provides the flexibility to define a hypothesis space of functions with varying degree of differentiability, due to this property they are often used to build machine learning models [Minasny 2005, Cornford 2002]. The degree of differentiability of the functions in the hypothesis space can be set as $[\nu - 1/2]$, i.e. $\nu = 1/2$ (equation 4.20) defines family of non-differentiable but continuous functions, $\nu = 3/2$ (equation 4.21) defines a family of functions differentiable only once and $\nu = 5/2$ (equation 4.22) defines a family of twice differentiable functions. Note, as ν tends to ∞ a t-distribution tends to a Gaussian distribution, similarly as ν

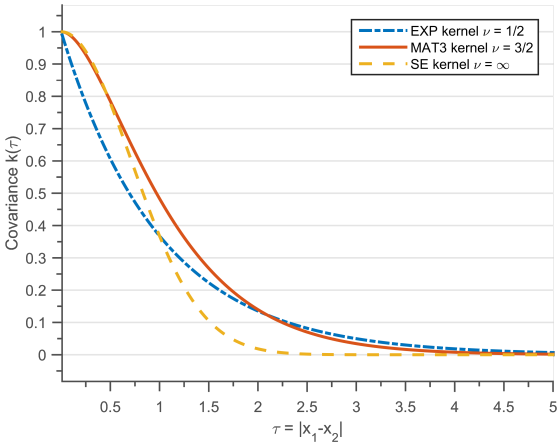
tends to ∞ the Matérn kernel tends to a SE kernel.

$$k_{Matern}(\nu = 1/2, d, \theta) = \theta^2_{amplitude} exp[-\frac{d}{\theta_{lengthScale}}] \tag{4.20}$$

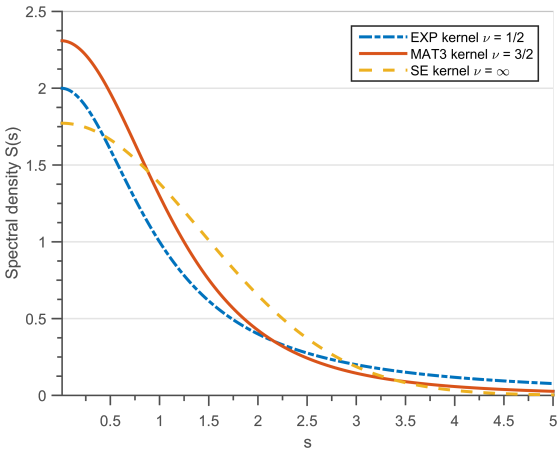
$$k_{Matern}(\nu = 3/2, d, \theta) = \theta^2_{amplitude}(1 + \frac{\sqrt{3}d}{\theta_{lengthScale}})exp[-\frac{\sqrt{3}d}{\theta_{lengthScale}}] \tag{4.21}$$

$$k_{Matern}(\nu = 5/2, d, \theta) = \theta^2_{amplitude}(1 + \frac{\sqrt{5}d}{\theta_{lengthScale}} + \frac{5d^2}{3\theta^2_{lengthScale}})exp[-\frac{\sqrt{5}d}{\theta_{lengthScale}}] \tag{4.22}$$

When $\nu = 1/2$ the Matérn kernel is also called the Ornstein-Uhlenbeck or Exponential kernel, this creates a hypothesis space of non-differentiable continuous functions and was used to explain the Brownian motion [Uhlenbeck 1930]. Figure 4.5(a) plots the covariance values of Exponential ($\nu = 1/2$), Matérn/ ($\nu = 3/2$) and SE kernel ($\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$) whereas figure 4.5(b) plots the power spectrum for the same kernels. We can see that the power spectrum ($S(s)$) of the SE kernel has lowest power for higher frequencies followed by Matérn ($\nu = 3/2$) and Exponential kernel, meaning that the SE kernel has more smooth functions in its hypothesis space followed by Matérn ($\nu = 3/2$) and Exponential kernel.



(a) Kernel density for exponential, Matérn ($\nu = 3/2$)and SE kernel. The hyper-parameters are amplitude ($\theta_{amplitude} = 1$) and length-scale ($\theta_{lengthScale} = 1$)



(b) Power spectrum for Exponential, Matérn ($\nu = 3/2$)and SE kernel. The hyper-parameters are amplitude ($\theta_{amplitude} = 1$) and length-scale ($\theta_{lengthScale} = 1$). Exponential and Matérn have more power at higher frequencies when compared to SE kernel.

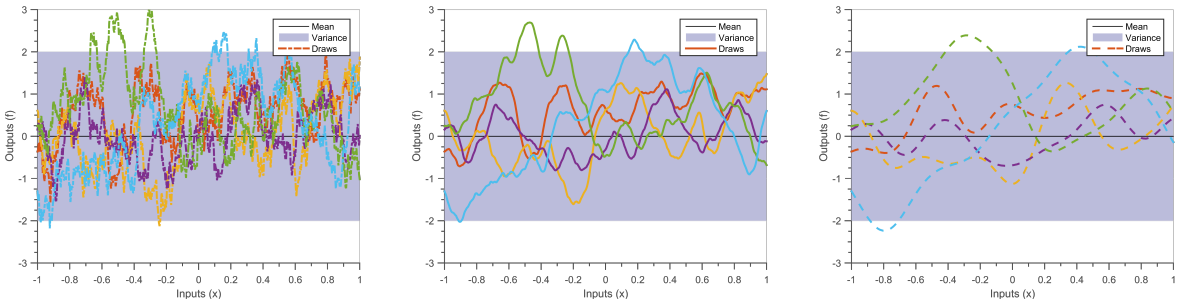
Figure 4.5: Covariance functions and Power spectrums for three different kernels

4.4.3 Experiments

Let us interpolate the data-set $\mathcal{D}_2$ used to choose the hyper-parameters in section 2.3, but this time using three different covariance functions. We will use Exponential kernel, Matérn ($\nu = 1/2$) and SE kernel and compare their interpolations.

We follow the standard framework of GP regression; we first draw random functions from the covariance function to judge the hypothesis space, we then calculate the posterior distribution conditioned on data-set $\mathcal{D}_2$, and finally optimize the marginal likelihood to compare the final predictions of the three covariance functions.

Figure 4.6 shows 5 random functions drawn for a zero mean GP with all the three covariance functions having the same values of hyper-parameters ($\theta_{lengthscale} = 1, \theta_{amplitude} = 1$), their corresponding covariance functions (figure 4.5(a)) and power spectrums (figure 4.5(b)) are shown in figure 4.5. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The colored (non-black) lines represent five functions drawn at random from a GP prior. We can observe that figure 4.6(a) draws non-differentiable functions while 4.6(b) and 4.6(c) draw smoother functions. For the same value of the length-scale, the Matérn ($\nu = 3/2$) kernel has faster variations in its functions, when compared to the SE kernel.



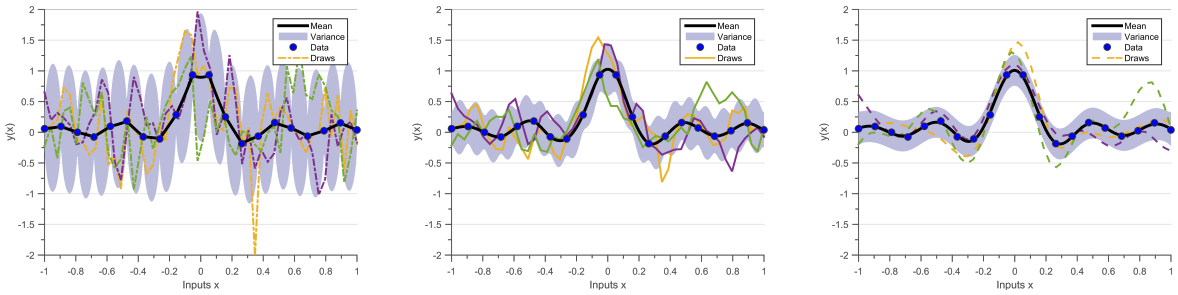
(a) Draws from a GP prior with mean zero, Exponential kernel (Matérn with $\nu = 1/2$) and hyper-parameters $\theta_{lengthscale} = 1, \theta_{amplitude} = 1$. Functions drawn from this kernel are non-differentiable.

(b) Draws from a GP prior with mean zero, Matérn kernel $\nu = 3/2$ and hyper-parameters $\theta_{lengthscale} = 1, \theta_{amplitude} = 1$. Functions drawn from this kernel are differentiable only once

(c) Draws from a GP prior with mean zero, SE kernel (Matérn with $\nu = \infty$) and hyper-parameters $\theta_{lengthscale} = 1, \theta_{amplitude} = 1$. Functions drawn from this kernel are infinitely differentiable

Figure 4.6: Prior distribution and five random draws from 3 different covariance functions. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior.

Figure 4.7 shows the posterior GP conditioned on the data-set $\mathcal{D}_2$ for three different covariance functions with the same hyper-parameters, here the marginal likelihood is not optimized. For similar values of hyper-parameters and data-set, Exponential kernel has highest variance followed by Matérn and SE kernel. This is a consequence of the bias vs variance trade-off since the SE kernel has the most restrictive hypothesis space (infinite differentiability is a stricter assumption), followed by Matérn $\nu = 3/2$ and Exponential kernel.



(a) Draws from a GP posterior with mean zero and Exopnential kernel (figure 4.6(a)) conditioned on the data $\mathcal{D}_2$. The posterior mean passes through the data-points, random functions drawn from exponential kernel are non-differentiable

(b) Draws from a GP posterior with mean zero and Matérn kernel $\nu = 3/2$ (figure 4.6(b)) conditioned on the data $\mathcal{D}_2$. The posterior mean passes through the data-points, random functions drawn from this kernel are differentiable only once

(c) Draws from a GP posterior with mean zero and SE kernel $\nu = \infty$ (figure 4.6(b)) conditioned on the data $\mathcal{D}_2$. The posterior mean passes through the data-points, random functions drawn from exponential kernel are infinitely differentiable

Figure 4.7: Posterior distribution and three random draws from 3 different covariance functions. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior.

Figure 4.8 shows the posterior GP conditioned on the data-set $\mathcal{D}_2$, for three different covariance functions with optimized hyper-parameters. All the three covariance functions estimate the same value of amplitude hyper-parameter, while having significantly different value of noise hyper-parameter (lowest noise estimated for Exponential kernel followed by Matérn ($\nu = 3/2$) and SE kernel).

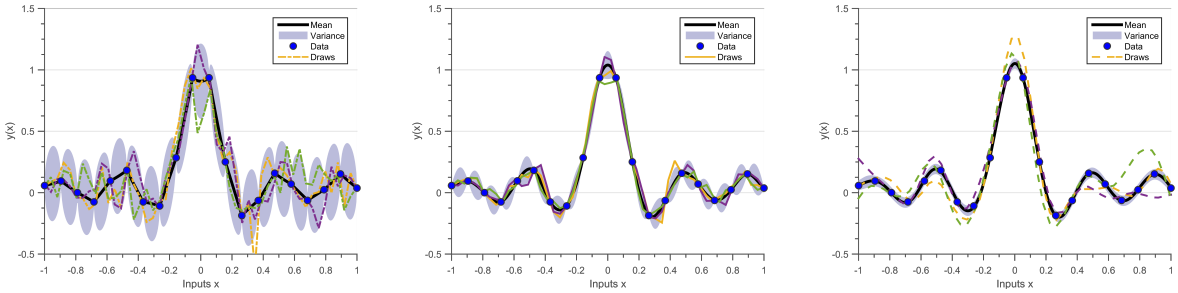
Bias vs Variance

The Exponential kernel defines a family of non-differentiable functions, hence functions in its hypothesis space have the flexibility to pass through all the data-points. Whereas, the SE kernel has less flexibility in the functions, due to its strict bias (infinitely differentiable). If the data-set does not have the same level of smoothness the SE kernel will find the closest function in its hypothesis space, and associate the difference to noise. Hence the Exponential kernel will almost always have a low noise

estimate than the SE kernel. The infinitely smooth assumption of the squared exponential function is unrealistic in several cases, making the Matérn kernels ($\nu = 5/2$) second most popular choice of kernels [Stein 1999, Cornford 2002].

Engineering judgment

This experiment only shows the different posteriors obtained for the same observational data and different functional forms of the covariance function. All three predictions can be the correct interpolations depending on the type of experiment, this is where engineering judgment is required. For example, if the data-set $\mathcal{D}_2$ was measured from a Brownian motion experiment then figure 4.8(a) would be the best fit, since the exponential kernel encodes prior information of Brownian motion (section 4.4.2).



(a) Draws from a GP posterior, conditioned on the data-set $\mathcal{D}_1$ with mean zero and Exponential kernel with hyper-parameters ($\theta_{lengthscale} = 0.215$, $\theta_{amplitude} = 0.312$ and $\sigma_n = 2.8e^{-}5$) that maximize the marginal likelihood ($max(ML) = -1$).

(b) Draws from a GP posterior, conditioned on the data-set $\mathcal{D}_1$ with mean zero and Matérn ($\nu = 3/2$) kernel with hyper-parameters ($\theta_{lengthscale} = 0.2$, $\theta_{amplitude} = 0.347$ and $\sigma_n = 1.7e^{-}5$) that maximize the marginal likelihood ($max(ML) = 2$).

(c) Draws from a GP posterior, conditioned on the data-set $\mathcal{D}_1$ with mean zero and se kernel with hyper-parameters ($\theta_{lengthscale} = 0.151$, $\theta_{amplitude} = 0.358$ and $\sigma_n = 0.01$) that maximize the marginal likelihood ($max(ML) = 8$).

Figure 4.8: Posterior distributions from three different covariance functions after maximizing the hyper-parameters.

Choosing functional form

If nothing is known about the data or type of experiment, and the number of hyper-parameters are same, then the covariance function with the maximum optimized marginal likelihood should be preferred. By this logic the SE kernel should be the preferred functional form of the covariance for the data-set $\mathcal{D}_2$. If the number of hyper-parameters are not the same, then marginal likelihood tends to be higher for covariance functions with greater number of hyper-parameters. [Duvenaud 2014, Lloyd 2014] propose to use the Bayesian Information Criterion (BIC)[3] to choose optimal covariance functions if the number of hyper-parameters is different.

[3]The BIC again performs a trade-off between data-fit and model complexity, for more details refer to section 4.5

4.4.4 Spectral Mixture Kernels

Spectral Mixture kernels define a more general class of stationary kernels exploiting the *Bochner's theorem* [Bochner 1959]. They define the power spectrum ($S(s)$) as a scale location mixture of Gaussians [Wilson 2013]. This has two benefits: firstly, with enough Gaussian components, scale location mixtures of Gaussians can approximate a curve up to arbitrary precision [Kostantinos 2000, Bishop 2006], secondly, the inverse Fourier transform of a scale location mixture of Gaussians can be evaluated analytically and is also a mixture of Gaussians.

$$S_{SM}(s,\mu,\sigma,w) = \sum_{q=1}^{Q} \frac{w_q}{\sqrt{2\pi\sigma_q^2}} \left(\exp\left[-\frac{(s-\mu_q)^2}{2\sigma_q^2} \right] + \exp\left[-\frac{(-s-\mu_q)^2}{2\sigma_q^2} \right] \right) \tag{4.23}$$

Here, S_{SM} is the power spectrum of the Spectral Mixture kernel. Each Gaussian component q has a mean μ_q, variance σ_q and weight w_q , there are total Q such components. The second term $\exp\left[-\frac{(-s-\mu_q)^2}{2\sigma_q^2}\right]$ is needed because a power spectrum of valid kernels should be symmetric around $s = 0$. The inverse Fourier transform of such a power spectrum will be a valid kernel and can be written as equation 4.24. For a detailed derivation refer to [Wilson 2014].

$$k_{SM}(d,\mu,\sigma,w) = \sum_{q=1}^{Q} w_q cos(2\pi\mu_q) exp[-2\pi^2 d^2 \sigma_q^2] \tag{4.24}$$

Here, k_{SM} is the covariance function of the above power spectrum (equation 4.23). The parameter w_q is the weight of the Gaussian component q, the mean of the Gaussian component μ_q defines the period of kernel while the variance σ_q of the Gaussian component denotes inverse of the length scale .

Pattern recognition

[Wilson 2014] demonstrate that the Spectral Mixture kernels can be used as a replacement for many available kernels, since they define a general class of stationary kernels. Due to their expansive hypothesis space, they can be used as a means to perform automatic pattern discovery. Their Fourier transform can provide more understanding of the observation data-set (this is very similar to interpreting the Fourier transform of dynamic data) [Wilson 2013]. One of the major hindrances of SE kernels is that they cannot be used for extrapolation, we cannot extrapolate an SE kernel a $\theta_{lengthScale}$ distance away from the last observation (figure 2.3). Since Spectral Mixture kernels can detect patterns in the data they can also be used to perform extrapolation[4].

[4]A detailed code can be found : https://people.orie.cornell.edu/andrew/code/#spectral

We propose to use this relationship between the covariance function and its Fourier transform to automatically identify parameters of a dynamic system. Dynamic engineering systems are generally parametrized by their modal frequencies and participation factors. In structural engineering, identification of modal frequencies is an important step for certification, while minor change in modal frequencies can help in fast discovery of failure. In the next section we apply the Spectral Mixture kernel to automatically identify the modal frequencies of a structural system, parts of the following work have been published in [Chiplunkar 2017b].

4.5 Application: Identifying Structural Dynamics Parameters

Modal analysis has been widely used as a means of identifying dynamic properties such as modal frequencies, damping ratios and mode shapes of a structural system. Traditionally, the system is subjected to artificial input excitations and the output deformations (displacements, velocities or accelerations) are measured. These later help in identifying the modal parameters of the system, this process is called Experimental Modal Analysis (EMA).

OMA

Since the last decade Operational Modal Analysis (OMA) has gained considerable interest in the structural dynamics community. OMA identifies the modal parameters only from the output measurements while assuming ambient excitations as random noise. OMA is cheaper because it does not require expensive experimental setup and can be used in real time operational use cases such as health monitoring [Peeters 2005, Shahdin 2010, Rainieri 2007].

MDOF

In the last few decades several algorithms primarily using the assumption of second order differential, Multi Degree Of Freedom (MDOF) system (equation 4.25) have been developed to find the modal parameters [Guillaume 2003, Richardson 1982].

$$\boldsymbol{M}\{\ddot{x}(t)\} + \boldsymbol{C}\{\dot{x}(t)\} + \boldsymbol{K_{Stiffness}}\{x(t)\} = \{\boldsymbol{f}(t)\} \tag{4.25}$$

Here, $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K_{Stiffness}}$ denote the mass, damping and stiffness matrices respectively. $\{x(t)\}$ and $\{\boldsymbol{f}(t)\}$ denote the displacement and force vectors at the time t, while $\ddot{x}(t)$ and $\dot{x}(t)$ denote the second and first time derivative of displacement respectively. Figure 4.9(a) shows an example of ambient measurements $x(t)$ on a structure. In almost all OMA algorithms the measurement $x(t)$ is assumed to be generated from a random force excitation.

Earlier Work The Natural Excitation Technique (NExT) [James III 1995] proves that the auto-correlation function $k(\tau)$ can be written as sum of decaying sinusoid's [Spitznogle 1970, Ibrahim 1977, Guillaume 2003]. The auto-correlation describes the similarity between measurements as a function of time lag τ between them (figure 4.9(b)). *NExT*

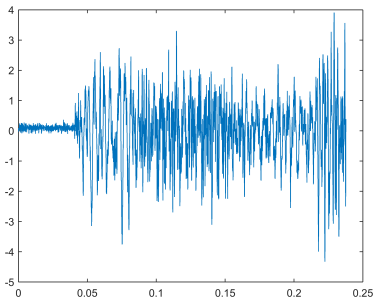
$$k(\tau) = \int x(t)x(t-\tau)dt \quad k(\tau) = \sum A_i exp(-\lambda_i \tau) sin(B_i \tau) \tag{4.26}$$

Here, $k(\tau)$ denotes the auto-correlation for random vector $x(t)$ as a function of time lag τ. While, λ_i and A_i denote the modal frequency and mode shapes for the i^{th} mode. The above coefficients are found by minimizing the least square error between the measured $k(\tau)$ and the predicted $k(\tau)$ from equation 4.26.

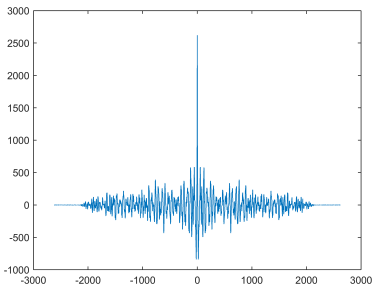
If we assume the measurement $x(t)$ to be a stationary random process, then according to *Bochner's theorem*, the Fourier transform of $k(\tau)$ (power spectrum $S(s)$) exists [Bochner 2016]. Figure 4.9(c) shows the power spectrum calculated for the measurement $x(t)$ shown in figure 4.9(a). Using the above mentioned second order differential assumption a Rational Fractional Polynomial (RFP) (equation 4.27) can be used to fit a power spectrum [Richardson 1982, Allemang 1998, Chauhan 2007]. *RFP*

$$S(s) = \int k(\tau) e^{-2\pi i s^T s} d\tau \quad S(s) = \frac{\sum a_k(s)^k}{\sum b_l(s)^l} \tag{4.27}$$

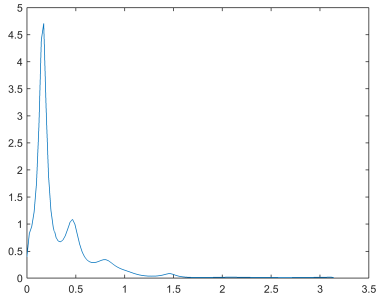
Here, the poles of the polynomial denote the modal frequencies, while other modal parameters can be derived from the coefficients a_k and b_l. The coefficients of the polynomial can be found by minimizing the least squared error. RFP based algorithms face problems of numerical stability as the value of number of modes (l) increases.



(a) Measured output on accelerometers $x(t)$



(b) Auto-correlation $k(\tau)$ of the measured output fig: 4.9(a)



(c) Power spectrum density $S(s)$ of the output fig: 4.9(a)

Figure 4.9: Different types of measurements for estimation of Modal parameters in OMA

Measurement	Auto-correlation	Power Spectrum
$x(t)$	$k(\tau) = \int x(t)x(t-\tau)dt$	$S(s) = \int k(\tau)exp(-2\pi i s^T \tau)d\tau$
Assumption: Second Order Differential		
	$\sum A_i exp(-\lambda_i \tau)sin(B_i \tau)$	$\frac{\sum a_k(s)^k}{\sum b_l(s)^l}$
Assumption: Gaussian Mixture Model		
$GP(0, k_{SM})$	$\sum w_i cos(2\pi\mu_i\tau)exp\{-2\pi^2\sigma_i^2\tau^2\}$	$\sum w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\{\frac{1}{2\sigma_i^2}(s-\mu_i)^2\}$

Table 4.1: Comparison of fitting functions

Contribution The above mentioned OMA algorithms NExT in the time lag (τ) domain and RFP in the frequency domain (s), assume the dynamic behaviour to be a second order differential system. This assumption fails for non-linear systems and for cases where modal frequencies are very close. Instead we propose to use the Spectral Mixture kernel to fit the measurement $x(t)$.

Using the same hypothesis used in section 4.4.4, we can say that a scale-location mixture of Gaussian components can approximate any distribution. We thus place a scale-location mixture of Gaussians on the power spectrum ($S(s)$), this means that we are assuming a prior distribution of $x(t)$ as a GP with a Spectral Mixture kernel (equation 4.28). The hyper-parameters of the system are: the mean μ_q defines the modal frequency, the variance σ_q which is a measure of damping, the weight w_q which defines the participation factor of component q and the total number of components Q.

$$\Pr[x(t)] = GP(0, k_{SM} = \sum_{q=1}^{Q} w_q cos(2\pi\mu_q)exp[-2\pi^2\tau^2\sigma_q^2) \tag{4.28}$$

Table 4.1

We would like to emphasize that keeping the computational complexities aside, fitting a Spectral Mixture GP in time-domain (equation 4.28), fitting Spectral Mixture (equation 4.24) on covariance values and fitting a scale-location mixture of Gaussians (equation 4.23) on the power spectrum are equivalent. In the publication [Chiplunkar 2017b] we fit a scale-location mixture of Gaussians on the power spectrum, here we propose to fit the GP on the measurements $x(t)$. Refer to table 4.1 for a more comprehensive view at various fitting functions.

Choosing Q

While the modal parameters can be chosen by minimizing the least square error, how to choose the number of modes is a recurring question in several OMA algorithms. This problem is partially resolved by using stabilization diagrams or mode identifica-

tion functions [Allemang 1998, Williams 1985, Shih 1988]. But in practical situations, engineering judgment is often required to estimate the optimal modal order. To automatically choose Q, we use the Bayesian Information Criteria (BIC) [Findley 1991] which penalizes more complex models to estimate the parameter Q. The BIC estimate has been earlier used to find the optimal functional form of covariance function [Duvenaud 2013]

$$BIC(Q) = N \ln(ML) + N_{hyp} \ln(N) \tag{4.29}$$

Here, N denotes the number of data-points to fit, ML denotes the marginal likelihood of the fit and N_{hyp} denote the number of hyper-parameters to fit. The BIC performs a trade-off between the data-fit term $N \ln(ML)$ and the complexity penalty term $N_{hyp} \ln(N)$, basically penalizing for over-fitting. Lowest value of BIC is preferred.

BIC

Generally, identification of modal frequencies requires engineering judgment, i.e. engineers have to look at the stabilization diagrams and figure out the optimal value of modal order. Using the Spectral Mixture kernel we have encoded the knowledge of peaks in the power spectrum which in combination with BIC has eliminated the need to perform costly engineering judgment exercises and has made the process automatic. We now compare the accuracy of finding modal frequencies using a Spectral Mixture kernel vs NExT [James III 1995] methodology for a toy data-set, and then later for a real data-set of HCT building [Brincker 2000].

4.5.1 Results on a toy data-set

In this section we conduct experiments, applying our approach on a highly damped 3 degree of freedom system. As stated earlier we fit a GP model having a Spectral Mixture covariance on the measurements $x(t)$ for varying number of Gaussian components Q. We then evaluate the BIC to find the optimal value of Q for this measurement. The results of automatic identification are then compared to that of NeXT identification method.

All experiments were performed on an Intel quad-core processor with 4Gb RAM. The Spectral Mixture technique suffers from multiple minimas and thus care should be taken while initializing the hyper-parameters[5].

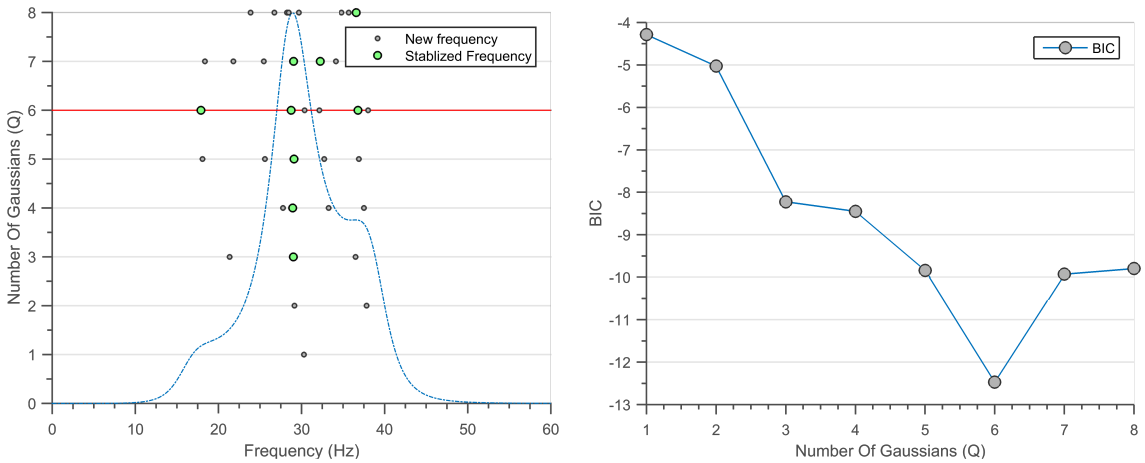
Table 4.2 shows the comparison of frequencies for Modal frequencies predicted using NeXT algorithm and Spectral Mixture algorithm. We can observe that the NeXT method cannot predict the third frequency with enough accuracy. This is because the third frequency has the hignest value of damping.

[5]For fast optimization we first fit a Gaussian Mixture model on the power spectrum to initialize the hyper-parameters and then optimize the marginal likelihood.

	Real Frequency	Error NeXT Frequency	Error Spectral Mixture
First Frequency (Hz)	17.5	2.7 %	0.28 %
Second Frequency (Hz)	30	3.4 %	4.1 %
Third Frequency (Hz)	35	12.34%	5.1%

Table 4.2: Comparison of Modal frequencies for toy data-set

Figure 4.10(a) shows the stabilization diagram with increasing number of Gaussians Q. As we progressively increase the number of components we start getting more and more modal frequencies. We call a frequency stabilized if the difference between modal frequencies of two subsequent Q is less than 1%. The green points are stabilized frequencies. Figure 4.10(b) shows the BIC criterion with increasing number of Gaussian's Q. We can see that that the BIC is minimum for $Q = 6$ and hence if we add any more Gaussians for our data-set we will be over-fitting. The red line in figure 4.10(a) shows the Q with minimum BIC and hence the stabilized frequencies for this automatically become our modal frequency predictions.



(a) Stabilization diagram with increasing number of Gaussians Q, the dots denote the stabilized frequencies. We can observe that as the number of Q increases the algorithm starts finding better and better modes.

(b) The BIC criterion with increasing number of Gaussian's Q. We can see that that the BIC is minimum for $Q = 6$ and hence if we add anymore Gaussian's for our data-set we will be performing over-fitting

Figure 4.10: Results of Spectral Mixture kernels on a toy data-set

4.5.2 Results on a HCT building data-set

We now apply our methodology on real ambient response of Heritage Court Tower (HCT) Building[6]. The responses are measured at 6 different points on the building resulting in 6 different power spectrums. We compare the modal frequencies obtained using Spectral Mixture kernel with the modal frequencies obtained in the original paper [Brincker 2000]. To compare the performance we automatically identify modal frequencies present in each of the 6 power spectrums and take the mean of the stabilized frequencies.

Table 4.2 shows the comparison of Modal frequencies predicted in [Brincker 2000] with the Spectral Mixture algorithm. The results of the two methods are very similar, although the frequencies obtained by Spectral Mixture GP are completely automatic.

	Spectral Mixture	[Brincker 2000]
First Frequency (Hz)	1.23	1.23
Second Frequency (Hz)	1.29	1.27
Third Frequency (Hz)	1.43	1.45
Fourth Frequency (Hz)	3.87	3.86
Fifth Frequency (Hz)	4.28	4.25

Table 4.3: Comparison of Modal frequencies for HTC data-set

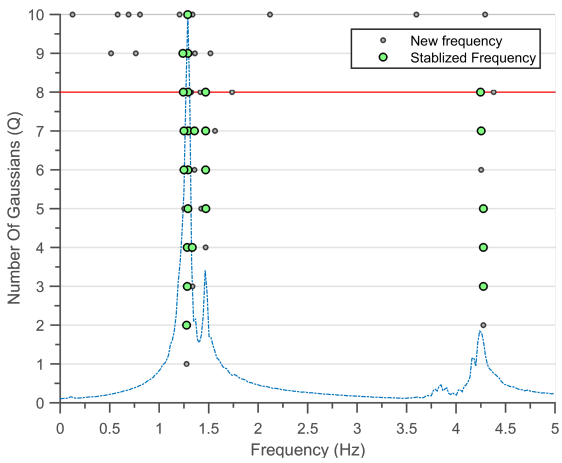
Figure 4.11(a) shows the stabilization diagram with increasing number of Gaussians Q. We can observe that as the number of Q increases the algorithm starts finding better and better modes, three modes start stabilizing from $Q = 5$. Figure 4.11(b) shows the BIC criterion with increasing number of Gaussian's Q. We can see that that the BIC is minimum for $Q = 8$ and hence if we add any more Gaussians for our data-set we will be over-fitting.

In the current setting of the Spectral Mixture model we propose an automatic way to identify the most important frequencies of a structural system. Neither the mode shapes nor the damping ratios are estimated in the current format. In the future we would like to derive a method to estimate mode-shape and damping ratio such that the contributions of neighbouring Gaussians are also taken into account.

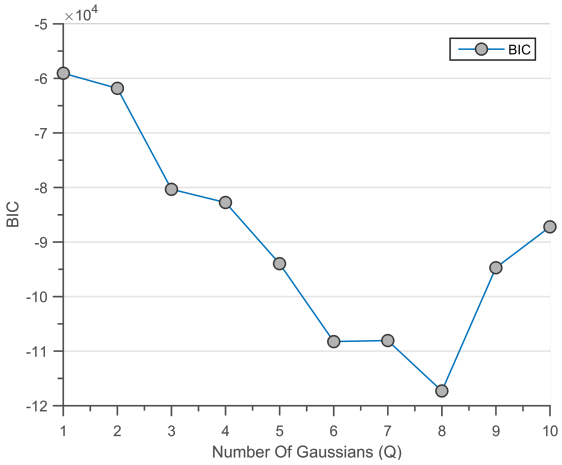
4.6 Summary and discussion

This chapter discussed only a small number of possible covariance functions, the table below provides a list of basic covariance functions that we have encountered thus far. Several other other functional forms of the covariance function exist in the literature, for example

[6]The data is available at : http://www.brinckerdynamics.com/oma-toolbox



(a) Stabilization diagram for one of the 6 power spectrums. The green dots denote the stabilized frequencies, the red line denotes minimum BIC. We can observe that as the number of Q increases the algorithm starts finding better and better modes.



(b) The BIC criterion with increasing number of Gaussian's Q. We can see that that the BIC is minimum for $Q = 8$ and hence if we add any-more Gaussian's for our data-set we will be per-forming over-fitting

Figure 4.11: Results of Spectral Mixture kernels on real data from HTC building

Gibbs kernel, Rational Quadratic kernel, Periodic kernel etc for more detailed list please refer to works of [Rasmussen 2005, Duvenaud 2013, Wilson 2014]. .

Kernel Name	Expression $k(x_i, x_j)$	Constituent functions
Constant	σ_c^2	Constant functions
Noise	$\sigma_n^2 \delta_{x_i x_j}$	White noise
Linear	$\phi(x_i)\Sigma\phi(x_j)$	Linear combination of ϕ
Neural Network	$\theta_1^2 \frac{2}{\pi} sin^{-1}\left(\frac{2x\theta_2 x'}{\sqrt{(1+2x^T\theta_2 x)(1+2x'^T\theta_2 x')}}\right)$	Linear combinations of Sigmoids
Standard Exponential	$\theta^2_{amplitude} exp[-\frac{\tau^2}{2\theta^2_{lengthScale}}]$	Infinitely Differentiable functions
Matérn	$\theta^2_{amplitude} \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu(\tau))}}{\theta_{lengthScale}}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu(\tau))}}{\theta_{lengthScale}}\right)$	Adjustable differentiability
Spectral Mixture	$\sum_{q=1}^{Q} w_q cos(2\pi\mu_q) exp[-2\pi^2\tau^2\sigma_q^2]$	Summation of periodic kernels

Table 4.4: List of covariance functions

We start off this chapter by defining a few important properties (section 4.2) of the covariance functions. Section 4.3 details a few non-stationary kernels and discusses how Bayesian linear regression can be effectively seen as a GP regression with linear covariance function. Section 4.4 describes stationary kernels, Using the *Bochner's theorem* we can effectively represent a stationary kernel into its Fourier transform, this gives rise to another interpretation of the constituent family of functions. Section 4.4.3 demonstrates the effects on posterior prediction for different functional forms of covariance functions. For each covariance function we try to give a visualization of the shape of constituent functions by randomly drawing functions from their priors.

Modal parameter estimation

The main contribution of this chapter is demonstration on how to use GP regression to automatically detect modal parameters of structural dynamics. To the best of our knowledge, such a method has not been used in the earlier literature to identify modal parameters. Using the Spectral Mixture kernels we demonstrate how to build models for structural dynamic experiments and automatically identify dynamic parameters such as modal frequency. This is a very early stage of application of Spectral Mixture Kernel for system identification, and there remain problems such as identification of mode-shape, and damping ratio in this algorithm. We wish to tackle these problems in the future.

As mentioned earlier the core aim of machine learning was to identify patterns and extrapolate on data automatically. There are two main approaches in pattern discovery for GPs; one is based on increasing the hypothesis space by defining a process over all stationary kernels [Wilson 2012]. The second which defines a language of kernels and iteratively adds basic kernels to come up with an explanation to the pattern in the data [Lloyd 2014]. In the next chapter we will look in detail at how to build more complex kernels using basic kernels. Which approach will be retained in the long term is difficult to predict but automatic pattern detection is a highly active research subject in GPs.

Chapter 5

Combining Basic Covariance Functions

Résumé

Souvent, le type d'information initiale que nous voulons intégrer n'est pas disponible sous la forme de fonctions de covariance basiques. Heureusement, plusieurs nouveaux noyaux peuvent être créés en fusionnant les noyaux de base que nous avons vus dans le chapitre précédent. Ce chapitre est inspiré par les travaux de [Bishop 2006, MacKay 2003, Durrande 2001, Durrande 2013a].

La section 5.2 décrit comment combiner les noyaux pour créer des noyaux unidimensionnels. Nous fournissons une explication intuitive de ce qui se produit lorsque nous ajoutons ou multiplions des noyaux, ces types de noyaux peuvent être utilisés pour détecter et séparer les modèles de données. La section 5.2.4 décrit comment créer des noyaux pour des dimensions plus élevées. Ce type de noyaux peut être utilisé pour effectuer une analyse de sensibilité ou projeter les données en dimension plus faibles.

Dans ce chapitre, nous utilisons une combinaison de noyaux de base pour identifier l'apparition de non linéarités dans les systèmes physiques. Par exemple, nous identifions l'initiation de la séparation de l'écoulement pour le profil d'aile NACA 0012 et l'initiation de la plasticité pour l'alliage AL6061 en utilisant un critère statistique pour la détection automatique de la non-linéarité (section 5.2.4) [Chiplunkar 2016b]. Nous construisons enfin un modèle de GP pour prédire la position du choc aérodynamique en régime transsonique (section 5.3.5) [Chiplunkar 2017a].

5.1 Introduction

What if the kind of patterns that we wish to encode are not possible by using the basic kernels described in chapter 4? What if we wish to encode several different patterns in our hypothesis space? Or what if we want to build models for data-sets with more than one input dimension? Thankfully, many new kernels can be constructed by merging a few basic kernels, this chapter answers the above questions.

The original contribution of this chapter is applying the newly created covariance functions to create engineering design models. The set of equations below are a few simple methods to create valid covariance functions [Bishop 2006, MacKay 2003, Durrande 2001, Durrande 2013a].

$$k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2) \tag{5.1}$$
$$k(x_1, x_2) = k_1(x_1, x_2) \times k_2(x_1, x_2) \tag{5.2}$$
$$k(x_1, x_2) = q(x_1)k_1(x_1, x_2)q(x_2) \tag{5.3}$$
$$k(x_1, x_2) = k_1(h(x_1), h(x_2)) \tag{5.4}$$
$$k(x_1, x_2) = g(g(k_1(x_1, x_2), x_1), x_2) \tag{5.5}$$
$$k(x_1, x_2) = \int k_1(x_1, u)k_2(u, x_2)du \tag{5.6}$$

Here, $k_1(x_1, x_2)$ and $k_2(x_1, x_2)$ are valid covariance function, while $q(x)$ and $h(x)$ are any continuous functions. Equation 5.6 defines a covariance function through convolution of two kernels, while equation 5.5 defines transformation through a linear operator $g(.) \in \mathcal{C}^2$, for more discussion on this equation refer to chapter 7.

Let us take the case of a 2-dimensional input vector[1] $\boldsymbol{x}$ such that $\boldsymbol{x} = \{x^1, x^2\}$ (x^1, x^2 are coordinates of $\boldsymbol{x}$ in the first and second dimension respectively). Then the following covariance functions are also valid.

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = k_1(x_1^1, x_2^1) + k_2(x_1^2, x_2^2) \tag{5.7}$$
$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = k_1(x_1^1, x_2^1) \times k_2(x_1^2, x_2^2) \tag{5.8}$$

The current chapter is written to provide intuition on what happens when we combine covariance functions. This chapter unfolds as follows; section 5.2 provides intuition on combining kernels for one-dimensional inputs, while section 5.3 details how to create covariance functions for higher-dimensions (equation 5.7 and 5.8) inputs. We then apply the newly constructed kernels, to first detect onset of non-linearity in experimental data, and later interpolate the shock pressures on a CRM wing in transonic regime.

[1] The superscript is used to denote the number of dimension and does not denote the power.

5.2 One dimensional inputs

Combining kernels can give rise to interesting features, in this section we provide intuition on combining kernels in one dimension. Section 5.2.1 details effects of multiplying kernels (equation 5.2) while section 5.2.2 describes effects of adding kernels (equation 5.1). Section 5.2.4 describes the Change-Point (CP) kernel. We use the CP kernel to detect start of plasticity in elastic structures such as beams and start of flow separation on an airfoil [Chiplunkar 2016b].

5.2.1 Multiplying Kernels

Multiplying two covariance functions acts as an AND operator, the resulting kernel has high value only if we have high value on both the kernels. Multiplying a Linear kernel L times, will result in a L^{th} order polynomial regression (equation 5.9).

$$k_{Lin} = w_0 + w_1 x_1^T x_2 \quad k_{Poly} = \prod_{i=1}^{L} \left(w_0^i + w_1^i x_1^T x_2\right) \tag{5.9}$$

Equation 5.10 shows the prior obtained after multiplying a Linear and a SE kernel. This covariance function resembles a SE covariance function but with the amplitude hyper-parameter ($\theta_{amplitude} = x_1 x_2$) proportional to the distance.

$$k_{Multi} = x_1 x_2 exp[-\frac{d^2}{2}] \tag{5.10}$$

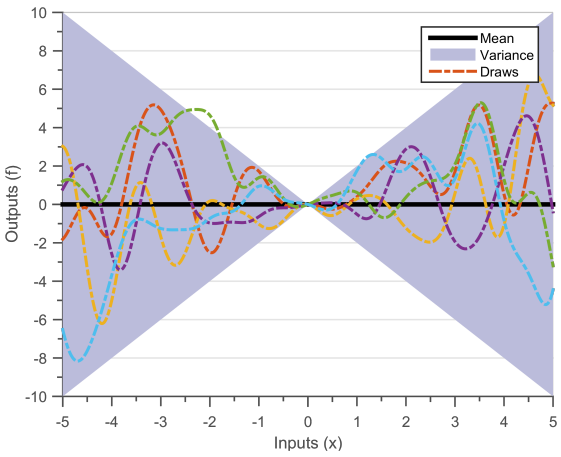
Figure 5.1(a) shows random draws obtained using k_{Multi} covariance, the hyper-parameters of the linear kernel are $w_0 = 0$ and $w_1 = 1$, this means that there is no intercept and $k_{Lin}(x_1, x_2) = x_1 x_2$. The hyper-parameters of the SE part are $\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$, similar to figure 4.6(c). Since multiplying two kernels is an AND operation, k_{Multi} tends to zero at $x = 0$ since k_{Lin} is zero at $x = 0$.

Figure 5.1(a)

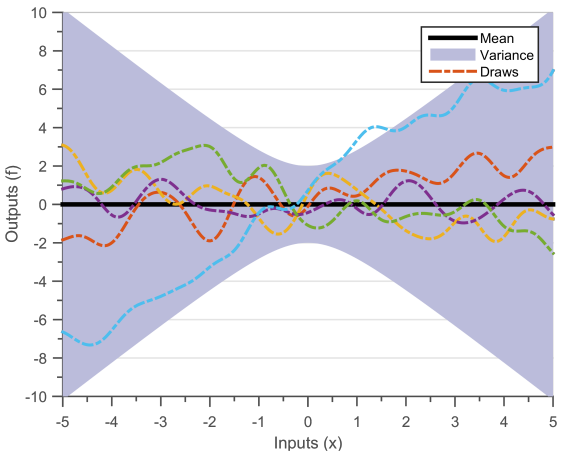
5.2.2 Adding Kernels

Adding two kernels acts as an OR operator, this means that the resulting kernel will have high value if either of the two kernels have high value [Durrande 2011].

Figure 5.1(b) shows the prior obtained after adding a Linear and a SE kernel. The hyper-parameters of the linear kernel are $w_0 = 0$ and $w_1 = 1$, this means that there is no intercept and $k_{Lin}(x_1, x_2) = x_1^T x_2$. The hyper-parameters of the SE part are $\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$, similar to figure 4.6(c).



(a) Draws from a GP prior with mean zero and kernel obtained by **multiplying** a Linear kernel with SE kernel. The hyper-parameters of the linear kernel are $w_0 = 0$ and $w_1 = 1$ while the hyper-parameters of the SE part are $\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$. We see that the variance at $x = 0$ goes to zero, since multiplication is an AND operation



(b) Draws from a GP prior with mean zero and kernel obtained by **adding** a Linear kernel with SE kernel. The hyper-parameters of the linear kernel are $w_0 = 0$ and $w_1 = 1$ while the hyper-parameters of the SE part are $\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$. We see that the variance at $x = 0$ goes to variance of SE kernel, since multiplication is an OR operation

Figure 5.1: Random draws from combining a Linear and SE kernel. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior. Random functions drawn from a linear GP are linear.

Adding Noise

The Linear kernel discussed in section 4.3.1 is a sum of three covariance functions; a constant covariance (w_0), a linear covariance ($w_1 x_1^T x_2$) and a white noise covariance function ($\sigma_n^2 \delta_{xx'}$) (equation 5.11). Similarly, the noisy posterior case discussed in section 2.3 is a case of adding a SE kernel with a white noise kernel, a heteroscedastic[2] noise model can be similarly created by adding several kernels together.

$$k(x_1, x_2) = w_0 + w_1 x_1^T x_2 + \sigma_n^2 \delta_{xx'} \tag{5.11}$$

An interesting consequence of adding kernels is that now we can decompose the result into additive parts. Suppose k_{Sum} is a covariance function by adding n covariance functions $k_1, k_2, \ldots, k_n$ (equation 5.1), then the posterior mean and covariance can be written as equation 5.12 and 5.13.

[2]Noise depending on the inputs

$$\mathbf{E}[f(x) \mid \boldsymbol{X}, \boldsymbol{y}, k_{Sum}] = \sum_{i=1}^{n} \boldsymbol{k_i}(\boldsymbol{x_* X})(\boldsymbol{K_{Sum}}(\boldsymbol{X}, \boldsymbol{X}))^{-1}\boldsymbol{y} \tag{5.12}$$

$$Cov[f(x) \mid \boldsymbol{X}, \boldsymbol{y}, k_{Sum}] = \sum_{i=1}^{n} \left[k_i(x_* x_*) - \boldsymbol{k_i}(\boldsymbol{x_* X})(\boldsymbol{K_{Sum}}(\boldsymbol{X}, \boldsymbol{X}))^{-1}\boldsymbol{k_i}(\boldsymbol{X x_*})\right] \tag{5.13}$$

Note that the precision matrix $(\boldsymbol{K_{Sum}}(\boldsymbol{X}, \boldsymbol{X}))^{-1}$ cannot be decomposed into its constituent covariance matrices. This is not an issue since this strategy is used to separate/discover individual effects in the data-set. For example, if we make a covariance function by adding a Linear kernel and an SE kernel, we can seperate the Linear part of the data-set from the non-linear part.

Discovering pattern

This comes very handy while iteratively discovering structure in the data. One method to make an optimal covariance structure by hand is to iteratively add new kernels until the posterior error variance represents a white noise. [Rasmussen 2005] use a sum of several kernels to interpolate CO_2 content in the atmosphere through the years. [Durrande 2013b] use a sum of periodic kernels to detect which genes are responsible for the 24-hour cycle (*circadian rhythm*) in *arabidopsis* plant. [Duvenaud 2013, Lloyd 2014] propose to automatically detect pattern by adding new kernels, and using the BIC measure to find the optimal covariance structure.

5.2.3 Change-Point kernels

The CP kernel was introduced to recognize changes in regimes, and adapt the covariance function accordingly. They were initially introduced to identify change points in time-series modelling [Osborne 2010, Saatçi 2010]. These kernels can be defined through addition and multiplication with sigmoidal functions (equation 5.14).

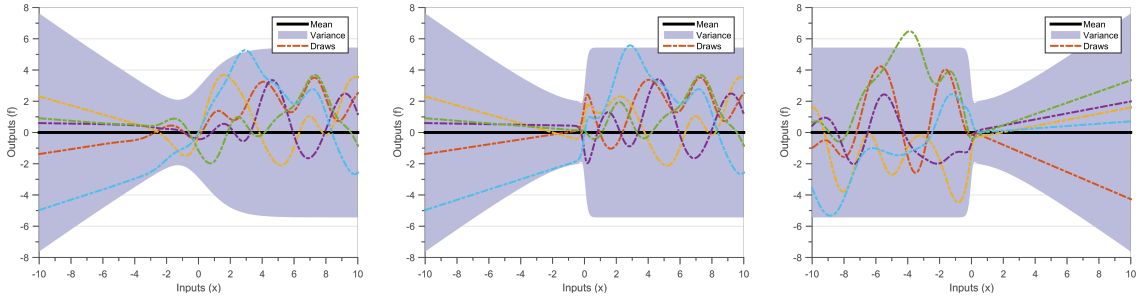
$$sigm(x, \theta) = \frac{1}{\theta_{intensity} + e^{\theta_{changeLocation} - x}} \tag{5.14}$$

$$k_{CP}(k_1, k_2, x_1, x_2) = sigm(x_1)k_1 sigm(x_2) + (1 - sigm(x_1))k_2(1 - sigm(x_2)) \tag{5.15}$$

Hyper-parameters

The hyper-parameters of this kernel are the $\theta_{changeLocation}$ which determines the location of the change point, $\theta_{intensity}$ determine the intensity of change between the two patterns, and the hyper-parameters of covariance functions k_1 and k_2. Figure 5.2 shows the randomly drawn functions from a CP kernel for varying values of $\theta_{intensity}$, where the regime changes from a Linear kernel to a SE kernel. The hyper-parameters of the linear kernel are $w_0 = 0$

and $w_1 = 1$, this means that there is no intercept and $k_{Lin}(x_1, x_2) = x_1^T x_2$. The hyper-parameters of the SE part are $\theta_{amplitude} = 1$ and $\theta_{lengthScale} = 1$, similar to figure 4.6(c). We see that as the value of $\theta_{intensity}$ increases the regime change happens more rapidly, while if $\theta_{intensity}$ changes sign then the order of regime reverses.



(a) Draws from a GP prior with mean zero and CP kernel (equation 5.15) between a Linear and a SE kernel with $[\theta_{intensity}, \theta_{changeLocation}] = [1, 0]$.

(b) Draws from a GP prior with mean zero and CP kernel (equation 5.15) between a Linear and a SE kernel with $[\theta_{intensity}, \theta_{changeLocation}] = [10, 0]$. Notice if the value of $\theta_{intensity}$ increases then the change between two patterns becomes more significant.

(c) Draws from a GP prior with mean zero and CP kernel (equation 5.15) between a Linear and a SE kernel with $[\theta_{intensity}, \theta_{changeLocation}] = [-10, 0]$. Notice if the sign of $\theta_{intensity}$ changes then the order of kernels gets reversed.

Figure 5.2: Random draws by having a change-point between a Linear and SE kernel. The solid black line defines the mean function, shaded blue region defines 95% confidence interval (2σ) distance away from the mean. The dashed lines represent five functions drawn at random from a GP prior. Random functions drawn from a linear GP are linear.

5.2.4 Application: Identifying onset of non-linearity using CP kernel

Several physical processes can be represented using a linear approximation in some part of their regime. This approximation is possible because the linear effects dominate in that part of the regime, but they eventually wear off and second and third order effects start becoming more powerful.

Linear assumption

This basic assumption is used in making simple models in several domains; for example in a material during the elastic regime $Stress \propto Strain$ is a basic linear approximation. The proportionality constant between Stress and Strain is called Young's Modulus, which is unique for each material. When the elastic regime starts wearing off, non-linear behaviour called plasticity starts taking over and the approximation $Stress \propto Strain$ is not valid anymore. Similarly in aerodynamics, when characterizing a flow over an airfoil during the

Linear regime $Lift \propto AngleOfAttack$, the airflow is attached on the airfoil during this regime. When the airflow starts separating from the airfoil the non-linear effects start dominating.

Contribution

The value of these basic physical parameters such as slope (eg. Young's Modulus, coefficient of lift) and location of change in regime (eg. start of plasticity and separation of flow) are progressively fed into further simulations. Generally, the slope and location of change points are evaluated using engineering judgment, this is a slow and costly process. We propose to estimate the slope and location of change-point automatically using a GP with CP kernel. Using the CP kernel which transitions from linear domain (linear kernel) to non-linear domain (SE kernel), prior information of the transition is encoded in the kernel structure.

We perform our experiments on openly available Stress and Strain data of Aluminum Alloy 6061 [Kaufman 1999] and Lift and Angle data of NACA 0012 airfoil[a]. To estimate the CP hyper-parameters, we again perform a 10-fold cross validation. The data-set will be randomly partitioned into 10 subsets containing an equal number of points. Of the 10 subsets, a single subset is retained as the test data-set, and the remaining 9 (10 - 1) subsets are used as training data. The cross-validation process is then repeated 10 folds, with each of the k subsets used exactly once as the validation data. The marginal-likelihood is optimized for each of the training data-set and location of change-point and slope of the linear regime is noted. We then compare their average with the values available in the literature.

[a]Airfoil data from: http://airfoiltools.com/airfoil/details?airfoil=n0012-il

Table 5.1 shows the results of physical parameters for AL 6061 when calculated using change-point kernel vs that available in the literature. We can observe that the change-point automatically predicts the correct values of Young's Modulus and start of non-linearity.

	Change-point	Literature
Young's Modulus (GPa)	68.5	68.9
Start of plasticity	0.94%	0.95%

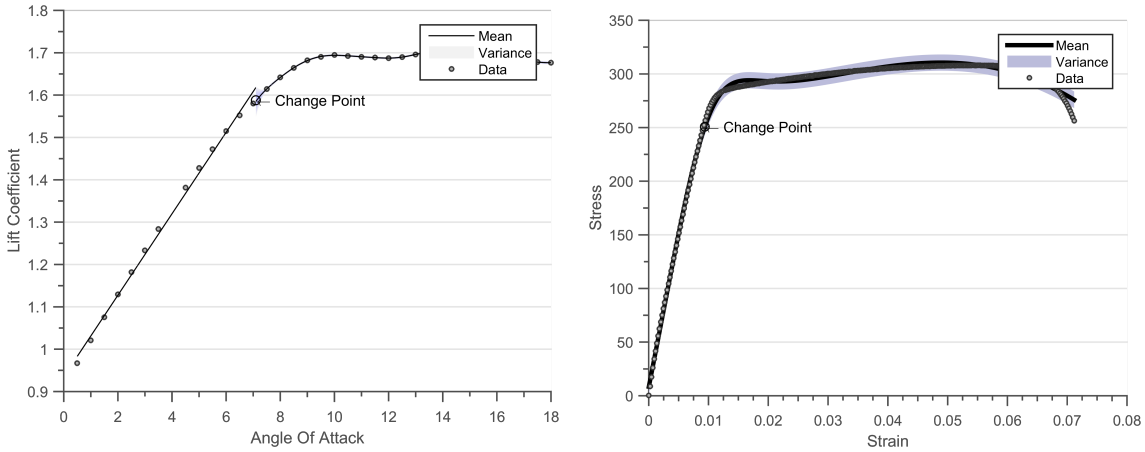
Table 5.1: Comparison of Young's Modulus for Al 6061 data-set

Figure 5.3

Figure 5.3 shows the posterior predictions when using the CP kernel. Figure 5.3(a) is the posterior distribution for the case of NACA 0012 airfoil, the Linear and SE regimes are plotted independently. Figure 5.3(b) shows the posterior distribution for the case of AL 6061. We can observe that the algorithm predicts a CP for the data-set, this is the point where the non-linear effects start dominating.

Multi-modality

The marginal likelihood of a change-point kernel has many local minimas, there is a local minimum at every observation point. This is because the kernel puts a non-linear regime at every observation point, hence a global optimizer should be used for optimization. The results of this study were be presented in the SIAM Uncertainty Quantification 2016 Conference [Chiplunkar 2016b].



(a) Posterior distribution for the case of NACA 0012 airfoil, the Linear and SE regimes are plotted independently.

(b) Posterior distribution for the case of AL 6061 alloy, the Linear and SE regimes are plotted independently.

Figure 5.3: Estimation of linear regimes using a change-point kernels

5.3 Multi-dimensional kernels

In this section we develop intuitions on how to build kernels for higher-dimensional inputs. This section demonstrates what happens when we add or multiply kernels across dimensions (equation 5.7 and 5.8), while also providing kernels on how to perform sensitivity analysis or to encode lower-dimensional structure (equation 5.4). We then apply the multi-dimensional covariance function to interpolate aerodynamic pressure (section 5.3.5), comparing the accuracy of GP interpolation with common POD technique [Chiplunkar 2017a].

5.3.1 Adding across dimensions

Consider an input data-set which is multi-dimensional $\boldsymbol{x} \in \mathbb{R}^{D_{inputs}}$. A simple additive kernel can be constructed by adding the kernels for individual dimensions [Hastie 1990]. This operation encodes the information that added dimensions are independent of each

other (equation 5.16).

$$k(\boldsymbol{d}, \boldsymbol{\theta}) = \sum_{i=1}^{D_{inputs}} (\theta^i_{amplitude})^2 exp\left[-\frac{(d^i)^2}{2(\theta^i_{lengthScale})^2}\right] \tag{5.16}$$

Here, $d^i = x^i_1 - x^i_2$ is the distance between two input points at the i^{th} dimension. Figure 5.4(b) is a randomly drawn function after adding two SE kernels, the hyper-parameters of both the SE kernels are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 0.2$.

5.3.2 Multiplying across dimensions

If we want to include interactions between two dimensions then their kernels can be multiplied together (equation 5.8). A kernel which allows for interaction between all the possible D_{inputs}-dimensions can be constructed by multiplying all kernels for all the dimensions. The multi-dimensional Automatic Relevance Determination (ARD) kernel (equation 5.17) can be seen as a multiplication of several one-dimensional kernels with different length-scales [Rasmussen 2005]. It is called ARD because the value of length-scale determines which dimensions are more relevant.

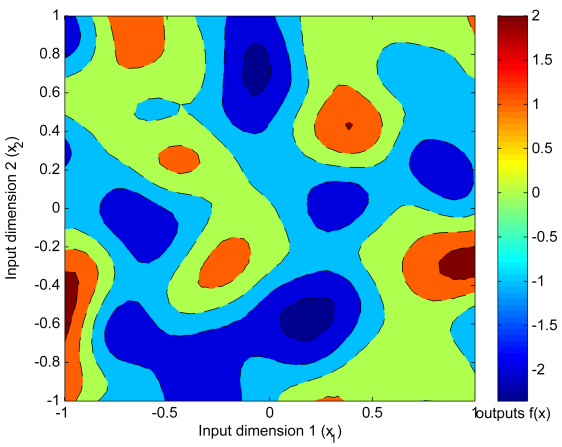
$$k(\boldsymbol{d}, \boldsymbol{\theta}) = (\theta_{amplitude})^2 \prod_{i=1}^{D_{inputs}} exp\left[-\frac{(d^i)^2}{2(\theta^i_{lengthScale})^2}\right] \tag{5.17}$$

Figure 5.4(a) is obtained after multiplying 2 SE kernels, the hyper-parameters of both the SE kernels are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 0.2$

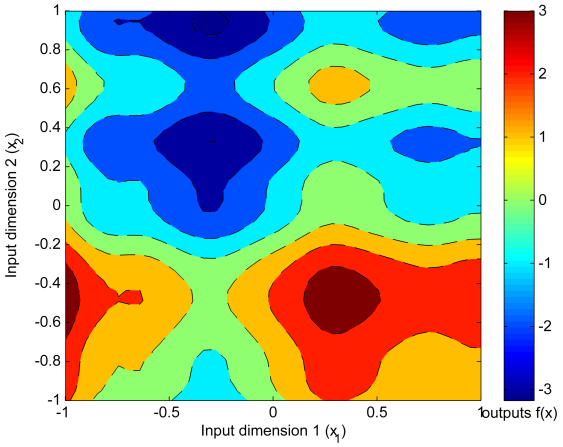
Matérn kernels have been found to have superior performance when compared to SE kernels on data-sets with high-dimensions [Le 2013]. It is argued that the Matérn kernel accounts for the concentration of measure effect in higher-dimensions. Since SE kernel is inverse Fourier of Gaussian density, high-dimensional samples of an SE kernel will be constrained to surface of the D_{inputs}-dimensional ellipse. This problem is less severe for a heavy tailed t-distribution power spectrum of Matérn kernel [Wilson 2014].

5.3.3 Sensitivity analysis

[Duvenaud 2011, Durrande 2013a, Chastaing 2015] define a class of additive kernels which are formed upon adding several low-dimensional interactions. Equation 5.18 is the basic form of covariance function which can be used to perform sensitivity analysis.



(a) Random draw from a 2 dimensional prior obtained after **multiplying** two SE kernels. The hyper-parameters of both the SE kernels are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 0.2$



(b) Random draw from a 2 dimensional prior obtained after **adding** two SE kernels. The hyper-parameters of both the SE kernels are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 1$

Figure 5.4: Random draw from a 2 dimensional prior.

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\theta)^2 \prod_{i=1}^{D_{inputs}} \left(1 + k^i(x_1^i, x_2^i)\right) \tag{5.18}$$

The above kernel will include all the possible interactions across dimensions. For example for a 3-dimensional input space, the above kernel will include all the first order terms (equation 5.19), all the second order terms (equation 5.21), and the third order term (equation 5.22).

$$\begin{aligned}
k_{first-order}(\boldsymbol{x_1}, \boldsymbol{x_2}) =& k^1(x_1^1, x_2^1) + k^2(x_1^2, x_2^2) + k^3(x_1^3, x_2^3) && (5.19)\\
k_{second-order}(\boldsymbol{x_1}, \boldsymbol{x_2}) =& k^1(x_1^1, x_2^1) \times k^2(x_1^2, x_2^2) + k^2(x_1^2, x_2^2) \times k^3(x_1^3, x_2^3) && (5.20)\\
&+ k^3(x_1^3, x_2^3) \times k^1(x_1^1, x_2^1) && (5.21)\\
k_{third-order}(\boldsymbol{x_1}, \boldsymbol{x_2}) =& k^1(x_1^1, x_2^1) \times k^2(x_1^2, x_2^2) \times k^3(x_1^3, x_2^3) && (5.22)\\
& && (5.23)
\end{aligned}$$

These kinds of kernels can be used to analyze the sensitivity of interactions between various dimensions (also called as ANalysis Of VAriance, ANOVA).

5.3.4 Low dimensional structure

We can also encode a low dimensional structure into family of functions by specifying the kernel as $k_{low} = k(\boldsymbol{x_1 H}, \boldsymbol{x_2 H})$ (equation 5.4), here $\boldsymbol{H}$ is a low rank matrix.

$$k_{low}(\boldsymbol{d}, \boldsymbol{\theta}) = (\theta_{amplitude})^2 exp\left[-\frac{1}{2}\boldsymbol{d\Sigma d}^T\right] \tag{5.24}$$

Here, $\boldsymbol{\Sigma}$ is $\boldsymbol{HH}^T/(\theta^2_{lengthscale})$ which encodes the low dimensional structure. Figure 5.3.4 is obtained after encoding a low-dimensional into SE kernels, the hyper-parameters of the SE kernel are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 0.2$ while $\boldsymbol{\Sigma} = \left(\begin{smallmatrix} 1 & 0 \\ -1 & 0 \end{smallmatrix}\right)$.

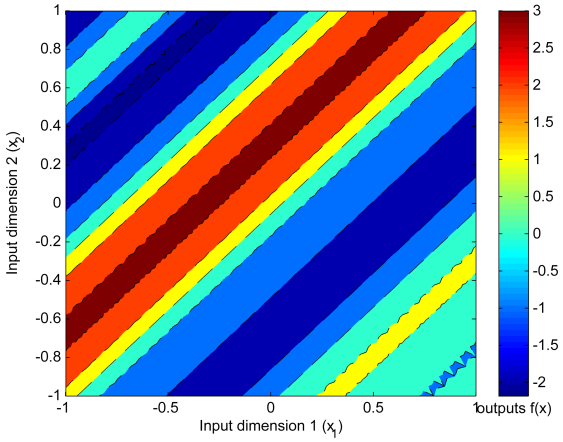


Figure 5.5: Random draw from a 2 dimensional prior which encodes a **low-dimensional** structure. The hyper-parameters of the kernel are $\theta_{amplitude} = 1$ and $\theta_{lengthscale} = 1$

When $\boldsymbol{\Sigma}$ is a diagonal matrix we get an ARD kernel

$$\begin{aligned} k(\boldsymbol{d}, \boldsymbol{\theta}) = (\theta_{amplitude})^2 exp\left[-\frac{1}{2}\boldsymbol{d\Sigma d}^T\right] &= (\theta_{amplitude})^2 exp\left[\sum_{i=1}^{D_{inputs}} -\frac{(d^i)^2}{2(\theta^i_{lengthScale})^2}\right] \\ &= (\theta_{amplitude})^2 \prod_{i=1}^{D_{inputs}} exp\left[-\frac{(d^i)^2}{2(\theta^i_{lengthScale})^2}\right] \end{aligned} \tag{5.25}$$

When the number of dimension increases significantly, the number of hyper-parameters also increases this makes the optimization of marginal likelihood inefficient. [Bouhlel 2016b] first reduce the dimensionality of the data-set and then perform interpolation thereby circumventing the problem of higher dimensions. [Garnett 2013, Tripathy 2016] use this covariance to reduce the dimensionality of input domain. *KPLS*

In the current section we have seen how to build covariance functions for multi-dimensional inputs. We now apply multi-dimensional kernels to build a surrogate model for Aerodynamic pressures. We validate our method on 2 test cases: the first in subsonic regime on a Flap Track Fairing (FTF) and the second in transonic regime on NASA's Common Research Model (CRM) Wing. The reader can also note that the results of these experiments were used during a recent Airbus Flight test campaign.

5.3.5 Application: Interpolation of aerodynamic pressures

Accurate prediction of aerodynamic pressures at a flight configuration is computationally expensive. Hence, it becomes advantageous to use surrogate models as approximations of high-fidelity aerodynamic models. A popular method of surrogate modelling in the aerodynamics community is by interpolating Reduced Order Models (ROM). A set of aerodynamic pressure snapshots is generated by performing CFD simulations for different aerodynamic parameters (eg. angle of attack α, Mach). Then orthogonal basis vectors are found in the parameter space for the set of pressures snapshots. Generally, Proper Orthogonal Decomposition (POD) [Tan 2003, Rosenbaum 2013, Braconnier 2011] (also called as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD)) is used to find the linear subspace. Finally, the reduced models are interpolated at the desired point in the parameter-space [Beckert 2001, Barrault 2004].

Pressure snapshot

Let us first start by defining a pressure snapshot. There exists a 3 dimensional spatial vector $\boldsymbol{\omega_i} \in \mathbb{R}^3$ such that $\boldsymbol{\omega_i} = \{(\omega_i^1, \omega_i^2, \omega_i^3)\}$. Here, $i \in [1, N_{nodes}]$ are the spatial coordinates of the i^{th} pressure node in a CFD mesh containing N_{nodes} pressure nodes. Similarly there exists a D dimensional parameter vector $\boldsymbol{d_j} \in \mathbb{R}^D$, for $\boldsymbol{d_j} = \{(d_j^1, d_j^2, \ldots, d_j^D)\}$. Here, $j \in [1, N_{experiment}]$ correspond to the j^{th} parameter set, while D corresponds to the number of parameters. The parameters can be any parameters which are desired to be interpolated, some common examples include Mach, Angle of Attack for steady aerodynamics and time or frequency for unsteady aerodynamics. We will only concentrate on interpolating steady aerodynamics in this section.

The pressure measured on the i^{th} pressure node for the j^{th} parameter set will be denoted as $p_j(\boldsymbol{\omega_i})$ defined by equation 5.26. We next define the matrix $\boldsymbol{\Omega} = \{\boldsymbol{\omega_1}; \boldsymbol{\omega_2}; \ldots; \boldsymbol{\omega_{N_{nodes}}}\}$ for $\boldsymbol{\Omega} \in \mathbb{R}^{N_{nodes} \times 3}$ containing the full spatial information of the aerodynamic mesh. Finally, the pressure snapshot for the CFD/experiment run j will be denoted as $\boldsymbol{P_j(\Omega)} = \{p_j(\boldsymbol{\omega_1}); p_j(\boldsymbol{\omega_2}); \ldots; p_j(\boldsymbol{\omega_{N_{nodes}}})\}$ for $\boldsymbol{P_j(\Omega)} \in \mathbb{R}^{N_{nodes}}$ defined by the equation 5.27.

$$p_j(\boldsymbol{\omega_i}) = f_{pressure}(\boldsymbol{\omega_i}, \boldsymbol{d_j}) \tag{5.26}$$

$$\boldsymbol{P_j(\Omega)} = f_{pressure}(\boldsymbol{\Omega}, \boldsymbol{d_j}) \tag{5.27}$$

Here, $f_{pressure}$ symbolizes the aerodynamic process which when applied to a spatial location ($\boldsymbol{\omega}$) and an aerodynamic parameter ($\boldsymbol{d}$) gives us the pressures, we eventually wish to approximate this process ($f_{pressure}$). The POD methodology decomposes the set of pressure snapshots ($\boldsymbol{P_j(\Omega)}$) into their eigen vectors ($\boldsymbol{\phi^l(\Omega)}$) and participation factors ($a^l(\boldsymbol{d_j})$). To reconstruct the pressure snapshot at a new point $\boldsymbol{d_{new}}$, the participation factors are interpolated and then linearly combined to give the new pressure snapshot (equation 5.28). For more details please refer to appendix B

POD

$$\boldsymbol{P_{new}(\Omega)} = \sum_{l=1}^{p} a^l(\boldsymbol{d_{new}})\boldsymbol{\phi}^l(\boldsymbol{\Omega}) \tag{5.28}$$

Due to the assumption of linear subspace, interpolation through ROM is highly efficient both in terms of cost and performance in the subsonic regime [Verveld 2016]. Unfortunately in the transonic regime, the shock creates a highly non-linear, almost discontinuous pressure distribution and the assumption of linear subspace does not hold [Li 2016]. Although the performance of ROM interpolators can be improved with larger number of samples [Franz 2014, Forrester 2008], we propose to improve the accuracy of prediction using the Distributed GPs regression for the same number of samples.

Contribution

We interpolate the pressure $p_j(\omega_i)$ by simply multiplying the kernels across dimensions (equation 5.29). To interpolate in subsonic regime we multiply Matérn $\nu = 5/2$ across dimensions, a Matérn kernel is chosen since it does not suffer from concentration of measure effect in higher dimensions. We are not interested in linearly separating the effects of individual dimensions, hence a simple multiplicative kernel will suffice for this problem.

$$\Pr[p_j(\omega_i)] = GP\left(0, k(x_1, x_2) = \prod_{i=1}^{D_{inputs}} k_{Mat(\nu=5/2)}(x_1^i, x_2^i)\right) \tag{5.29}$$

The transonic regime often contains a shock on the surface of the wing. This shock almost creates a discontinuity in the pressure field. We know from section 4.3.2 that Neural Network kernels can represent these type of almost discontinuous functions in its hypothesis space. Hence to interpolate in transonic regime we use the Neural Network kernel in the dimension of shock. For example, if we know that the shock appears on the chord-wise direction, then we can represent the pressure as equation

5.30.

$$\Pr[p_j(\boldsymbol{\omega_i})] = GP\left(0, k(\boldsymbol{x_1}, \boldsymbol{x_2}) = k_{NN}(x_1^{chord}, x_2^{chord}) \times \prod_{i=1}^{D_{inputs}-1} k_{Mat(\nu=5/2)}(x_1^i, x_2^i)\right) \tag{5.30}$$

Here, x_i^{chord} represents the chord wise dimension of the i^{th} input, while k_{NN} and $k_{Mat\nu=5/2}$ represent the Neural Network and Matérn ($\nu = 5/2$) kernels respectively.

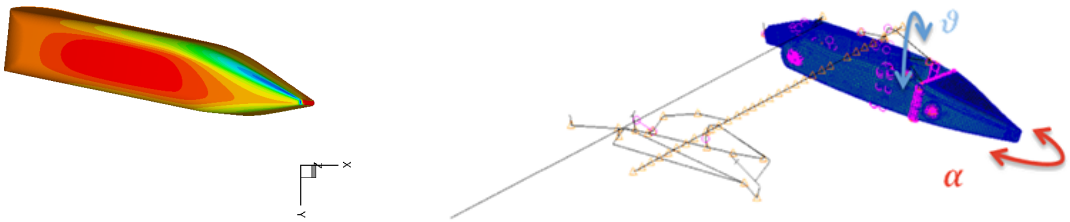
We now test the performance of Distributed GPs and POD+I on two sets of numerical experiments. Firstly, we test the accuracy on a detailed Flap Track Fairing (FTF) design [Bosco 2016] in subsonic regime (section 5.3.5) based on RANS simulation from the elsA solver [Cambier 2008]. Finally, we compare the accuracy on CRM wing in the transonic regime using the elsA solver and a k-Omega-SST turbulence model [Vassberg 2014]. elsA® [Cambier 2008] is a CFD simulation platform that allows representation of both internal and external aerodynamics from the low subsonic to the high supersonic flow regime.

Interpolation in subsonic regime

FTF

A FTF is situated below the wing and is used to deploy flaps for landing and take-off configurations. A FTF experiences heavy dynamic excitation due to the exhaust coming from engine. The dynamic nature makes the design of FTF a challenging task, where each simulation can last for 2 days. If we can effectively interpolate pressure snapshots then a 2 day dynamic simulation can be reduced to a few hours.

Interpolation of FTF pressure snapshots has been earlier studied using POD methodology [Bosco 2016]. In this section we use this data-set to validate the interpolation capabilities of a Distributed GPs algorithm.



(a) FTF Pressure snapshot (b) FTF parameters $\boldsymbol{d_i} = (\theta_i, \alpha_i)$

Figure 5.6: Details of the FTF

The FTF parameters chosen for this analysis are θ, the rotation around the longitudinal axis of the FTF, and α, the rotation around the *spigot* axis, main connection between the

track and the wing structure figure 5.6(b). The surface mesh of the CFD skin contains almost 36 thousand nodes (more precisely $N_{nodes} = 36802$). We run the simulation for 9 different values of α and 9 different values of θ ($N_{experiment} = 9 \times 9 = 81$). In this particular case Reynolds Averaged Navier-Stokes (RANS), equations are used with Spalart-Allmaras turbulence model to simulate the flow around the FTF. Figure 5.6(a) shows one particular pressure snapshot.

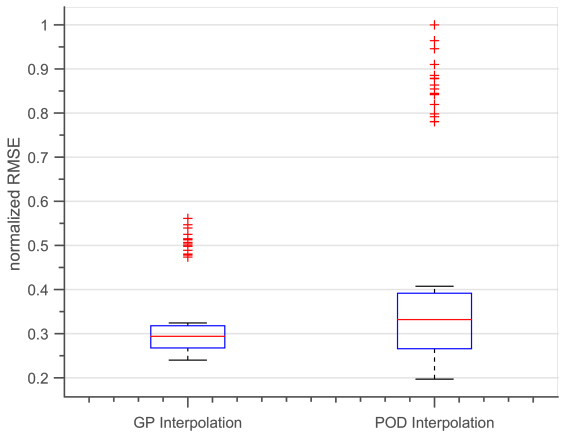
$\boldsymbol{d_i} = (\theta_i, \alpha_i)$

We use the Leave One Out (LOO) Cross Validation method to quantify the performance of the two methodologies. $[\alpha, \theta]$ pairs are removed one by one from the database to create a new training set. The new training set is used to perform interpolation according to POD+I and Distributed GPs. Pressure snapshot is reconstructed for the missing $[\alpha, \theta]$ pairs. The methods are finally compared by evaluating the Root Mean Square Error (RMSE) and time of prediction for each pair case.

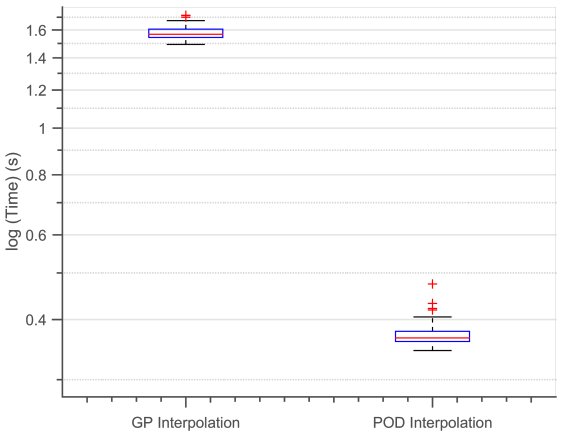
LOO-CV

While performing Distributed GPs regression, we learn a model between the input vector $\boldsymbol{x_{ij}} = [\omega_i^1, \omega_i^2, \omega_i^3, \alpha_j, \theta_j]$ and the pressures ($y_{ij} = p_{ij}$). We build a multi-dimensional kernel by multiplying 5 Matérn kernels, different length-scale for each dimension. As discussed earlier we propose to use Matérn kernel since the SE kernel has a very restrictive hypothesis space for high-dimensional regression. Effectively we are learning a GP model for $N = 36802 \times 80 = 2.9$ million data-points, there are 1000 points in each expert.

Distributed GPs



(a) Box plots of normalized RMSE for the two different model types. POD has a RMSE of 0.48 ± 0.27 whereas Distributed GPs has a RMSE of 0.37 ± 0.1. The mean Distributed GPs prediction is 20% better than POD. Outliers are extrapolation cases



(b) Time taken to perform prediction for the two different model types. POD takes $2.35s \pm 0.11s$ whereas Distributed GPs takes $37.84s \pm 4.35s$ to perform the interpolation. The average POD method is 19 times faster than Distributed GPs.

Figure 5.7: Results for elsA interpolation

Figures 5.7(a) denote the RMSE estimates for different pairs. For RMSE, the Distributed GPs performs marginally better than POD. The outliers in the box-plots denote cases where the removed pairs are on the edges of the domain. Since the removed pairs are

on the edges of the domain, there are no CFD snapshots surrounding these pairs. Hence we are extrapolating during the edge cases. POD has a RMSE of 0.48 ± 0.27 whereas Distributed GPs has a RMSE of 0.37 ± 0.1. The average Distributed GPs prediction is 20% better than POD. Figure 5.7(b) shows the time taken to perform prediction for the two model types. We are only comparing the time to perform interpolation, time to learn the models will be much longer. POD takes $2.35s \pm 0.11s$ whereas Distributed GPs takes $37.84s \pm 4.35s$ to perform the interpolation.

Figure 5.7

Although the GP technique can more efficiently capture non-linear behavior we see a relatively low improvement in performance for the amount of time invested. Deciding between the simple and time-tested POD method or costly and accurate Distributed GPs interpolation in the subsonic regime can be a delicate task and mostly depends on the preferences of the final user.

Interpolating in transonic regime

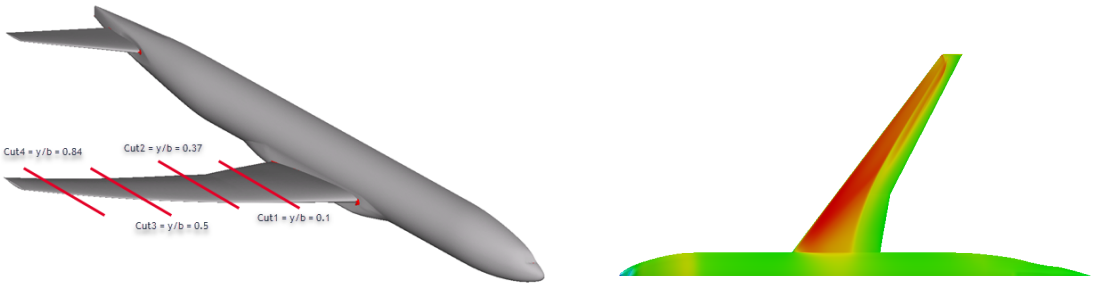
We now proceed to compare the accuracy of the two methods in transonic regime on the CRM wing model proposed by NASA. Since the introduction of the CRM for the 4th Drag Prediction Workshop [Vassberg 2014], the CRM has been become a very widely used test case for applied computational aerodynamics. Due to the widespread experience and availability of wind-tunnel test results for the CRM configuration, it is a natural case to benchmark interpolations.

Shocks

Due to the shape of an airfoil, airflow is accelerated on the upper surface of the wing. This causes shocks to appear on the upper surface of the wing in the transonic regime. Shocks are sudden changes (almost discontinuous) in pressure and are important for estimating the performance of the aircraft [Jameson 1974, Cole 2012]. Moreover, an aircraft flies in the transonic regime for 80% of its journey (cruise). Hence accurate prediction of location and strength of a shock is very important during design. Since POD is a linear subspace reduction method it has difficulties in interpolating a discontinuous shock regime [Verveld 2016].

Again we use the elsA® solver to perform simulations on the design. We use the $\kappa - \omega$ SST turbulence model to perform predictions since it has good performance in the fuselage wing interaction regions [Menter 2003, Vassberg 2014]. The CFD was run for a combination of 21 values of $\alpha \in [1 : 0.1 : 3]$ and 5 values of $Mach \in [0.84 : 0.005 : 0.86]$, hence $N_{experiment} = 21 \times 5 = 105$. Figure, 5.8(b) shows one of the pressure snapshots for $\alpha = 2$ and $Mach = 0.85$. We then cut the wing at four distinct locations $y/b = [0.105, 0.37, 0.5, 0.84]$ (figure 5.8(a)) to clearly observe different types of aerodynamic behavior. Here, y denotes the y-distance from aircraft axis and b denotes the span of one wing.

As detailed in the earlier section we again use LOO-CV for comparing the performance of the two methods. The POD+I method has been run as described in ap-



(a) CRM. The four red lines are the four cuts at $y/b = [0.105, 0.37, 0.5, 0.84]$. Here, y denotes the y-distance from aircraft axis and b denotes the span of one wing.

(b) Pressure snapshot on the CRM for $\alpha = 2$ and $Mach = 0.85$, using elsa solver and $\kappa - \omega$ SST turbulence model. We can observe double shock pattern appearing on the outer sections of the wing

Figure 5.8: CRM shape for aerodynamic simulations

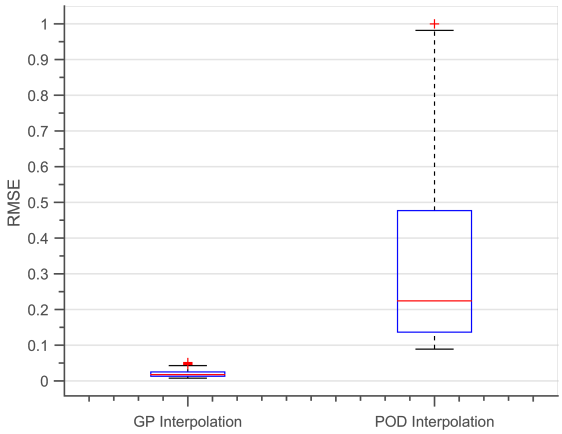
pendix B. For GP regression, we learn a GP model between $y_{ij} = p_{ij}$ and input vector $\boldsymbol{x_{ij}} = [x_i^{chord}, \alpha_j, Mach_j]$. We build a multi-dimensional kernel by multiplying 2 Matérn kernels (for α and $Mach$ dimensions) and one Neural Network kernel (for x^{chord} dimension). The shock will appear in the spatial dimension and hence using a Neural Network kernel in that dimension lets us capture the discontinuity more accurately.

Figure 5.9(a) denote the RMSE estimates for different pairs. POD has a RMSE of 0.32 ± 0.23 whereas Distributed GPs has a RMSE of 0.02 ± 0.01. The average Distributed GPs prediction is 16 times better than POD in transonic regime. The outliers in the box-plots denote cases where the removed pairs are on the edges of the domain. Since the removed pairs are on the edges of the domain, there are no CFD snapshots surrounding these pairs. Hence we are extrapolating on the outliers. Figure 5.9(b) shows the time taken to perform prediction for the three different model types. POD takes $0.5s \pm 0.03s$ whereas Distributed GPs takes $40.6s \pm 2.3s$ to perform the interpolation. In transonic regime GP has a significantly better error performance and becomes the obvious choice for interpolation.

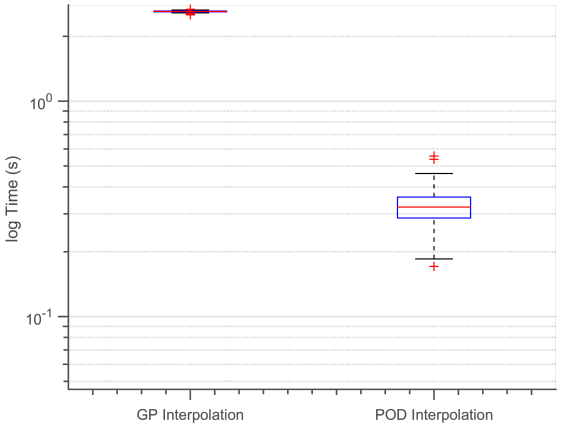
Figure 5.9

Figure 5.10(a) shows the comparison between predicted pressures using the POD method and the Distributed GPs for case of interpolation. Reconstruction is performed on the pressure snapshot at $\alpha = 2$ and $Mach = 0.85$ for the $y/b = 0.105$. We can observe that the shape of shock has been smoothed out by the POD method. Figure 5.10(b) shows comparison between POD method and Distributed GPs for extrapolation, the extrapolation is being performed for the aerodynamic parameter ($Mach$) and not the spatial parameter. Reconstruction is performed on the hidden pressure snapshot of $\alpha = 2$ and $Mach = 0.84$ which is an extrapolation case. We can observe that POD introduces errors both for the intensity of shock, and location of shock for the extrapolation case, this explains the high amount of error in figure 5.9(a).

Figure 5.10



(a) Normalized RMSE for the two different model types. POD has a RMSE of 0.32 ± 0.23 whereas Distributed GPs has a RMSE of 0.02 ± 0.01. The average Distributed GPs prediction is 16 times better than POD in transonic regime. The outliers in the box-plots denote cases where the removed pairs are on the edges of database. Since the removed pairs are on the edges of database matrix, there are no CFD snapshots surrounding these pairs. Hence extrapolation is performed during the edge cases.



(b) Time taken to perform prediction for the two different model types. POD takes $0.5s \pm 0.03s$ whereas Distributed GPs takes $40.6s \pm 2.3s$ to perform the interpolation. This is the time to perform prediction and not learning the model. The average POD method is 80 times faster than Distributed GPs.

Figure 5.9: Results for CRM interpolation

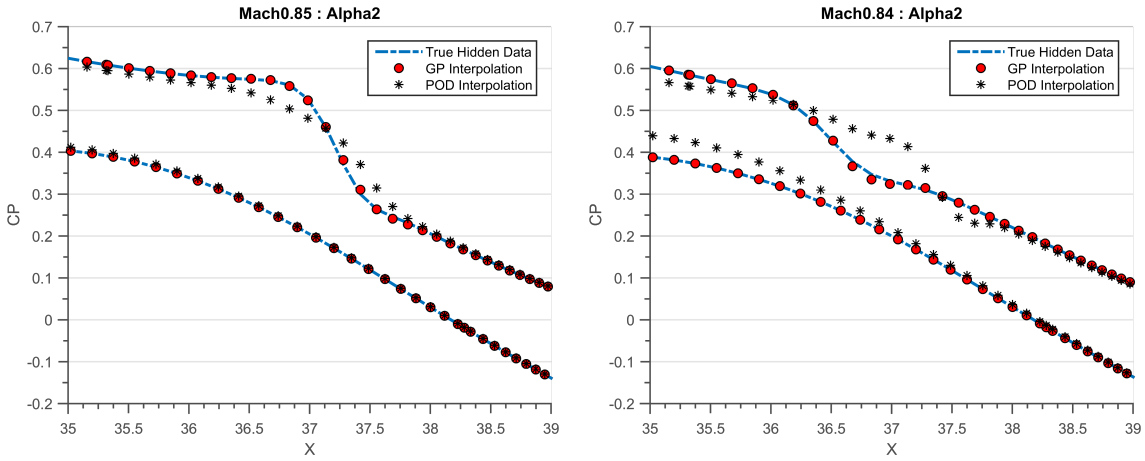
Comparison across cuts

We next study the accuracy of Distributed GPs for different airfoils on the wing. Using the methodology described earlier we build a Distributed GPs model for each airfoil and measure the accuracy of interpolation performed for each cut using the LOO-CV methodology.

Figure 5.11

Figure 5.11(a) shows the RMSE performance across cuts. The performance of interpolation deteriorates as we go further away from the fuselage. This is primarily because as we go further away from the fuselage double shocks start appearing on the airfoil as observable from figure 5.8(b). Figure 5.11(b) shows interpolation performed by the POD method and Distributed GPs methods at $\alpha = 2$ and $Mach = 0.85$, for the last cut ($y/b = 0.84$). While, POD smooths out the double shock pattern, Distributed GPs lacks the accuracy observed in figure 5.10(a).

Figures 5.12(a) and 5.12(b) show the evaluation of pressures upon varying $\alpha \in [1, 3]$ at locations $y/b = 0.105$ and $y/b = 0.84$ respectively. The color coding denotes coefficient of pressure for the upper side of airfoil, The x-axis denotes chord-wise location and y-axis denotes α. White lines denote presence of a pressure snapshot due to CFD run, everything



(a) Comparison between POD method and Distributed GPs for **interpolation**. The X-axis denotes chord dimension, only showing chord-section near shock for clarity. The Y-axis denotes the coefficient of pressure. Reconstruction is performed on the pressure snapshot at $\alpha = 2$ and $Mach = 0.85$ for the $y/b = 0.105$. We can observe that the intensity of shock has been smoothed out by POD method

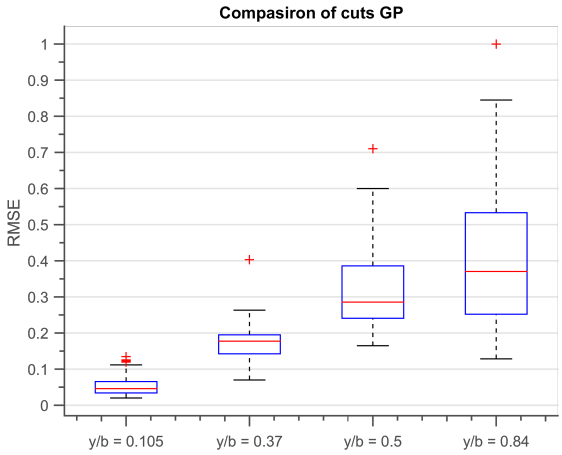
(b) Comparison between POD method and Distributed GPs for **extrapolation** (the extrapolation is being performed for the aerodynamic parameter ($Mach$) and not the spatial parameter). The X-axis denotes chord dimension, only showing chord-section near shock for clarity. The Y-axis denotes the coefficient of pressure. Reconstruction is performed on the pressure snapshot at $\alpha = 2$ and $Mach = 0.84$ for the $y/b = 0.105$. We can observe that POD introduces errors both for the intensity of shock and location of shock for this case

Figure 5.10: Comparison of pressure interpolations for first cut $y/b = 0.105$. Here we compare the accuracy of prediction for interpolation and extrapolation cases.

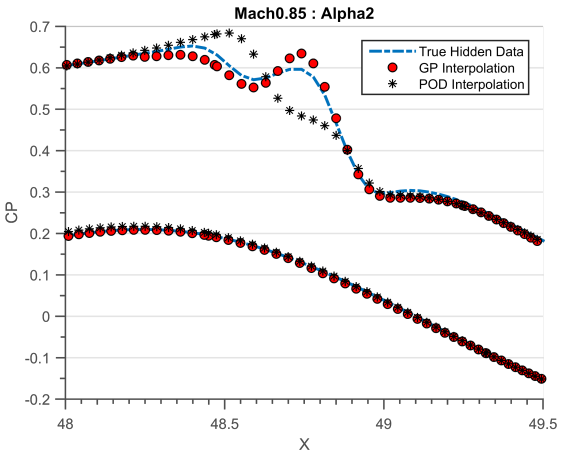
in between is interpolation. Dashed black lines denote constant pressure contours Color between two contours has been smoothed for clarity. For figure 5.12(a) we observe a strong shock near $\alpha = 3$ which slowly gets converted to a weak shock near $\alpha = 1$. The presence of a single shock is also the reason why Distributed GPs performs better at this cut location. For figure 5.12(b) we observe a single shock near $\alpha = 3$ which slowly gets converted to a double shock pattern. The zone from single to double shock is a very interesting point for performance, since the wing drag is minimum during this transition phase. Distributed GPs starts performing badly near the transition phases, this can be observed by the small pools of $C_P = 0.5$ at the transition phase from single to double shock.

Double shocks

The current section presents a comparison between two different surrogate model building methods: time-tested surrogate modelling methods such as POD coupled with spline interpolation and upcoming machine learning methods such as Distributed GPs for subsonic and transonic regimes. The Distributed GPs performs marginally better than POD technique in subsonic regime but is many times slower. On the contrary for the case of transonic regime Distributed GPs clearly outperforms POD+I method. This is mainly due



(a) Normalized RMSE for different airfoils based on Distributed GPs. The mean RMSE of different cuts from $y/b = [0.105, 0.37, 0.5, 0.84]$ is $[0.053, 0.17, 0.31, 0.40]$ respectively. The accuracy of interpolation deteriorates as we go farther away from the fuselage. This is due to appearance of double shock on the outer section of wing.



(b) Comparison between POD method and Distributed GPs for interpolation. The X-axis denotes chord dimension, only showing chord-section near shock for clarity. The Y-axis denotes the coefficient of pressure. Reconstruction is performed on the pressure snapshot at $\alpha = 2$ and $Mach = 0.85$ for the location $y/b = 0.84$. While POD smooths out the double shock pattern Distributed GPs also lacks the accuracy observed in figure 5.10(a).

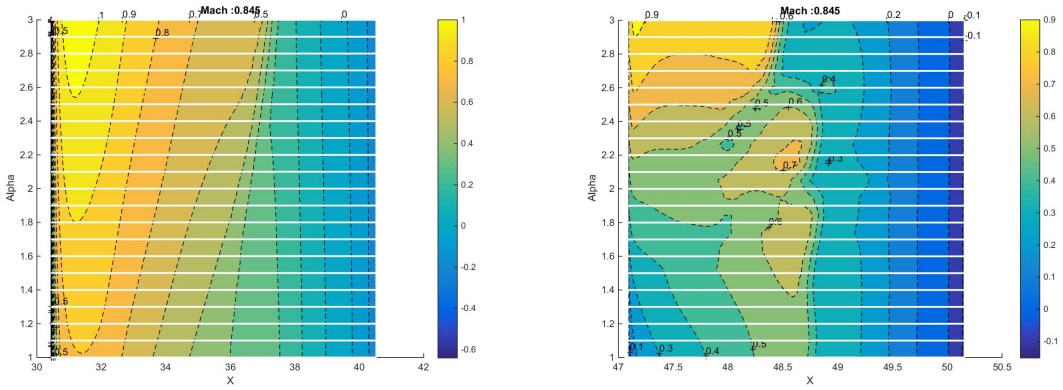
Figure 5.11: Performance of Distributed GPs across cuts

to the presence of shock on the airfoil.

Plots like figure 5.12(b) give a quick understanding of the flight regime for very few simulation runs. These plots can also be used to find transition regimes from single shock to double shock, which are very interesting performance points. In the future we wish to improve reconstruction of double shock patterns by improving the length scale and improve the choice of experts for Distributed GPs.

5.4 Summary and discussion

In the last two chapters we have tried to answer the question: How to add *a-priori* information of a pattern in a learning algorithm? We present a sample of the wide variety of available covariance functions. Due to the ability to create new kernels, encoding prior information into the structure can be performed easily. If we have an *a-priori* information about the pattern of function to be learned, then embedding this information into the GP algorithm greatly improves accuracy and cost of prediction. Table 5.2 lists a few commonly known combinations of covariance functions in the literature.



(a) Interpolation performed at constant $Mach = 0.845$ and $\alpha = [1, 3]$ for the location $y/b = 0.105$. The color coding denotes coefficient of pressure for upper side of airfoil, The x-axis denotes chord-wise location and y-axis denotes α. White lines denote presence of a pressure snapshot due to CFD run, everything in between is interpolation. Dashed black lines denote constant pressure contours color between two contours has been smoothed for clarity. We observe a strong shock near $\alpha = 3$ which slowly gets converted to a weak shock near $\alpha = 1$.

(b) Interpolation performed at constant $Mach = 0.845$ and $\alpha = [1, 3]$ for the location $y/b = 0.84$. The color coding denotes coefficient of pressure for upper half of airfoil, The x-axis denotes chord-wise location and y-axis denotes α. White lines denote presence of a pressure snapshot due to CFD run, everything in between is interpolation. Dashed black lines denote constant pressure contours color between two contours has been smoothed for clarity. We observe a single shock near $\alpha = 3$ which slowly gets converted to a double shock pattern.

Figure 5.12: Pressure reconstructions for constant $Mach = 0.845$ and sweeping $\alpha \in [1, 3]$

Model	Structure	Citation
Linear Regression	$k_{constant} + k_{linear} + k_{noise}$	
Polynomial	$k_{constant} + \prod k_{linear} + k_{noise}$	
Ordinary kriging	$k_{SE} + k_{noise}$	[Krige 1951]
Simple kriging	$k_{constant} + k_{SE} + k_{noise}$	
Universal Kriging	$\prod k_{linear} + k_{SE} + k_{noise}$	[Matheron 1963]
Multiple Kernel	$\sum k_{SE} + k_{noise}$	
Spectral Mixture	$\sum cosk_{SE} + k_{noise}]$	[Wilson 2013]
Change point	$\sum CP(k_{Lin}, k_{SE}) + k_{noise}]$	[Osborne 2010]
Additive GPs	$\prod_i (1 + k^i_{SE})]$	[Duvenaud 2011]

Table 5.2: Combination of covariance functions available in literature

Section 5.2 describes how to combine kernels to create one-dimensional kernels. While section 5.2.4 describes how to build kernels for higher dimensions. There are two main contributions of this chapter, first we develop a novel methodology to detect the start of

a non-linear regime in an engineering design problem. This is thanks to the change-point kernel which lets us define change from one regime to another [Chiplunkar 2016b]. Second, we encode the information of shock to interpolate pressures in the transonic regime. This strategy gives significant gains in accuracy when compared to the standard POD+I for aerodynamic interpolation [Chiplunkar 2017a].

In the next two chapters we will tackle the remaining questions posed in section 1.6 of the thesis. Chapter 6 discusses how to perform extrapolation given a computer simulation of experiments, while chapter 7 discusses how to encode prior information of relationships between measurements into a GP regression.