POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Civile

Tesi di Laurea Magistrale

Detection of Cracks in Beams Using Treed Gaussian Processes

Relatori:
Prof. Cecilia Surace
Corr. Ing. Marco Civera

Candidata:
Chiara Longhitano

A.A. 2017-2018

Contents

Abstract

Detection of crack is a relevant and interesting topic. Methods through this field is studied are various.
In this work a new and very leading-edge method to locate the crack is treated: Treed Gaussian process. This method is an evolution of the Gaussian process, used in the past. The notable advantage lies in the fact that, whereas in the GP the curvature of the measured mode shape was used for the analysis, with TGP model the measured mode shape itself is employed. This involves a reduction of the amplification of the noise due to differentiation compared to the GP method. Nevertheless, GP method remains absolutely essential since TGP splits the input dataset in homogeneous regions where single GP methods are applied. In each region the covariance function is the same, save in the region where damage is localized. The split point should locate the crack. The Genetic Algorithm aims at optimizing the tree prior parameters and improving the achievement of the results.

Keywords: Structural health monitoring (SHM); Damage detection; Gaussian processes (GPs); Treed Gaussian process (TGPs); Optimization Techniques.

Chapter 1

Introduction

During the last years the identification of damage in a structure has been a matter that continues to be of considerable importance for the Structural Health Monitoring (SHM) [5].
The strategy used to investigate damage from SHM, in dynamics field, was focused on the natural frequencies.
The principal issue of this method is that frequencies are influenced by the environment, also when non particular phenomenon takes place, and specifically when crack occurs.
The main aim of the research is to detect and locate the numbers of cracks in beams by discerning singularities in measured mode shape.
It is shown that crack introduces a discontinuity in the mode shape. Usually human eye is not able to detect this discontinuity without the use of particular means.
Since in this study a beam-like structure will be concerned with, the discontinuity has been detected analyzing the second derivative, which represents the local curvature of the beam. Analyzing the second derivatives in the process of detection of the crack the noise is amplified due to the process of discrete differentiation.
So the base idea of the presented method is the use of measured mode shape in order to overcome the problem related to natural frequencies of the structure and the problem of amplification of noise of the curvature.
A method with does not need differentiation of the mode shape is the Gaussian process (GP) regression method.
In this work a development will be used, well known as *Treed Gaussian process* (TGP).
It allows to detect not only the location of the crack but also the number of them.

Both GP and TGP fall under machine learning field , where a computer systems has the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed and producing best possible outputs [17].
TGP method is a valid alternative to classical GP method, because of more flexibility and it permits to treat non-stationary data. The input space is divided through treed binary split and in each region GP is applied with simple covariance function. The split are recursive and the crack should be located where the splits occur also when the noise is present. So, in other word, the split point should occurs where the crack is localized in the beam.
Certainly, the processed data are not necessarily nonlinear along their whole range, for which a GP method jumps to a Limiting Linear model where it is unnecessary to recur to TGPs.
The simulations will be presented first of all adopting a single crack, and subsequently two cracks.
The treed GP is implemented in R by means of the package called "tgp", with a very intuitive and easy interface proposed by Gramacy.
A practical limitation of the method is the large amount of time needed for the computations when large dataset are analyzed.
In order to achieve better results, some arguments of treed GP function coded in R are optimized through Genetic Algorithm (GA).
In the first section a background of GP, treed GP and Genetic Algorithm will be treated .
In the second section the implementation of the TGP, by means of software, will be presented.
In the third section the results will be analyzed in the event that the measured input data are 100 or 60; first of all considering the dataset noise free and then adding different level of noise.
In the last section the conclusion of the work will be dealt with.

1.1 Other approaches to detect the crack

Modal- based approaches are the oldest method used for crack detection. Other approaches have been developed during the years for this purpose such as:

- Acoustic emission that is an efficient method for crack detection less for damage assessment [2];

- Thermal imagining, even if it detects only the surface damage, it is expensive- when thermografy use is not possible- due to the fact that external stimulus source is needed;

- Local inspection approaches as: X-radioscopy, Eddy current, ultrasonic investigations and so on. Their main criticism is that they can be applied only for local detection.

So, although new techniques are implemented, nowadays the modal- based continues to be the most known and used method.

Chapter 2

Bayesian Treed Gaussian process

2.1 Background and motivation

Treed Gaussian process involves partitions of the input space in regions which have features that are homogeneous as possible.
As announced in the introduction, the purpose of the method is the detection of damage in a beam. The split point, where partitions occurs, represents the switch point in which the characteristics of the region are not homogeneous anymore, in other words the split point localizes the crack, because in that point the characteristics change due to the presence of the damage.
The tree splits have to be the simplest one, the most important quality that they must gain is a good level of "accurancy"[14].
The variable and the position of the partition are chosen to minimize the impurity of the node.
The Trees are generated through a partition that is binary and recursive. Binary, since a parent node could always be split in two child nodes, and the binary split does not involve loss of generality of the variable because the sum of the binary split variable is equal to the variable itself before that partitions occurred.
Recursive means that each child node could become a parent node, if it is not a final node.
At each leaf different GPs are applied. One of the first study about partitions tree and the concept of CART (Classification and Regression Tree) model are introduced by L. Breiman [11]
A classification and regression tree (CART [4]) models are flexible method for specifying the conditional distribution of a variable Y, given a vector of predictor value x.[4]

At each split, a Gaussian process regression is applied. For this purpose to be much clearer possible it is appropriate to deal briefly with the Gaussian method, since it is applied at each leaf of the tree division.

2.2 Gaussian Process

The training set, $D = \{(x_i, y,i)|i = 1, ..., n\}$, is composed from all the observations. An observation is defined as the pair of a vector of input data x, of D dimension, and the corresponding scalar output y. In a simpler way the training set becomes D=(x,y).

A Gaussian process is a distribution over functions, indeed it does not return a single value but a probability distribution, so for this reason in the past it acquired much importance in the machine learning field. An example of Gaussian process regression in shown in the figure 2.1. The five crosses represent the observations, instead the green, red and blue lines are free random functions obtained from the posterior, e.g. the prior on the five observations. The gray area represents pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region) [3].

According to the Bayesian paradigm a priori knowledge of the parameters, which ruled the model, are defined and at the beginning of the process predictive, fit on the training data distribution, are used to give prior beliefs about posterior distributions.

To overcome the problem of the reduced applicability of Bayesian model, this one is not applied on the input directly, but it more appropriated the use of a particular space , called features space (with N- dimension ad N>D). The input data are mapped through specific functions, and often this operation occurs without any parameters dependence. In this case for the mapping, a function $\phi = (1, x^T)$ can be defined in which x is the input vector and w is a parameters vector (weight) of the linear model.

The predictive distribution of the output, $f(\mathbf{x})$, is a multivariate normal distribution in which *mean* and *variance* as defined as follows:

$$f(\mathbf{x}) = GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.1}$$

$$m(\mathbf{x}) = E[f(\mathbf{x})] = \phi(\mathbf{x})^T \boldsymbol{\beta} \tag{2.2}$$

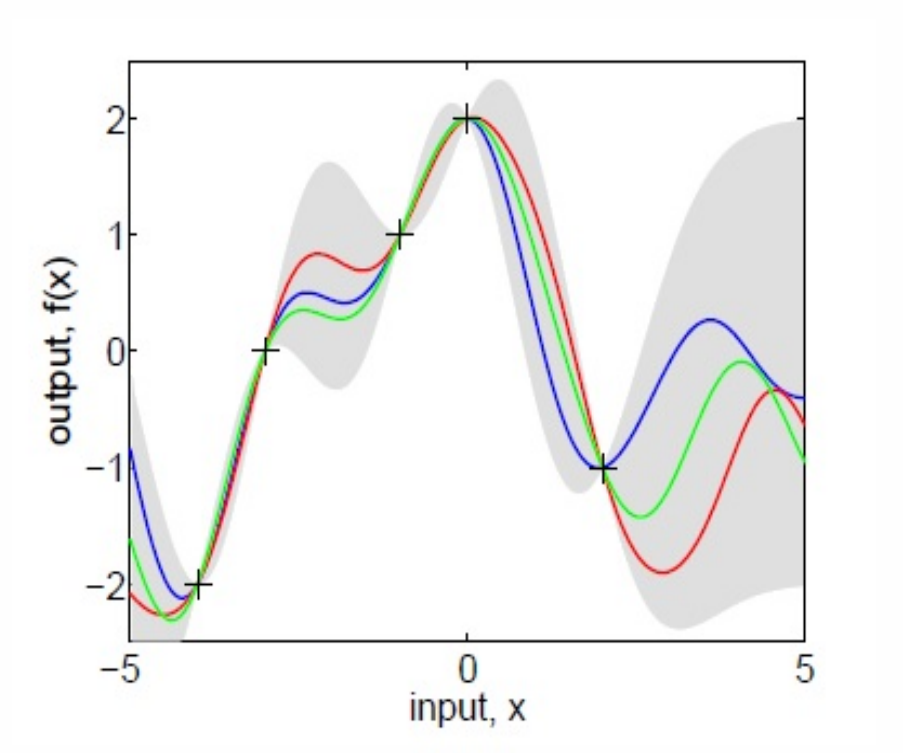

Figure 2.1: Example Gaussian process regression. Imagine from: *Gaussian Processes for Machine Learning* C. E. Rasmussen [3]

$$k(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2.3}$$

The mean represents a least-squares regression fitted through the training data. ϕ^T is a regression function of $\mathbf{x}$, and $\boldsymbol{\beta}$ are regression coefficients [12].
The Gaussian covariance is one of the most important feature of the process. It can be described by means of different functions, two of the most used are:

1. **Exponential covariance function**

$$k(\mathbf{x}, \mathbf{x}') = cov\{f(\mathbf{x}), f(\mathbf{x}')|\sigma^2, B\} = \sigma^2 \exp\{-(\mathbf{x} - \mathbf{x}')^T B(\mathbf{x} - \mathbf{x}')\} \tag{2.4}$$

B is a matrix which constitutes the roughness of the output, instead σ^2 is a height parameter. This kind of covariance function is infinitely differentiable and the smoothness of the GP process is a consequence of that. In addition of a covariance function, a good choice of σ^2 and B is fundamental. These parameters are contained in a vector called θ and together with w they are called hyperparameters.

2. **Polynomial covariance function**

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2\{(1 + \mathbf{x}\mathbf{x}')^T\} \tag{2.5}$$

Also in this case the parameters σ^2, N controls the amplitude and the roughness of the function.

In the first phase, the process would be considered without adding noise.

Bayesian method

As specified, the Bayesian approach implies that a *prior* distribution of the parameters is established to define the model.
This distribution is defined as a Gaussian distribution with null mean and covariance matrix Σ_p: $w \approx \mathrm{N}(0, \Sigma_p)$.
For w and σ^2 an improper prior is assigned, called also weak prior, since few information are known about them [1]. As concerns B hyperparameters, the marginal likelihood does not have a solution in closed form; it is assumed known, but it can be estimated by MAP maximum a posterior method [20].
The marginal likelihood is defined as the likelihood times the prior:

$$p(\mathbf{y}|x) = \int p(\mathbf{y}|\mathbf{f}, x)p(\mathbf{f}|x)\, df \tag{2.6}$$

The posterior distribution that is of the same family of the prior and likelihood is called conjugate.
The prior distributions are very useful if they are conditioned on input data -considered noise free in a first phase- in order to generate the posterior distribution. Indeed, for this purpose it is highly relevant that prior distributions are in line with input data.
The predictive output function f^* and the observed output value f give [3]:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) & K(X,x^*) \\ K(x^*,X) & K(x^*,x^*) \end{bmatrix}\right) \tag{2.7}$$

$K(X, x^*)$ is the covariance matrix of input X and predictive values x^* and the other matrices are defined in the same way.

The noise-free predictive distribution is:

$$\begin{aligned}\mathbf{f}^*|x^*, X, \mathbf{f} \sim N(K(x^*, X)K(X, X)^{-1}\mathbf{f}, \\ K(x^*, x^*)K(x^*, X)K(X, X)^{-1}K(X, x^*))\end{aligned} \tag{2.8}$$

The focus is not f, so it can be marginalized out by means of 2.6.
The noise free case is not realistic so the Gaussian noise can be added. It is summed to the covariance function which assumes the form [3]:

$$k(\mathbf{y}, \mathbf{y}') = k(\mathbf{x}, \mathbf{x}') + \sigma^2\delta_{i,j} \tag{2.9}$$

$$k(\mathbf{y}, \mathbf{y}') = \sigma^2 \exp\{-(\ \mathbf{x\text{-}x'})^T B(\ \mathbf{x\text{-}x'})\} + \sigma_n^2\delta_{i,j} \tag{2.10}$$

$\delta_{i,j}$ is the Kronecker delta, non-zero only if $i = j$, instead, σ_n is the noise variance. The term $\sigma_n^2, \delta_{i,j}$ is also summed in the polynomial covariance function to take into account the noise.
With reference to the joint distribution of the observed training data, y, the covariance can be written as [3]:

$$k(\mathbf{y}) = K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I \tag{2.11}$$

The training outputs y is defined as a Gaussian distribution so they are related to f values:

$$\mathbf{y} \sim N(\mathbf{f}, \sigma_n^2 I) \tag{2.12}$$

After this assumption (relation 2.12), in the equation 2.7 the noise is added:

$$\begin{bmatrix}\mathbf{y} \\ \mathbf{f}^*\end{bmatrix} \sim N\left(0, \begin{bmatrix}K(X, X) + \sigma_n^2 I & K(X, x^*) \\ K(x^*, X) & K(x^*, x^*)\end{bmatrix}\right) \tag{2.13}$$

Our goal is not the prediction of $p(y|f^*)$, but it is the prediction of the conditional distribution of $p(f^*|y)$, using Gaussian properties. As the noise free case, the mean and variance of a predictive distribution for a Gaussian process are [3]:

$$m(\mathbf{f}^*) = K(x^*, X)[K(X, X) + \sigma_n^2 I]^{-1} y \tag{2.14}$$

$$cov(\mathbf{f}^*) = K(x^*, x^*) - K(x^*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, x^*) \tag{2.15}$$

The mean represents a "best estimate", and the variance determines the uncertainty of the respective Gaussian regression model[12].
From observations of the data the hyperparameters has to be chosen in optimal manner. The best strategy in order to optimize the hyperparameters is to maximize the logarithmic form of the marginal likelihood 2.6 using a method based on the gradient:

$$\log p(\mathbf{y}|\mathbf{x}) = \frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log(2\pi) \tag{2.16}$$

2.3 Treed Gaussian Process

CART model is able to divide the input space in homogeneous regions in which stationary Gaussian processes with linear trend can be applied. After the last subdivision, the input space is split in a number of non-overlapping regions, until the terminal node is reached [14].
The model is specified through a set of θ parameters for each tree structure T.
By means of the Bayesian approach it is possible to define the prior probability distribution $p(\theta, T)$ through the relation:

$$p(\boldsymbol{\theta}, T) = p(\boldsymbol{\theta}|T)p(T) \tag{2.17}$$

In this way $p(T)$ can be analyzed separately, without any dependence from θ.
According to Chipman [4], and following the Bayesian approach, a tree prior $p(T)$ at the leaves is specified by means of a tree generating process in order to condition the size of the tree and to define the splitting rules.
The process begins with a null tree, the tree grows following a rules of the type: $x_i > s_i$, if the split respects that rule the tree forks on the right side, otherwise to the left one. All the data are in the same region before tree process starts; each internal point, in which a tree forks, is linked to two splitting probabilities denominated respectively: $p_{RULE}(\rho|\eta, T)$ and $p_{SPLIT}(\eta, T)$.
$p_{SPLIT}(\eta, T)$ is the probability that a terminal node "η" is split, instead,
$p_{RULE}(\rho|\eta, T)$ is defined as the probability of assigning splitting rule "ρ"to "η" if it is split. Both the probabilities are assigned for an intermediate tree T.

$p_{SPLIT}(\eta, T)$ is expressed through the equation:

$$p_{SPLIT}(\eta, T) = \alpha(1 + d_\eta)^{-\beta} \tag{2.18}$$

d_η is the depth of the node η, instead α and β are parameters which govern the size and the shape of the tree. The values assumed from α and β are discussed later. An example of tree partitions is showed in figure 2.2.

The discontinuity is held in the boundary regions where the tree splits in two

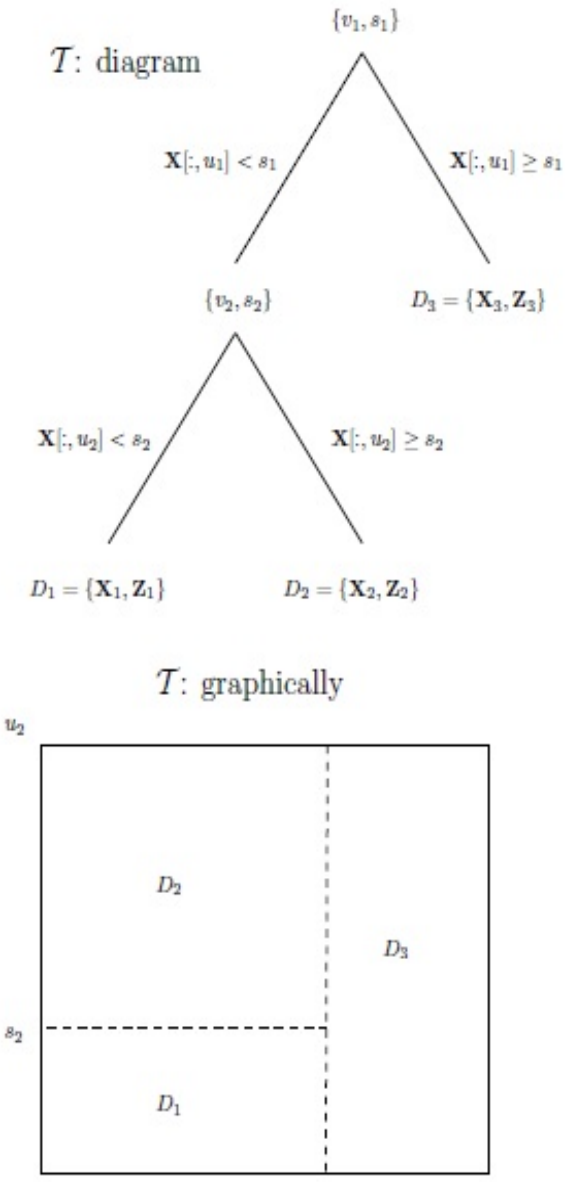

Figure 2.2: Example of partitions of tree in three regions and two splits from "*Bayesian Treed Gaussian Process Models With an Application to Computer Modeling*" by R. Gramacy [14].

different branches.

R. Gramacy [14], suggests a hierarchical generative GP model as a prior specifications for the new adding hyperparameters,unlike the GP.

Let $D = \{\mathbf{X}_v, \mathbf{y}_V\}$ be the data vector for each non overlapping region with n_v observations, the hierarchical generative model is define as [14]:

$$\begin{aligned} \mathbf{y}_v|\boldsymbol{\beta}_v, \sigma_v^2, K_v &\sim N_{nv}(\phi_v\boldsymbol{\beta}_v, \sigma_v^2, K_v), \\ \boldsymbol{\beta}_0 &\sim N_m(\mu, B), \\ \boldsymbol{\beta}_v|\sigma_v^2, \tau_v^2, W, \boldsymbol{\beta}_0 &\sim N_m(\boldsymbol{\beta}_0, \sigma_v^2\tau_v^2 W), \\ \tau_v^2 &\sim IG(\alpha_\tau/2, q_\tau/2), \\ \sigma_v^2 &\sim IG(\alpha_\sigma/2, q_\sigma/2), \\ W^{-1} &\sim W((\rho V)^{-1}, \rho) \end{aligned} \tag{2.19}$$

W and IG are the Wishart and inverse-gamma distribution respectively. W is a $m \times m$ matrix, in which m is the number of covariates in the input data plus an intercept($m = mx + 1$); the hyperparameters μ, B, V, ρ, α_σ, q_σ, α_τ and q_τ are treated as known. K_v is the correlation matrix of GP model which describes a multivariate normal likelihood with linear trend. $\boldsymbol{\beta}$ are regression coefficients. The covariance matrix of a Gaussian regression process can be written as the sum of a correlation matrix, corresponding to the exponential power family as 2.4, and the term "$\mathbf{g}\boldsymbol{\delta}$".

The term g, called *nugget*, takes into account the noise case and it does not allow that covariance matrix to be singular [14]:

$$k(x_i, x_j|\mathbf{g}) = k * (x_i, x_j|g) + g\delta_{i,j} \tag{2.20}$$

which δ_{ij} is the Kronecker delta function and $k*$ is called correlation function, that according with Gramacy [14], it changes slightly its form compared to the exponential one 2.4 expressed in the Gaussian process section:

$$k^*(\mathbf{x}_j, \mathbf{x}_k|\mathbf{d}) = \exp\left\{\sum_{i=1}^{m_X} \frac{|x_{ij} - x_{ik}|^{p_0}}{d_i}\right\} \tag{2.21}$$

$\mathbf{d}_i$ is a range parameter defined positive and assumes different value in each splitting region, this implies that the correlation function is stationary but not isotropic. The correlation between two points of the region is an Euclidean dis-

tance as the term $|x_{ij} - x_{ik}|$ shows. Instead, p_0, which rules smoothness of the process, can assume a value in a range between zero and two.
With the present model one is not sure that along the boundary the process is continuous.
The parameters θ are defined as:

$$\boldsymbol{\theta} = \boldsymbol{\theta}_0 \cup U_{v=1}^{R} \boldsymbol{\theta}_v \tag{2.22}$$

in which $\boldsymbol{\theta}_0$ represents the parameters: $W, \boldsymbol{\beta}_0, \gamma$. They are generated from the hierarchical model at the fist step, instead, $\boldsymbol{\theta}_v$ includes: $\boldsymbol{\beta}, \sigma^2, K, \tau^2$.
It is not easy to establish a distribution over a tree model and priors are not able to find a good posterior inference, so in this case it is useful proceeding by means of Markov chain Monte Carlo (MCMC).
MCMC is the instrument used to investigate the posterior distribution in order to obtain $\boldsymbol{\theta}_v$ conditioned on the hierarchical prior parameters $\boldsymbol{\theta}_0$, once drawn $\boldsymbol{\theta}_v$, the next step is to draw $\boldsymbol{\theta}_0$ from $\boldsymbol{\theta}_v$ and so on.
Parameters space, excluding the covariance parameters, are sampled through Gibbs steps. The roughness parameters, contained inside the covariance matrix, need Metropolis-Hasting(MH) ratio.
The main idea of Markov Chain Monte Carlo (MCMC) is to establish a Markov chain whose stationary distribution is the posterior distribution of interest, and then collect samples from that chain. The transition probabilities from state α_n to α_{n+1} of the Markov chain, representing samples from the posterior of $\boldsymbol{\theta}$, can be set up in two ways: using the Metropolis-Hastings (MH) algorithms or Gibbs sampling [8].
Metropolis-Hasting algorithm is based on an **acceptance ratio**. In order to simulate the passage between two samples, e.g. α_n and the subsequent one α_{n+1}, a proposal density $Q(\alpha_n, \alpha*)$ is created, in which $\alpha*$ is the artificial sample used to pass from α_n to α_{n+1}. Fixing $\alpha_{n+1} = \alpha*$ one can calculate the acceptance ratio (R):

$$R(\alpha_n, \alpha^*) = \min \left\{ \frac{Q(\alpha_n, \alpha*_n)}{Q(\alpha^*, \alpha_n)} \frac{P(Y|\alpha^*)P(\alpha^*)}{P(Y|\alpha_n)P(\alpha_n)}, 1 \right\} \tag{2.23}$$

$\alpha_{n+1} = \alpha*$ is accepted or rejected depend on this condition:

$$\alpha_{n+1} = \begin{cases} \alpha * \ with\ prob.\ R \\ \alpha_n\ with\ prob.\ 1 - R \end{cases} \tag{2.24}$$

Following this method the most important advantage is that the norming constant which should be calculated using Bayesian approach is not needed in this case, avoiding the calculation of integral not resolvable in closed form.
Regards Gibbs sampling it is a special case of MH, in which the acceptance ratio R is equal to one; this implies that $Q(\alpha*, \alpha_n) = p(\alpha * |Y)$ and the proposal are always accepted. Parameters with conditionally conjugate priors can usually be sampled with Gibbs steps. [8]
These procedures are required to draw the parameters θ_v, first of all, for a single leaf, then for all the leaf, defining all the parameters of that tree $(\theta|T)$.
After generating a tree it is needed to move toward another space of parameters and towards a new tree. Jump Markov chain Monte Carlo (RJMCMC) is the means used to integrated out tree dependence, it is an extension of MH algorithm. This method is used to take into account that the parameters space changes moving on different trees by means of the Jacobian term. The MH ratio is analogous both MCMC and RJMCMC if the Jacobian term is equal to one; this happens that occurs when the proposals are taken from the prior. The obtained posterior distributions are used for the future estimations of the model.
R. Gramacy [14] follows Chipman [4] method in order to sample the joint posterior of $(\theta, T\ )$, drawing $\theta|T$ and then $T|\theta$. The method is slightly different, indeed in addition to Chipman [4] operations *change, swap, grow* and *prune*, R. Gramacy [14] suggests a *rotate* operation described as [4]:

- GROW: Randomly pick a terminal node which is split in into two child nodes and use the p_{rule} defined in the prior.
- PRUNE: Randomly pick a parent of two terminal nodes and turn it into a terminal node by collapsing the node below it.
- CHANGE: Randomly pick an internal node, and randomly reassign it a splitting rule according to p_{rule} used in the prior.
- SWAP: Randomly pick a parent child pair that are both internal nodes.

Swap their splitting rules unless the other child has the identical rule, in which case swap the splitting rule of the parent with that of both children.

- ROTATE: rotation of leaf which can be a right or left rotation. An example of a good right rotation is shown in 2.3 [14].

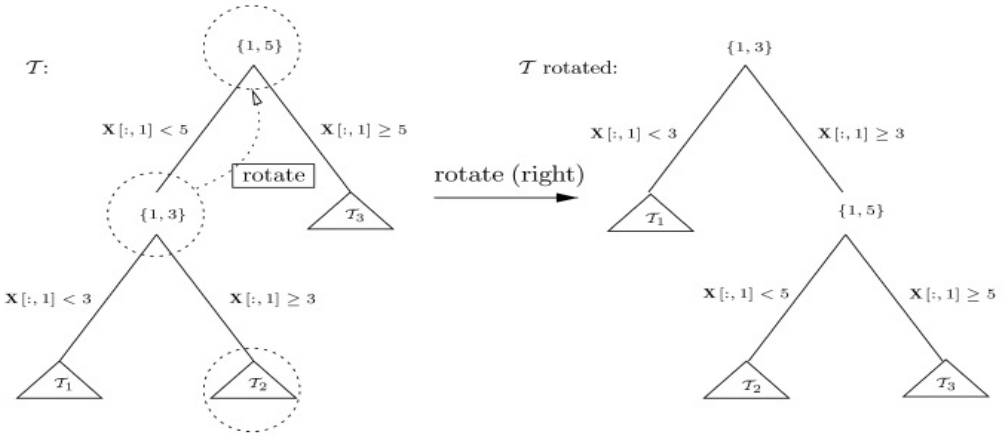

Figure 2.3: Rotation on the same variable, T_1, T_2 and T_3 as shown in Gramacy paper [14].

Since *grow* and *prune* are more complex operations because they include the addition or removal of a part of region. Indeed, the addition provides the definition of new parameters, instead, the removal provides that the parameters are absorbed from the parents or discarded. The new parameters related to the covariance are proposed from the prior therefore the Jacobian term is unitary. *Grow* and *prune* are reversible counterparts.
As regards *swap* operation can involve an empty child node, so it can be substituted with *rotate* operation, expunging the problem that could occur with *swap* operation, moreover rotation is useful for a better mixing of the Markov chain by providing a more dynamic set of candidate nodes for pruning, thereby helping it escape local minima in the marginal posterior of T [14].

2.3.1 Kriging

The Kriging is a prediction of outputs y, known with this name in geostatistic field, which is conditioned on the covariance structure.

The discontinuity of the posterior predictive surface is held in the boundary, even if the discontinuity is mitigated with the operations *grow*, *prune*, *swap* and *change*. Also the aggregate mean tend to smooth out the discontinuity along the boundary partitions, however this increases the uncertain uncertainty in the posterior distribution for the research of T. The predictive output y is normally distributed with mean and variance [14]:

$$\begin{aligned}\hat{y}(\mathbf{x}) = E(\mathbf{y}(x)|data, \mathbf{x} \in D_v) = \boldsymbol{\phi}^T(\mathbf{x})\tilde{\boldsymbol{\beta}}_v - \\ - \mathbf{K}_v(x)^T\mathbf{K}_v^{-1}(\mathbf{Z}_v - \mathbf{F}_v\tilde{\boldsymbol{\beta}}_{\boldsymbol{v}})\end{aligned} \tag{2.25}$$

$$\begin{aligned}\hat{\sigma}(\mathbf{x})^2 = var(\mathbf{y}(\mathbf{x})|data, \mathbf{x} \in D_v) = \\ = \sigma_v^2[\kappa(\mathbf{x}, \mathbf{x}) - \mathbf{q}_v^T(\mathbf{x})\mathbf{C}_v^{-1}\mathbf{q}_v(\mathbf{x})]\end{aligned} \tag{2.26}$$

with:

- $\boldsymbol{\phi}^T = (1, \mathbf{x}^T)$, $\mathbf{k}_v(\mathbf{x})$ a n_v vector;
- $\mathbf{k}_{v,j}(\mathbf{x}) =$
  $= \mathbf{K}_v(\mathbf{x}, \mathbf{x}_j)$ for all $\mathbf{x}_j \in \mathbf{X}_v$;
- $\mathbf{C}^{-1} = (\mathbf{K} + \phi\mathbf{W}\phi^T/\tau^2)^{-1}$;
- $\mathbf{q}(x) = \mathbf{k}(x) + \tau^2\mathbf{F}\mathbf{W}\phi(y)$;
- $\kappa(x, y) = K(x, y) + \tau^2\phi^T\mathbf{W}\ \phi(y)$.

2.4 Gaussian Processes and Limiting Linear Models

The Gaussian process and a fortiori Treed Gaussian process could be not needed when part of the input space is linear. According to R. Gramacy [14], one may think that the model "*jumps*" from GP to Linear model. Indeed, the linear model can be considered as a special limit case of GP, with some benefits: first of all the reduction of computation time and furthermore computation efficiency. After this considerations the first relation of 2.19 equations, which describes the prior in GP model changes in order to adapt itself at the linear model:

$$\mathbf{y}_v|\boldsymbol{\beta}_v, \sigma_v^2 \sim N_{nv}(\phi_v\boldsymbol{\beta}_v, \sigma_v^2, I), \tag{2.27}$$

One can be noticed that now the identical matrix I- with dimension nxn- is used in order to parameterize the linear model. The transition between GP model and the linear model can be occurred in different ways:

1. Fixing the range parameter d null in order to obtain: $\mathbf{K} = (1 + g)\mathbf{I}$;

2. Cressie [6] focused his attention on the nugget parameters g, and its interaction with d. He concluded that a linear model can be obtained fixing a non zero d and a finite g.

The jump between the two models can be carried out using boolean operator b_i, that is equal to one when GP model must to be applied. The parameters of the correlation function are multiplied for the boolean operator, which increases the parameter space of an m_X indicators. According with Cressie [6], the boolean operator entails, for these values of g and d, a non linear GP model; but when the prior assumed the form of the equation 2.27, the process jumps towards a linear model.

2.4.1 Kriging for Limiting Linear Model

The prediction in the previous section are referred to equations 2.25 and 2.26. In the linear model some terms are canceled as $\kappa(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{x})$, leading to a simplified mean and variance which describe, also in this case, the normally distributed output y. The simplified equations are:

$$\hat{y}(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\tilde{\boldsymbol{\beta}}_v \tag{2.28}$$

$$\begin{aligned}\hat{\sigma}(\mathbf{x})^2 = \sigma_v^2[1 + \tau^2\boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}) - \\ \tau^2\boldsymbol{\phi}^T(\mathbf{x})\mathbf{F}^T((1+g)\mathbf{I} + \tau^2\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\mathbf{f}(x)\tau^2]\end{aligned} \tag{2.29}$$

Re-writing the equation 2.29, using the Woodbury formula, and setting $\mathbf{V}_{\hat{\beta}} = (\tau_{-2} + \mathbf{F}^T F(1 + g))^{-1}$ and $\mathbf{W} = \mathbf{I}$, one can be obtained:

$$\hat{\sigma}(\mathbf{x})^2 = \sigma_v^2\big[1 + \mathbf{f}^T(\mathbf{x})\mathbf{V}_{\hat{\beta}}\mathbf{f}(\mathbf{x})\big] \tag{2.30}$$

The equation 2.30, used under linear model condition, has an important benefit because it implies the inversion of $(m_x \times 1)x(m_x \times 1)$ matrix, against the inversion of $n \times n$ matrix.

2.5 Optimization methods- GA

In order to obtain better results some parameters are optimized using *Genetic algorithm* (GA). In details, the parameters α and β are optimized in the equation 2.18 in order to obtain a good choice of the shape and the size of the trees.
Genetic algorithms are the most frequently type of Metaheuristic evolutionary algorithm, which generate solutions to optimization problems using techniques inspired by natural evolution.
Metaheuristic evolutionary algorithms (EAs) are stochastic methods based on the principle of natural selection to guide the evolution. These algorithms present advantages, such as:

- robustness: they only need information about the objective function;
- works with most functions: discontinuous, multimodal, etc...
- effectiveness: the probability of convergence to a local minimum can be very small.

The algorithm starts with the creation of a generation, represented by chromosomes, called population. Each chromosomes is split in gene. The population size depends on the nature of the problem. Often, the initial population is generated randomly, taking account of the possible solutions in the search space. During each successive generation, a portion of the existing population is selected to generate a new generation. Only the part of satisfying population can mate to create descendants, with chromosomes inherit from both parents. The past experience is crucial in order to extract the best samples.
The fitness function is the function that the algorithm has to optimize and the means by which the quality of a solution is measured, to put it as simply, it is equivalent to an objective function in the other optimization method and it depends on the analyzed problem. Each generations produce solutions and a new generations in relation to how the fitness function is optimized. A fitness function and a probabilities of survival in the next generation are associated at all the population members.
Bearing in mind that, at the beginning of the process the random selection of population is carried out, the fitness function will be a function of very low quality, improving its quality gradually in the successive generations. Solutions, which

are chosen to form new population (offspring), are selected according to the fitness value of the previous generation. The new population should be better than the old one in order to maximize or minimize the optimum function in relation to the optimum of the problem. In order to generate and create the new population the genetic operation as *cross-over*, *mutation* and *inversion* are involved:

- Cross-over: exchange of chromosomes between different character. Since in this field each chromosome is codified by bits, this operation is a random exchange of bit;

- Mutation: part of the genes mutates in relation to known and set coefficients. Reasons of why the mutation is applied can be searched in an improving of fitness function or in a removal of the cases to find a local optimum. A coefficient rules the number of a needed mutation.

- Inversion: it regards exchange chromosomes (bits) which belongs to the same character.

Cross-over is crucial to improve solutions of best characters and the duplication of them; mutation and inversion, instead, allow the diversification of the population. The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), mutation and inversion. This is repeated until the fitness is satisfied. The success of the optimization and the speed of convergence are influenced by the choice of GA parameters. This techniques is very easy to use but the convergence could be very slow compared with the other methods.
The Figure 2.4 summarizes very well the mode of operation of the genetic algorithm.

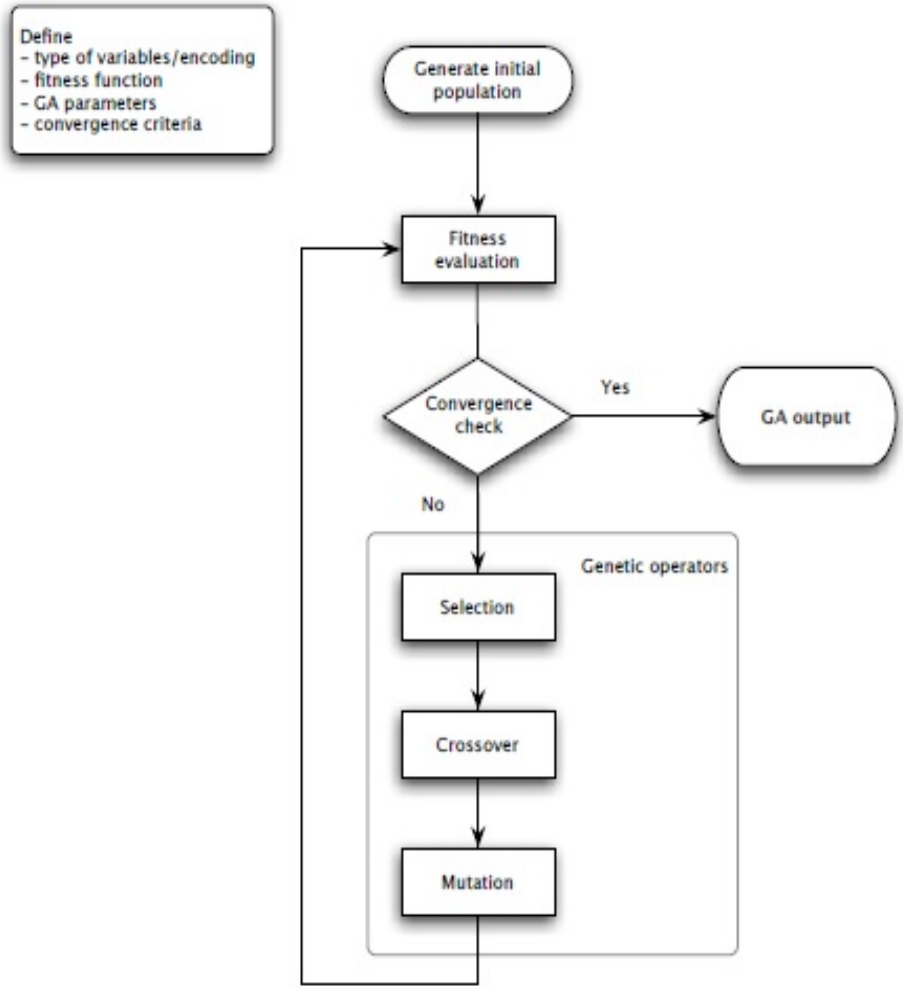

Figure 2.4: Flow-chart of genetic algorithm from "GA: A Package for Genetic Algorithms" by Scrucca [18].

2.6 Identification and localization of the damage

The identification of of the damage in a beam employs the Treed Gaussian process mentioned above.

The method is applied on the measured mode shape, having the sagacity to discriminate the cracked and uncracked area, since the stiffness matrix changes based on these criteria.

Indeed, the effect of the crack in a beam affects the stiffness matrix much more than other elements such as mass matrix.

So unlike the beam where no discontinuity occurs, the cracked beam is defined with stiffness matrix which changes their form based on the position of the discontinuity the whole beam. In th GP model it was needed to define different covariance functions so that the GP can discern between cracked and uncraked area of the beam. The new correlation function was defined as [10]:

$$k'(x_i, x_j) = \begin{cases} k_{poly}(x_i, x_j) + k_{exp}(x_i, x_j) & x_i < x', x_j < x' \\ k_{poly}(x_i, x_j) + k_{exp}(x_i, x_j) & x_i > x', x_j > x' \\ k_{poly}(x_i, x_j) & otherwise \end{cases} \tag{2.31}$$

x' was the point where the crack is localized. The last equation of 2.31 underlined that in the area far from the discontinuity, the applied covariance function was the exponential one, instead, in the area close to the crack the sum of polynomial and exponential covariance function was needed. An optimal procedure in order to establish the position of the discontinuity, x'- which was not define a priori- was to maximize the marginal likelihood of the mode shape data, mentioned in the equation 2.16 for hyperparameters, moving forward the point where the stiffness matrix changed. The presence of a peak in the predictive posterior distribution was a clear manifestation of the discontinuity.

The treed Gaussian process does not need of a switch point in the covariance kernel because, as explained, each region is split in homogeneous part in which different GP are applied. The damage is localized where a branch of the tree occurs.

Chapter 3

Implementation

3.1 Treed Gaussian process

Treed Gaussian process is coded in a package known as "*tgp*", implemented by means of the program R.
TGP method is used to detect the crack by means of the mode shape of the cracked cantilevered beam.
100 or 60 sensors, uniformly distributed along the length of the beam, have been utilized for the measure.
In a first moment only one crack, placed at 0.3m, is analyzed.
In a second moment the noise is added. The analyzed data include signal noise to ratio (SNR) equal to 100, 80, 60.
Signal to noise ratio is defined as the ratio between the desired power and the noise of the system, it can be expressed in decibel as:

$$SNR_{dB} = 10 \log_{10} \frac{P_{signal}}{P_{noise}} \tag{3.1}$$

In a third moment, the case of two cracks is analyzed; the first crack is placed at 0.3m and the second one at 0.6m. In this case of multiple crack, the beam measures are only 100, uniformly distributed along its length.
The "*tgp*" package implements Bayesian nonstationary, semiparametric nonlinear regression with Treed Gaussian process models with jumps to the Limiting Linear Model (LLM).[15]
The functions, dealt with in Gramacy paper [15], have level of complexity more and more higher and they allow the regression processes.

The seven functions collected in the "*tpg*" package are summarized below:

1. *blm*: to code Linear model ;
2. *btlm* to code Treed Linear Model;
3. *bcart*: to code Treed Constant Model;
4. *bgp*: to code Gaussian Process Regression;
5. *bgpllm*: to code Gaussian Process Regression with jumps to the Limiting Linear Model;
6. *btgp*: to code Treed Gaussian process Regression;
7. *btgpllm*: to code Treed GP with jumps to the Limiting Linear Model.

The b before each function represents the usage of the *boolean* operator.
Great significance is given to *tgp class-objects* which gather knowledge of posterior predict distribution, of MAP (Maximum a Posteriori) trees and adaptive samplings. They are obtained as a list of outputs of the seven tgp functions.
In this work the function, which fits better the problem, is the *btgpllm*, so a treed Gaussian Process regression with jumps to the Limiting Linear Model.
The code shows the parameters which governs the function behavior, in this phase the default ones are displayed below. Their usefulness will be described in the next sections as well as the best option or value given at them to fit adequately the analyzed model.

Listing 3.1: *btgpbllm*

```
> btgpllm(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
+ corr = "expsep", tree = c(0.5, 2), gamma=c(10,0.2,0.7),
+ BTE = c(2000, 7000, 2), R = 1, m0r1 = TRUE, linburn = FALSE,
+ itemps = NULL, pred.n = TRUE, krige = TRUE, zcov = FALSE,
+ Ds2x = FALSE, improv = FALSE, sens.p = NULL, nu = 1.5,
+ trace = FALSE, verb = 1, ...)
```

3.2 Description of "*btgpllm*" arguments

The arguments are described in detail and a study of them is needed in order to obtain a better estimation of the output of the model.
Sometimes, the arguments have different values from the default ones. Below the arguments of the function are listed and analyzed. The reasons of why some options are chosen against the default choice is explained.

- **X** contains the input data, it can be a vector, a data frame or a matrix. In this work it is defined as a sequence of value (vector) of length equal to the beam length uniformly divided in step of 100 or 60 points. It depends on in how many points the beam is divided for the measure.

- **Z** is the vector of the output value.

- **XX** is the vector of predict input locations defined as a sequence of value from zero to the beam length subtracting a unit. It is uniformly divided in step of 100 or 60 points, depending on how many points the beam is divided for the measure.

- **meanfn** is the mean function of the model. It is set to "linear" as the default option suggests. An alternative could be to put it equal to "constant" but it is not a good choice in our case.

- **bprior** is the default prior specified in an improper way to be a linear β-defined in 2.19- prior. In this case W = ∞ (2.19), so it no need to specify β_0. This kind of option is a good choice when the signal noise ratio (SNR) is very lower, so high noise is expected. In a first moment where no noise is considered *bprior* is fixed equal to $bflat$(linear prior), when noise increases the hierarchical normal prior, "$b0$", (2.19)is demonstrated that works well.

- **corr** indicates the correlation function chosen, **corr** equal to *"expsep"* corresponds to choose a separable power exponential family. Other correlation functions can be chosen as the isotropic Matern ("*matern*") or single-index Model ("*sim*") but the using of them do not improve the results so they will be not treated.

- **BTE** vector summarize the MCMC parameters in which the first value represent the Burn-in period of MCMC, the second term is Total (time) at

which it is stopped and the last one, Every, represents the predictive values which are saved at each round.

- **linburn**: in order to improve MCMC performance it is possible to initialize it with some additional rounds calibrated by means of rounds of Treed Bayesian Cart model (using the function *btlm*). This implies a pre-splits of the input space with linear model. This process occurs before the Burn-in period. By way of example, here it is showed a piece of MCMC code.

Listing 3.2: MCMC rounds

```
burn in:
**PRUNE** @depth 2: [1,0.151515]
r=1000 d=[0] [0] [0.00941218] [0] [0.417061] [0.00742607];
mh=1 n=(29,11,18,13,13,16)

Sampling @ nn=99 pred locs:
**PRUNE** @depth 2: [1,0.591837]
r=1000 d=[0] [0] [0.0749705] [0] [0.0111204]; mh=1 n=(27,11,32,11,19)
**CPRUNE** @depth 0: var=1, val=0.808081->0.806122, n=(80,20)
r=2000 d=[0.0238184] [0.136605] [0.0964063] [0.0402586];
mh=4 n=(27,11,44,18)
**PRUNE** @depth 2: [1,0.826531]
r=3000 d=[0.0419871] [0.0341552] [0.0665801]; mh=3 n=(27,11,62)
r=4000 d=[0.0172427] [0.0167988] [0.0827976]; mh=3 n=(27,11,62)
r=5000 d=[0.0249484] [0.0140079] [0.0655174]; mh=3 n=(27,11,62)
r=6000 d=[0.0264499] [0.0221825] [0.0742443]; mh=3 n=(27,11,62)
Grow: 1.761%, Prune: 2.049%, Change: 60.56%, Swap: 87.2%
finished repetition 2 of 2
removed 3 leaves from the tree
```

The rounds number of MCMC is indicated with r: the first $r = 1000$ defines the BURN-IN rounds and the second $r = 7000$ the number of total rounds. The d parameter inside the squared brackets indicates the d-range parameter of a separable correlation function, when d=0 the model is linear (LLM).

- **R**: the MCMC uses the MH algorithm in order to sample the input space. As Chipman [4] noticed, the chain moves local for a long time inside a tree region. Indeed, when the tree is found- which means that the algorithm

converges- it is considered more worthwhile to move from one mode to others inside the tree space, so that the entire space tree can be inspected. The best way to overcome this problem is the restarting the MH algorithm, in this way it is drawn toward new modes and the algorithm loses any information about the last mode where it was before the restart. In order to do this, the parameter **R** is set equal to two; it means that only two restart are expected because for each restart a large amount of time is needed for the computation.

- **Ds2x**: DOE is the acronym used to define the Design of experiment. In a statistical field it allows to plan, order or head the training dataset of the experiment in order to define relationships between them and properties which define the system. The main purpose is the achievement of a predict model useful to have an idea about relationships and properties of the system.
  DOE, in the computer simulations area, is also known as Sequential Design and Analysis of Computer Experiments (SDACE).
  Active learning, *query learning* or *selective sampling* are the appellatives given to the design experiment in "Machine Learning" field. The term learning is aimed at revealing the use of learning algorithm in order to act and control the input dataset so that the design of experiment gains its goals.
  There is no a single function in *tgp* package which allows to use this method of samples. A set of arguments must be adapted in the function "*btgpllm*" as compared to the default ones.
  In the Gaussian process two active learning algorithm are fundamental for this purpose:

  1. *Active LearningMacKay*- ALM;
  2. *Active LearningCohn*- ALC.

  The Active LearningMacKay is activated by default and implemented by MCMC in the tgp package. It maximizes the expected information design through the selection of a location- $\hat{x}$- which gets the highest standard deviation in predictive output. [13]
  In details, this algorithm computes the norm (or width) of predictive quan-

tiles obtained by samples from the Normal distribution of prediction given by equations 2.25 and 2.26.[9]

ALC is another algorithm used for the same purpose of the previous one. In order to operationalize the algorithm, the boolean argument **Ds2x** is set to **TRUE**, computing $\Delta\sigma^2(\hat{x})$. $\Delta\sigma^2(\hat{x})$ is the reduction of the variance due to the fact that the vector of locations contains a new element $\hat{x}$, selecting from a set of candidates $\hat{X}$. The ALC algorithm aim is the selecting $\hat{x}$ so that the expected squared error, averaged over the input space, would be the minimum possible; so the predictive variance is globally reduced, even if the reduction is averaged on the all locations $\mathbf{y}$. Usually, the vector $\mathbf{Y}$ coincides with $\hat{\mathbf{X}}$ Let $\hat{\mathbf{x}} \in \hat{\mathbf{X}}$ be the adding location, conditioned on a tree, the variation of variance at a point $\mathbf{y} \in \mathbf{Y}$ is given by [13]:

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\hat{\mathbf{x}}) = \hat{\sigma_{\mathbf{y}}}^2 - \hat{\sigma_{\mathbf{y}}}^2(\hat{\mathbf{x}}) \tag{3.2}$$

$\hat{\sigma}^2_{\mathbf{y}}$ and $\hat{\sigma}^2_{\mathbf{y}}(\hat{\mathbf{x}})$ are defined as:

$$\begin{cases} \hat{\sigma}^2_{\mathbf{y}} = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_N^T(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N^T(\mathbf{y})] \\ \hat{\sigma}^2_{\mathbf{y}}(\hat{\mathbf{x}}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_{N+1}(\mathbf{y})^T\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})] \end{cases} \tag{3.3}$$

Using the partition equation of Barnett, $\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x})$ can be expressed in a more appreciable form:

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\hat{\mathbf{x}}) = \frac{\sigma^2[\mathbf{q}_N^T(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\hat{\mathbf{x}}) - \kappa(\hat{\mathbf{x}},\mathbf{y})]^2}{\kappa(\hat{\mathbf{x}},\hat{\mathbf{x}}) - \mathbf{q}_N^T(\hat{\mathbf{x}})\mathbf{C}_N^{-1}\mathbf{q}_N(\hat{\mathbf{x}})} \tag{3.4}$$

It is also possible to compute the difference in predictive variance when a data location $\hat{\mathbf{x}}$ is added, averaged over $\mathbf{y} \in \mathbf{Y}$ [13] :

$$\Delta\hat{\sigma}^2(\hat{\mathbf{x}}) = |\mathbf{Y}|^{-1}\sum_{\mathbf{y}\in\mathbf{Y}}\Delta\sigma^2_{\mathbf{y}}(\hat{\mathbf{x}}) \tag{3.5}$$

Experimentally one is noticed that ALC algorithm spent more time to select the adding value $\hat{x}$, for this reason it is not a default option, it is active only if the experiments allows it without computations increase a lot.

Although both ALM and ALC have the same purpose to order and rank the input predictive location vector, in order to improve the posterior distribu-

tion the first algorithm samples much more in central region of partitions than the boundary one.

As described in chapter 2 when the Limiting Linear model fits adequately the model the equations are simplified because the correlation function becomes: $\mathbf{K} = (1 + g)\mathbf{I}$. Under this conditions the reduction in variance $\Delta\sigma_y^2(x)$ has the following form [9]:

$$\Delta\hat{\sigma}_{\mathbf{y}}^2(\hat{\mathbf{x}}) = \frac{\sigma^2[\boldsymbol{\phi}^T(\mathbf{y})\mathbf{V}_{\beta_N}\boldsymbol{\phi}(\hat{\mathbf{x}})]^2}{1 + g + \boldsymbol{\phi}^T(\hat{\mathbf{x}})\mathbf{V}_{\beta_N}\phi(\hat{\mathbf{x}})} \tag{3.6}$$

When $\mathbf{y}$ and $\hat{\mathbf{x}}$ are in different partition area the difference of variance is equal to zero ($\Delta\sigma_x^2(\hat{\mathbf{x}})$). Since the **Ds2x** is set to **TRUE** in this work the ALC algorithm is used during the simulations.

- **trace** is set to to **FALSE**; when it is **TRUE** it saves all the parameters of the model in the vector **XX**, according with the location values, for all the simulations.In this case due to the huge amount of data this option could gravely reduce the speed of the computations.

- **pred.n**: an important rule in the *tgp* package is played by the function "predict". It returns (if it is TRUE) prediction at the inputs X.

- **IMPROV**: by default this option is not operating because it involves a large amount of time needed for the computations.
  The improvement, as the name suggests, is reached employing a criterion in which the search locations $\hat{\mathbf{x}}$ are sorted. This criterion is based on sampling the posterior predictive distribution of Gaussian Process regression. The focus is the search of the maximum improvement in the future distribution. The minimum between all the input locations is chosen as the value which implies the higher improvement.
  As mentioned before the arguments **IMPROV** could be an integer, in such case the optimization criterion is applied to a chosen m number of locations, whereas when it is set on **TRUE**, as in this case, the criterion is applied to the entire vector XX. Usually is almost impossible detected all locations. The process is recursive.
  Let $I(x) = max f_{min} - f(x), 0$ be the improvement of one location and

f_{min} is the minimum response in the search, the "*tgp*" generates search pattern which consists of m locations that recursively maximize (over a discrete candidate set) a sequential version of the expected multi-location improvement defined as: $E[I^g(x1; ....; xm)]$. I^g is equal to [16]:

$$I^g(x1; ....; xm)] = (\max(f_{min} - f(x_1)), ..., (f_{min} - f(x_m)), 0)^g \qquad (3.7)$$

The equation 3.7 estimates the maximum expectations for the posterior distribution, even if an estimation for the full distribution is impossible to achieve.

The output of this optimization procedure is a vector XX in which the predictive locations are sorted in decreasing order of improvement.

This approach reaches the optimum after few iterations that increase when the number of MCMC rounds are higher, as our case.

- **krige**: by default is equal to TRUE. Its purpose is the collection of kriging means and variances at predictive locations.

- The arguments **sens.p**, **nu**, **zcov** and **verb** are set equal to the default value or option. In other words, they are not involved in this study because:
  sens.p is referred to sensitivity analysis not dealt with;
  nu fixes parameters for Matrn correlation function that it is not involved in these simulations;
  zcov, when it is active it compute the predictive covariance matrix, but being computational expensive it has been avoided. By default only the variance is calculated;
  verb indicates the level of verbosity related to MCMC passages. Depends on its value MCMC algorithm shows progress meter on the console of the program:(*grow*, *prune*, *change* and *swap*). By default is fixed equal to two, so only *grown* and *prune* are displayed.

- **tree** represents a vector which contains the prior parameterization of α and β discussed in equation 2.18. α and β are defined in an interval of $[0, 1]$ and $[0, 2]$ respectively, but the chosen default values are 0.5 for α and 2 for β. It can be noticed that varying manually α and β there was an improvement of expected results. So, in order to improve the model the *Genetic algorithm*

is applied to find the best value of α and β. The implementation of the algorithm is discussed in the next section.

In order to have a look, in details, of the package "tgp" the reading of tpg guide is recommended [15].

3.2.1 Genetic algorithm code

The **R** program is provided with a GA package with a very easy interface for the users. The default GA code is:

Listing 3.3: *GA* package

```
> ga(type = c("binary", "real−valued", "permutation"),
+ fitness, ...,
+ min, max, nBits,
+ population = gaControl(type)$population,
+ selection = gaControl(type)$selection,
+ crossover = gaControl(type)$crossover,
+ mutation = gaControl(type)$mutation,
+ popSize = 50,
+ pcrossover = 0.8,
+ pmutation = 0.1,
+ elitism = base::max(1, round(popSize∗0.05)),
+ updatePop = FALSE,
+ postFitness = NULL,
+ maxiter = 100,
+ run = maxiter,
+ maxFitness = Inf,
+ names = NULL,
+ suggestions = NULL,
+ optim = FALSE,
+ optimArgs = list(method = "L−BFGS−B",
+ poptim = 0.05,
+ pressel = 0.5,
+ control = list(fnscale = −1, maxit = 100)),
+ keepBest = FALSE,
+ parallel = FALSE,
+ monitor = if(interactive())
+ { if(is.RStudio()) gaMonitor else gaMonitor2 }
+ else FALSE,
+ seed = NULL)
```

In order to adequate the code at the analyzed case, the vector *type*, that collects the decision variables, is set to *real-valued* since real numbers are object of the study.

The arguments *gaControl(type)* used for the population, selection, crossover and mutation are a default option. In details, this function generates a random population of the specified type (real values in our case), a fitness proportional selection, a local arithmetic crossover and a uniform random mutation. [18]

The fitness function represents the function to optimize. It has been assumed as a difference between the position of the crack for a (known) measured position and for a particular value of α and β. Optimizing this function means that the difference is close to zero, in other word optimization occurs when the position of crack coincides in the two case.

The *min* and *max* represent the minimum and the maximum of the range for the variable α and β .

Only this described parameters has been changed compared to the default ones. The comprehension of the other arguments is not object of the study, for more details [19] is recommend.

Chapter 4

Data and results

4.1 Case study

The TGPs performance are evaluated applying the method to a cantilever beam with the following geometric e mechanical characteristics:

Youngs Modulus	density	length	squared cross-section
$2.06 \times 10^{11} N/m^2$	7850 kg/m^3	14m	0.02 m

Table 4.1: Geometrical and mechanical characteristics

As described in 2, the stiffness matrix of the cracked element feels the effect of the presence of crack. It is computed through the principle of linear elastic fracture mechanism. The mode shape, that is employed in the TGP method, was drawn numerically by means of the finite elements model.
The degree of freedom of uncracked beam, modeled with two nodes, are two at each node: the transverse displacement and the rotation. The beam is analyzed by means of Euler-type finite elements.
For the cracked beam, the employing method is that suggested by G.L.QianS. and S.Jiang [7].
Mode shapes have been calculated numerically using the finite element model.
The crack depth is equal to 0.4 m and it is supposed that it does not spread with the time.
The TGP model was run 100 times for each mode due to the fact that it is statistical method. At the end of each run the split points were localized and the MAP (maximum a posteriori) tree was chosen and printed. The MCMC algorithm, used for the simulations, in addition to the first initialization, by means of

LINBURN argument, it is initialized with a BURN-IN period of 1000 rounds followed by a 7000 total rounds.
The data are presented, first of all, in the event that the measured points are 100 points and then 60.
The first case studies includes that the crack is located at 0.3m from the the fixed end of the beam, then the multiple crack case is simulated; the cracks are placed at 0.3 and 0.6 m from the fixed support.

4.2 Results

The results are summarized in the tables: 4.2, 4.3, 4.4, 4.6 and 4.7.

100 points SNR=∞								
MODE 1			MODE 2			MODE 3		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{100}{100}$	$\frac{51}{100}$	$\frac{16}{100}$	$\frac{94}{100}$	$\frac{86}{100}$	$\frac{0}{100}$	$\frac{80}{100}$	$\frac{96}{100}$	$\frac{0}{100}$
MODE 4			MODE 5			MODE 6		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{67}{100}$	$\frac{89}{100}$	$\frac{3}{100}$	$\frac{53}{100}$	$\frac{66}{100}$	$\frac{74}{100}$	$\frac{77}{100}$	$\frac{91}{100}$	$\frac{12}{100}$
MODE 7			MODE 8			MODE 9		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{74}{100}$	$\frac{73}{100}$	$\frac{37}{100}$	$\frac{20}{100}$	$\frac{45}{100}$	$\frac{51}{100}$	$\frac{43}{100}$	$\frac{52}{100}$	$\frac{74}{100}$
MODE 10			MODE 11			MODE 12		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{25}{100}$	$\frac{55}{100}$	$\frac{76}{100}$	$\frac{17}{100}$	$\frac{75}{100}$	$\frac{75}{100}$	$\frac{14}{100}$	$\frac{10}{100}$	$\frac{56}{100}$
MODE 13			MODE 14			MODE 15		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{11}{100}$	$\frac{37}{100}$	$\frac{99}{100}$	$\frac{3}{100}$	$\frac{17}{100}$	$\frac{96}{100}$	$\frac{0}{100}$	$\frac{8}{100}$	$\frac{0}{100}$

Table 4.2: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 100 without noise.

100 points SNR=100								
MODO 1			MODO 2			MODO 3		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{45}{100}$	$\frac{31}{100}$	$\frac{64}{100}$	$\frac{89}{100}$	$\frac{91}{100}$	$\frac{0}{100}$	$\frac{80}{100}$	$\frac{92}{100}$	$\frac{0}{100}$
MODE 4			MODE 5			MODE 6		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{64}{100}$	$\frac{94}{100}$	$\frac{13}{100}$	$\frac{54}{100}$	$\frac{59}{100}$	$\frac{76}{100}$	$\frac{74}{100}$	$\frac{71}{100}$	$\frac{32}{100}$
MODE 7			MODE 8			MODE 9		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{72}{100}$	$\frac{76}{100}$	$\frac{43}{100}$	$\frac{7}{100}$	$\frac{43}{100}$	$\frac{39}{100}$	$\frac{48}{100}$	$\frac{46}{100}$	$\frac{79}{100}$
MODE 10			MODE 11			MODE 12		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{43}{100}$	$\frac{48}{100}$	$\frac{19}{100}$	$\frac{22}{100}$	$\frac{20}{100}$	$\frac{82}{100}$	$\frac{18}{100}$	$\frac{13}{100}$	$\frac{46}{100}$
MODE 13			MODE 14			MODE 15		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{8}{100}$	$\frac{15}{100}$	$\frac{96}{100}$	$\frac{1}{100}$	$\frac{8}{100}$	$\frac{88}{100}$	$\frac{0}{100}$	$\frac{3}{100}$	$\frac{86}{100}$

Table 4.3: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 100 with SNR=100.

Since the differentiation can alter the results causing a non correct mode of operation of MCMC during the simulations, 5% of input data are removed from each edge analyzing the first and the second derivatives through TGP method.
As the tables of the results show, varying the measured points or the SNR, it is observed a general tendency in all the studied cases. Indeed, proceeding from mode 1 to mode 15, the performance of TGP decreases.
The reason of that can be explained looking at mode shape function. Indeed, in the last mode (especially 12, 13, 14 and 15), the function, which fits the measured points, has a lot of local maximum and minimum points and this confuses the TGP algorithm and it does not guarantee good performance.
The signal noise to ratio, that reveals the noise level in the measured points, plays an important rule on the final outputs of the model.

100 points SNR=80								
MODE 1			MODE 2			MODE 3		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{8}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{19}{100}$	$\frac{20}{100}$	$\frac{0}{100}$	$\frac{63}{100}$	$\frac{97}{100}$	$\frac{1}{100}$
MODE 4			MODE 5			MODE 6		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{53}{100}$	$\frac{89}{100}$	$\frac{20}{100}$	$\frac{17}{100}$	$\frac{8}{100}$	$\frac{72}{100}$	$\frac{67}{100}$	$\frac{62}{100}$	$\frac{37}{100}$
MODE 7			MODE 8			MODE 9		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{73}{100}$	$\frac{80}{100}$	$\frac{47}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{57}{100}$	$\frac{29}{100}$	$\frac{50}{100}$
MODE 10			MODE 11			MODE 12		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{37}{100}$	$\frac{40}{100}$	$\frac{24}{100}$	$\frac{12}{100}$	$\frac{26}{100}$	$\frac{85}{100}$	$\frac{23}{100}$	$\frac{15}{100}$	$\frac{61}{100}$
MODE 13			MODE 14			MODE 15		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{10}{100}$	$\frac{24}{100}$	$\frac{91}{100}$	$\frac{0}{100}$	$\frac{9}{100}$	$\frac{59}{100}$	$\frac{0}{100}$	$\frac{1}{100}$	$\frac{3}{100}$

Table 4.4: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 100 with SNR=80.

First of all, we can take a look at the tables 4.2 4.3, 4.4 and 4.5 in which 100 points of the beam are been measured. Specifically, it is noticed that, when noise is present, the model fits better mode shapes rather than rotation or curvature (respectively ϕ' and ϕ'') for the first modes. On the contrary, for the last modes the first derivatives achieves the best results, even though the differentiation could lead to an addition of noise.

In the table 4.5 the data related to the curvature (ϕ'') are not present, this is due to the fact that the model is not able to fit adequately the points, because both level of noise (SNR=60) and differentiation noise lead to a lower performance of the TGP model than the previous cases.

Generally, the second derivatives looks to produce better results with a higher number of forks. Actually, the performance is not the best due to the fact that

100 points SNR=60					
MODE 1		MODE 2		MODE 3	
ϕ	ϕ'	ϕ	ϕ'	ϕ	ϕ'
$\frac{0}{100}$	$\frac{3}{100}$	$\frac{4}{100}$	$\frac{0}{100}$	$\frac{2}{100}$	$\frac{0}{100}$
MODE 4		MODE 5		MODE 6	
ϕ	ϕ'	ϕ	ϕ'	ϕ	ϕ'
$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{29}{100}$	$\frac{50}{100}$	$\frac{0}{100}$
MODE 7		MODE 8		MODE 9	
ϕ	ϕ'	ϕ	ϕ'	ϕ	ϕ'
$\frac{19}{100}$	$\frac{57}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{22}{100}$	$\frac{35}{100}$
MODE 10		MODE 11		MODE 12	
ϕ	ϕ'	ϕ	ϕ'	ϕ	ϕ'
$\frac{20}{100}$	$\frac{38}{100}$	$\frac{0}{100}$	$\frac{9}{100}$	$\frac{0}{100}$	$\frac{1}{100}$
MODE 13		MODE 14		MODE 15	
ϕ	ϕ'	ϕ	ϕ'	ϕ	ϕ'
$\frac{4}{100}$	$\frac{12}{100}$	$\frac{0}{100}$	$\frac{4}{100}$	$\frac{0}{100}$	$\frac{1}{100}$

Table 4.5: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 100 with SNR=80.

often the method forks twice because it does not perfectly locate the branch point but a range where the crack is. Different GPs are involved one for each branch tree.

The tables 4.6 and 4.7 show how the numbers of measured points can change the trend of the final output. Indeed, in this case the available data are only for 60 points uniformly distributed along the beam.

Due to the lower number of points, TGPs are less efficient to detect the point where the discontinuity is located. The results, yielding with the first derivatives, produce much more tree branches, instead in the second derivatives the crack effect is less evident.

In the table 4.7 for 60 measured points with adding noise, only the modes 1, 6 and 11 are displayed since the input data for the other modes were not present

60 points SNR=∞								
MODE 1			MODE 2			MODE 3		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{38}{100}$	$\frac{100}{100}$	$\frac{92}{100}$	$\frac{8}{100}$	$\frac{67}{100}$	$\frac{27}{100}$	$\frac{13}{100}$	$\frac{43}{100}$	$\frac{95}{100}$
MODE 4			MODE 5			MODE 6		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{4}{100}$	$\frac{35}{100}$	$\frac{91}{100}$	$\frac{10}{100}$	$\frac{29}{100}$	$\frac{72}{100}$	$\frac{10}{100}$	$\frac{41}{100}$	$\frac{71}{100}$
MODE 7			MODE 8			MODE 9		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{9}{100}$	$\frac{11}{100}$	$\frac{62}{100}$	$\frac{0}{100}$	$\frac{1}{100}$	$\frac{0}{100}$	$\frac{2}{100}$	$\frac{10}{100}$	$\frac{26}{100}$
MODE 10			MODE 11			MODE 12		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{0}{100}$	$\frac{6}{100}$	$\frac{14}{100}$	$\frac{0}{100}$	$\frac{21}{100}$	$\frac{21}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$
MODE 13			MODE 14			MODE 15		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{0}{100}$	$\frac{1}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$	$\frac{0}{100}$

Table 4.6: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 60 with without noise.

in literature.

As it is clear, higher number of points- in which the mode shape is measured - yields a higher efficacy of the TGPs.

The figures 4.1 and 4.5 display the modes 1 and 6 in which: the points represent the measured mode shape and the red line the model interpolation.

In this case the crack is located at 0.3m and 100 points along the beam are measured. The main aim of this thesis is the identification of a known crack position in order to demonstrate the efficacy of the method also when the crack position is not known a priori.

The tree representation is displayed in figures 4.2 and 4.6, both for mode 1 and 6. The figures 4.3 4.7 show the results for the all 100 simulations when mode 1 and 6 are analyzed. The mode 1 has a mean branch point of 0.294m, the model branch

60 points SNR=100								
MODE 1			MODE 6			MODE 11		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{11}{100}$	$\frac{81}{100}$	$\frac{66}{100}$	$\frac{8}{100}$	$\frac{44}{100}$	$\frac{79}{100}$	$\frac{0}{100}$	$\frac{21}{100}$	$\frac{24}{100}$
60 points SNR=80								
MODE 1			MODE 6			MODE 11		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{2}{100}$	$\frac{23}{100}$	$\frac{97}{100}$	$\frac{2}{100}$	$\frac{26}{100}$	$\frac{85}{100}$	$\frac{0}{100}$	$\frac{9}{100}$	$\frac{24}{100}$
60 points SNR=60								
MODE 1			MODE 6			MODE 11		
ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''	ϕ	ϕ'	ϕ''
$\frac{9}{100}$	$\frac{16}{100}$	$\frac{29}{100}$	$\frac{0}{100}$	$\frac{19}{100}$	$\frac{54}{100}$	$\frac{0}{100}$	$\frac{8}{100}$	$\frac{21}{100}$

Table 4.7: Number of forks, which detect the crack at 0.3m, when the number of measurement points is set to 60 with different SNR.

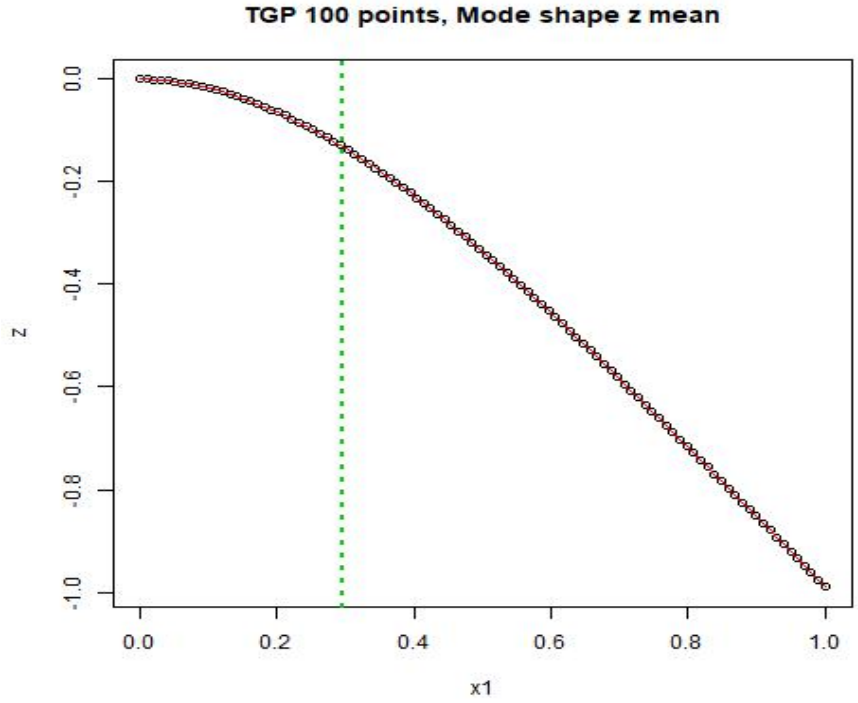

Figure 4.1: Representation of a simulation of mode 1 without adding noise when the beam is divided in 100 points. The measured mode shape ϕ are represented by circle and TGP model regression through red line. The green vertical line locates the branch point coincident with crack position.

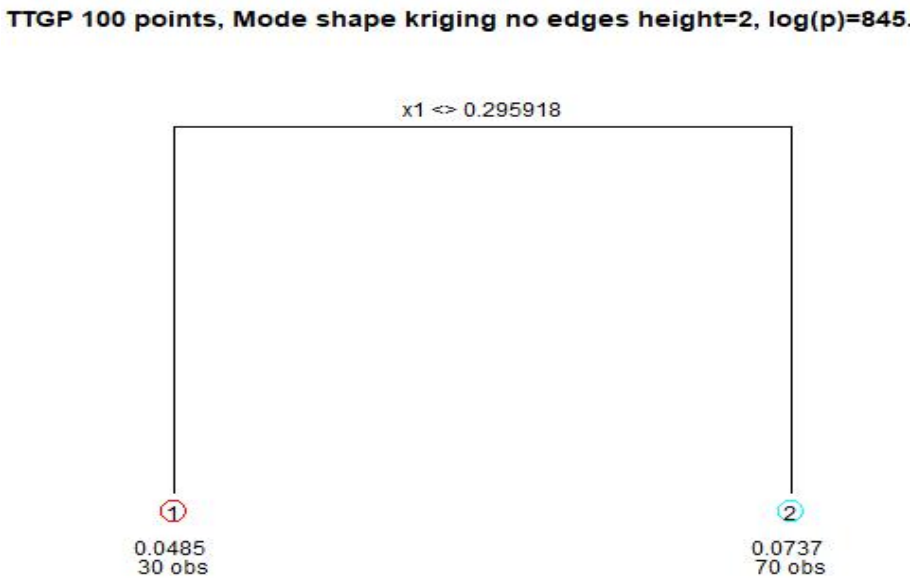

Figure 4.2: Tree partitions of a simulation of the measured mode shape 1 without adding noise when the beam is divided in 100 points.

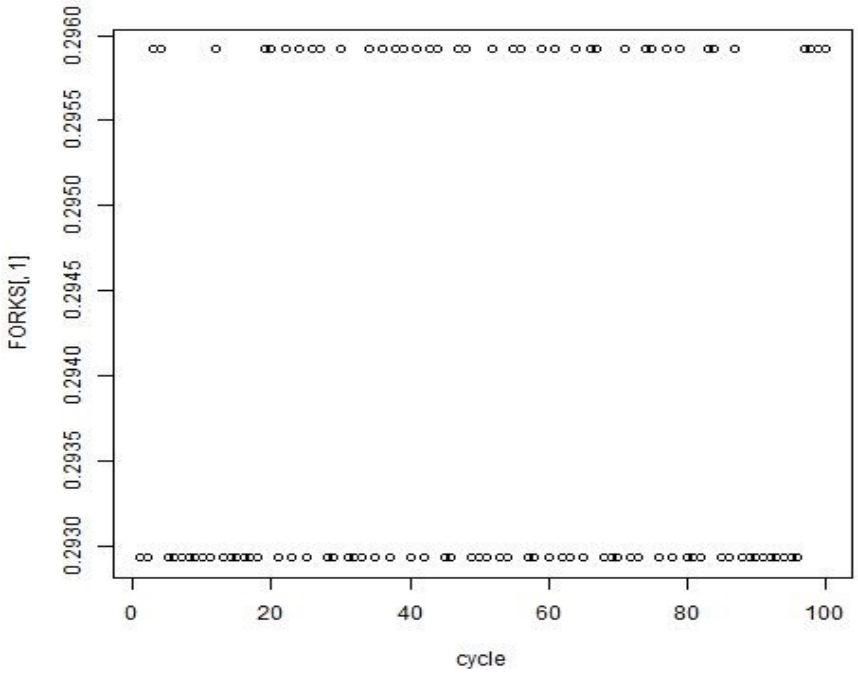

Figure 4.3: Estimation of crack for the mode shape 1 without adding noise when the beam is divided in 100 points.

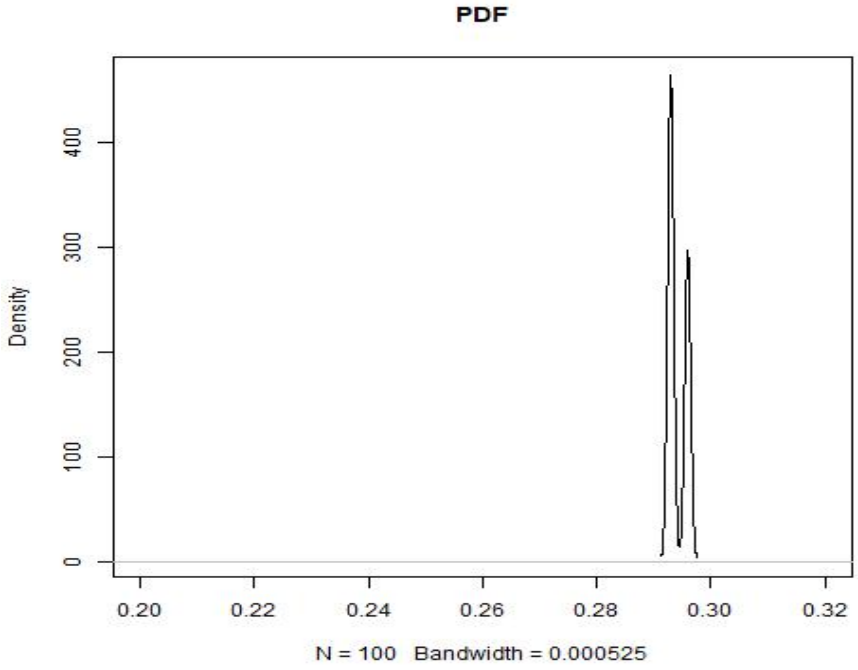

Figure 4.4: Estimation of density function for mode 1 without adding noise when the beam is divided in 100 points.

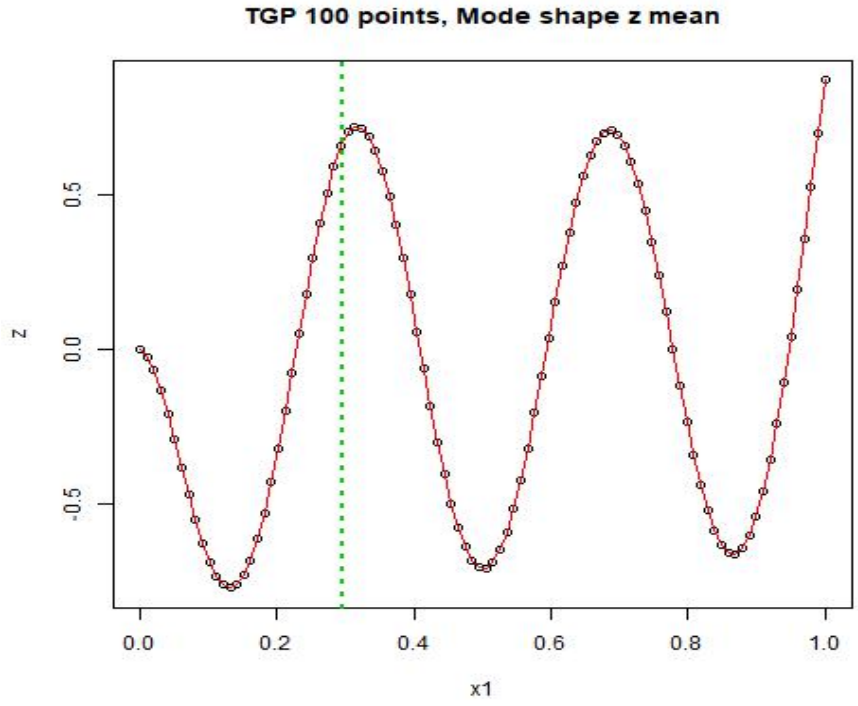

Figure 4.5: Representation of a simulation of mode 6 without adding noise. The measured mode shape ϕ are represented by circle and TGP model through red line. The green vertical line locates the branch point coincident with crack position when the beam is divided in 100 points.

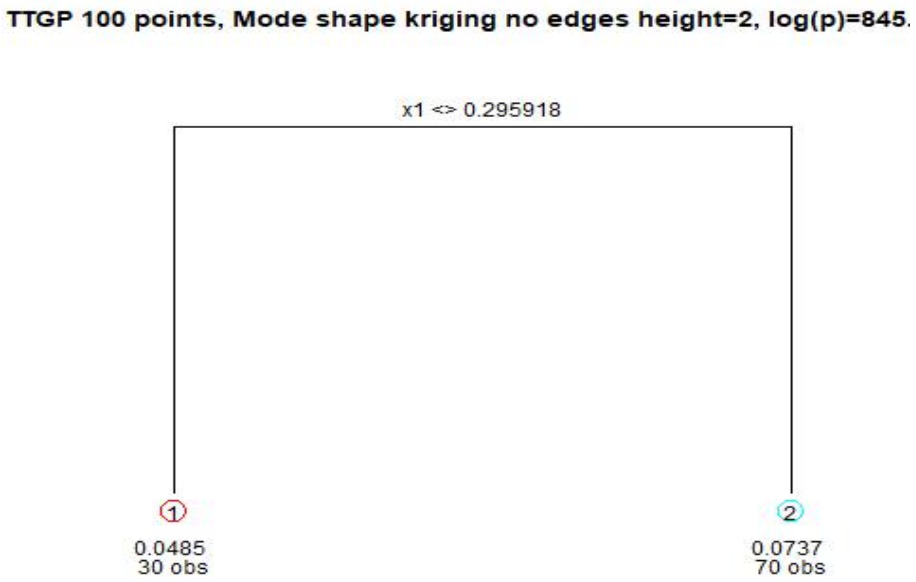

Figure 4.6: Tree partitions of a simulation of measured mode shape 6 without adding noise when the beam is divided in 100 points.

100/100 and the discontinuity is found or in point 0.2929m or in 0.296m. Related to the mode 6 the points are concentrated near 0.3m save the three points which are representative of the two branches of the tree.
By means of the estimation of crack positions (figures 4.3 and 4.7) the estimated probability function is printed for mode 1 and 6 in the figures 4.4 and 4.8. It is noticed that for both modes the data are gathered in a range close to 0.3, specifying that the TGP method fits well the data.

The previous case are analyzed adding a signal noise to ratio equal to 100 and the results are showed in figures: 4.9, 4.10, 4.11 and 4.12.
Due to the noise presence (SNR=100) from the figures 4.9 and 4.10 it seems clear that, related to mode 1, the TGP identifies the crack in points slightly lower or higher than 0.3m. The mean branch point is located at 0.31m.

The figures 4.11 and 4.12 underlines that for mode 6 the TGP fits well the points. Two points appear close to 0.15m because they correspond to the two branch trees case. The mean branch point is 0.294m.
The figures 4.13, 4.15,4.14 and 4.16 concern the the mode 1 and 6 when 60 measured points are available.
The points reduction leads to a lower success of the model, especially for the mode

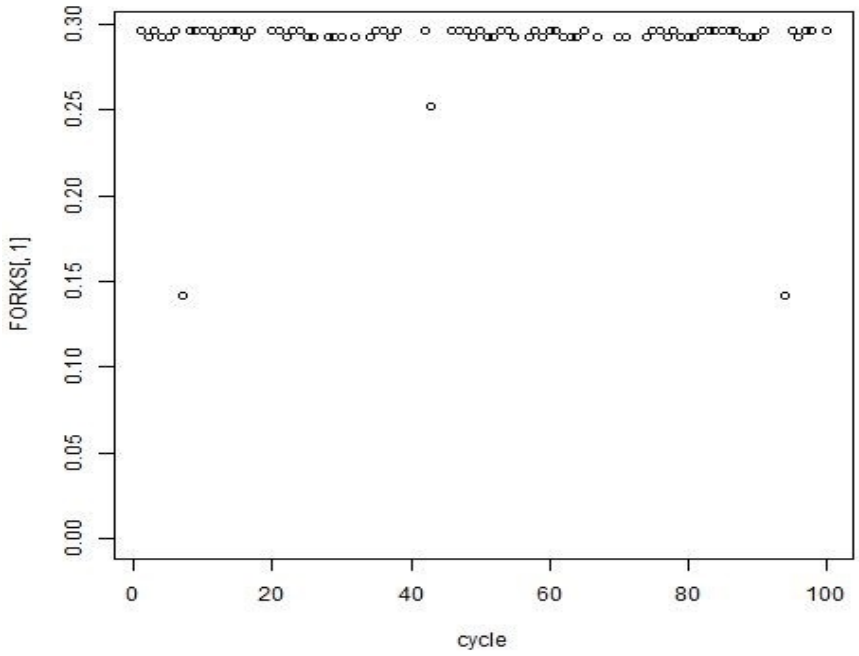

Figure 4.7: Estimation of crack for the mode shape 6 by means of TGP method without adding noise when the beam is divided in 100 points.

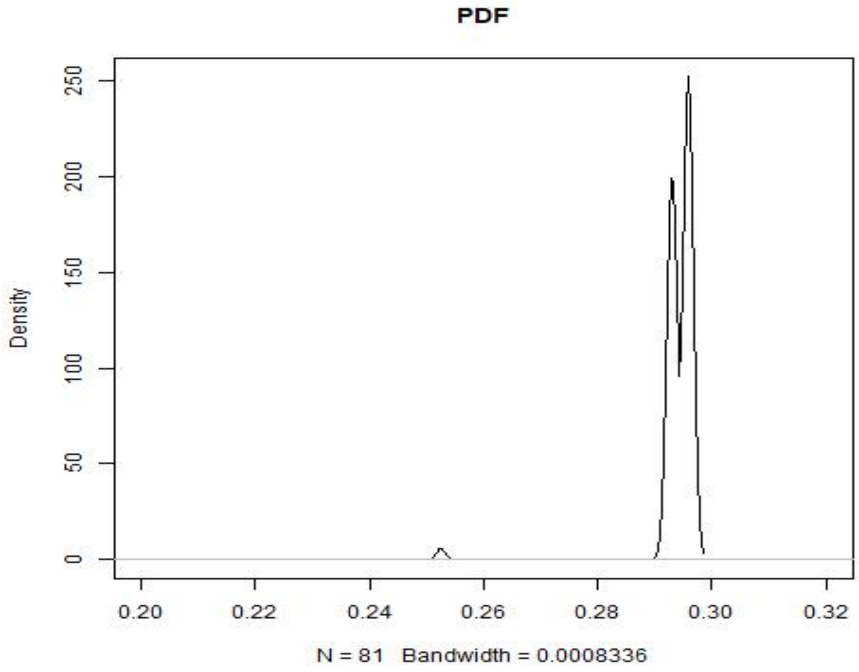

Figure 4.8: Estimated density function for mode 6 without adding noise when the beam is divided in 100 points.

6, because the function is characterized up and down mode shape, and TGPs are not able to detect the crack. The mean branch points are 0.290m and 0.291m for

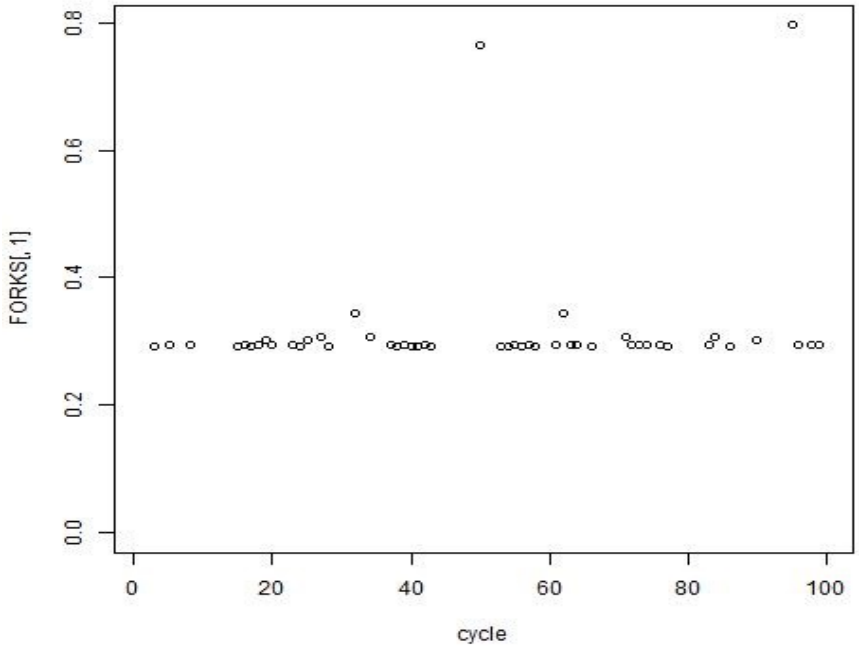

Figure 4.9: Estimation of crack for the mode shape 1 by means of TGP method with SNR=100 when the beam is divided in 100 points.

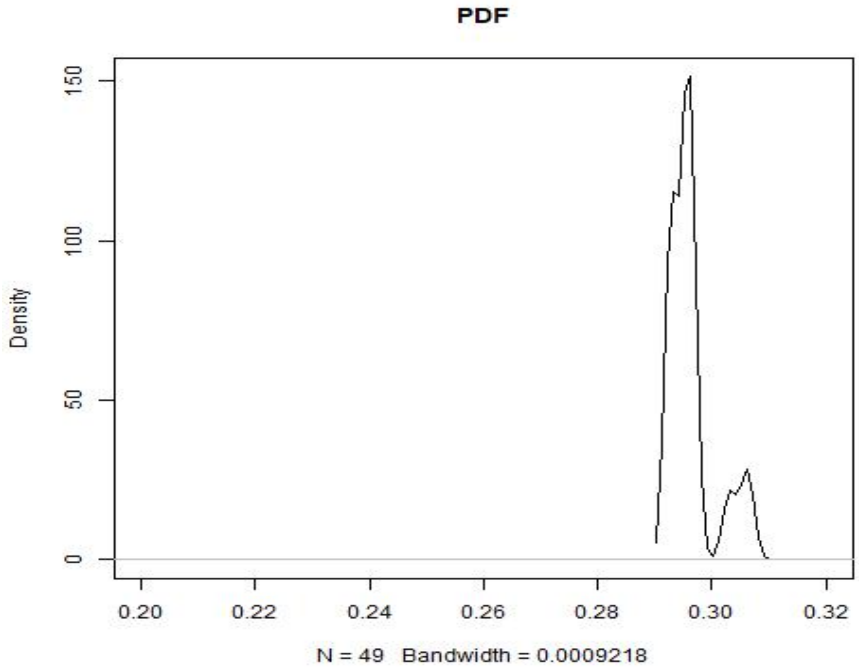

Figure 4.10: Estimated density function for mode 1 with SNR=100 when the beam is divided in 100 points.

the mode 1 and 6 respectively.

When the numbers of points decreases (60 points) and the noise is added, the

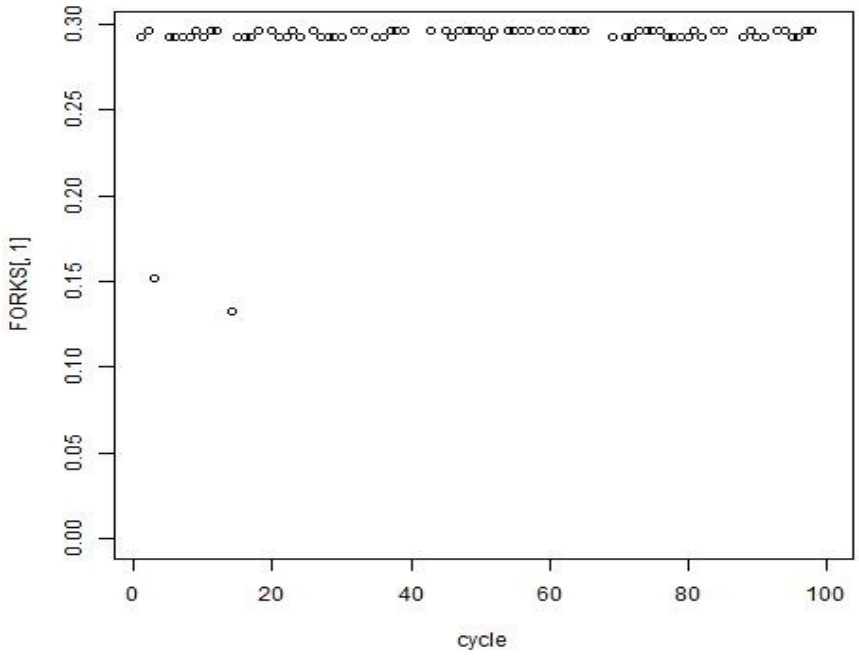

Figure 4.11: Estimation of crack for the mode shape 6 by means of TGP method with SNR=100 when the beam is divided in 100 points.

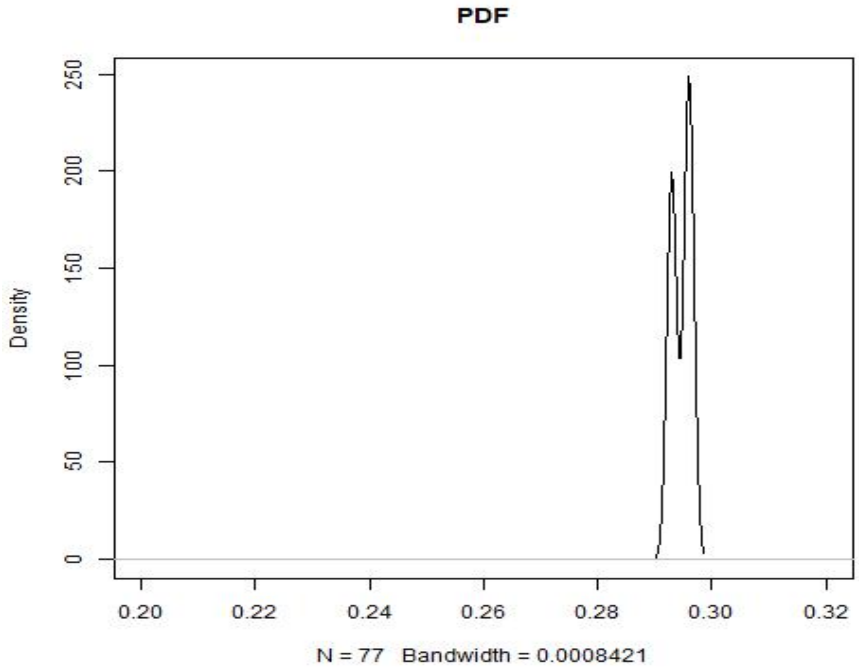

Figure 4.12: Estimated density function for mode 6 with SNR=100 when the beam is divided in 100 points.

rotation (ϕ') is preferred to fit adequately the data by the TGP model. Indeed, the performance are very higher; in figures 4.17 and 4.18 the results are presented

for first derivatives of mode 1 and 6 when SNR is equal to 100.

Due to the lower numbers of points and due to the noise, this technique works well a lower number of times especially for the mode 6 (figure 4.19), where the noise is amplified of the its shape. In both modes- 1 and 6- the range where the results are evaluated is a little higher than 0.3 as showed by the figures 4.18 and 4.20. Overall, also in this case the model appears to be operating well.

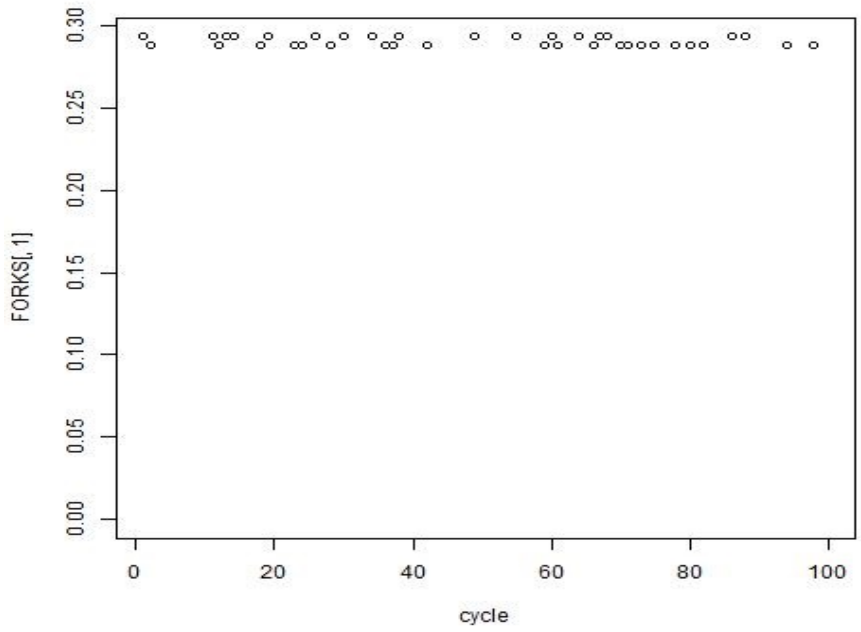

Figure 4.13: Estimation of crack for the mode shape 1 by means of TGP method without noise when the beam is divided in 60 points.

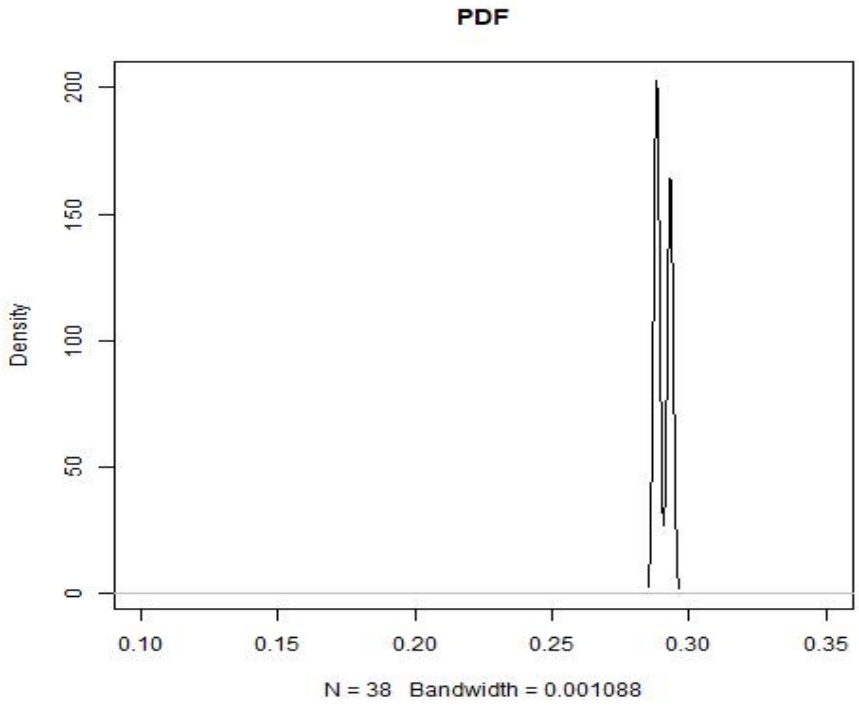

Figure 4.14: Estimated density function for mode 1 without noise when the beam is divided in 60 points.

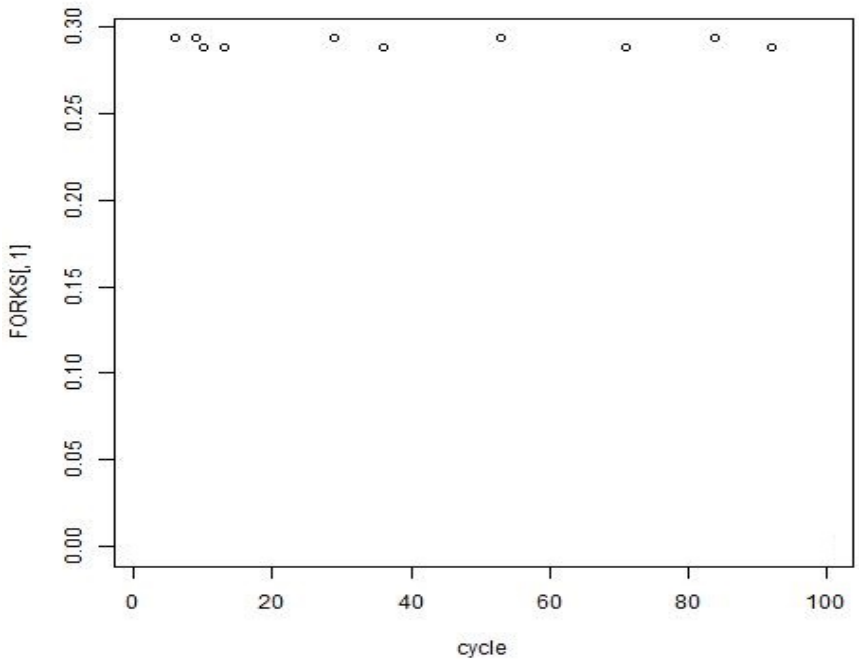

Figure 4.15: Estimation of crack for the mode shape 6 by means of TGP method without noise when the beam is divided in 60 points.

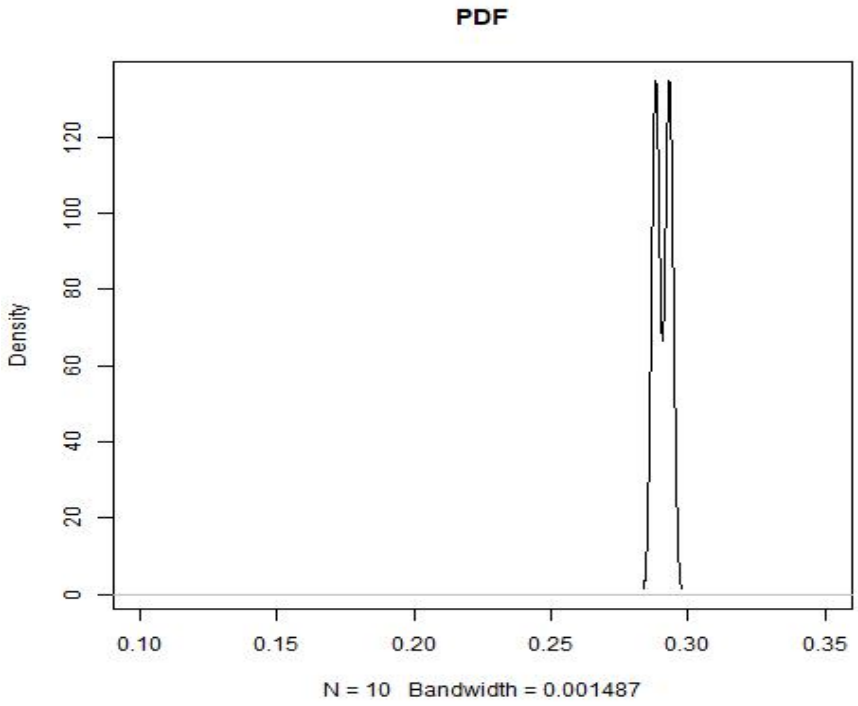

Figure 4.16: Estimated density function for mode 6 without noise when the beam is divided in 60 points.

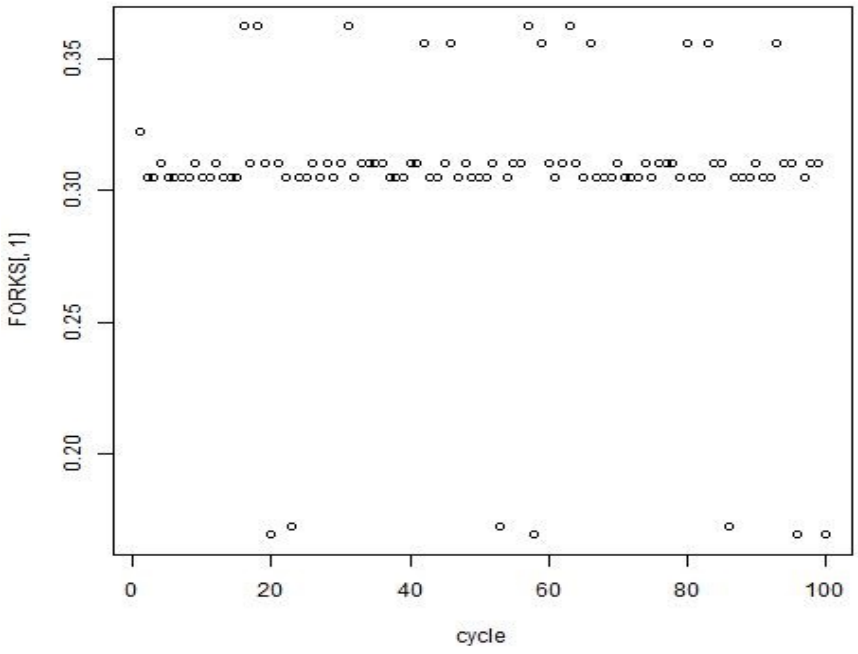

Figure 4.17: Estimation of crack for the first derivatives of the mode shape 1 by means of TGP method with SNR=100 and the beam divided in 60 points.

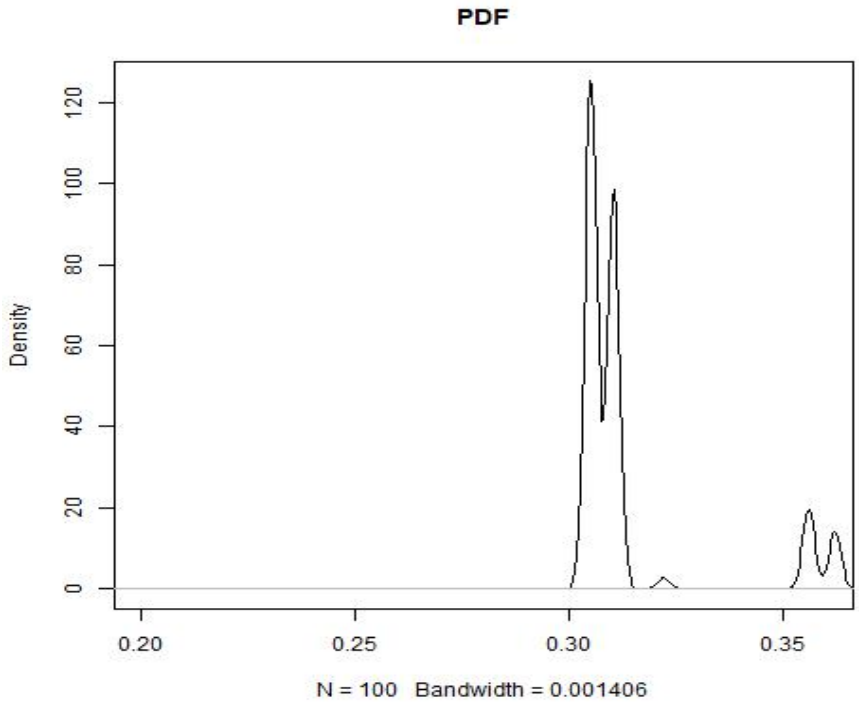

Figure 4.18: Estimated density function for the first derivatives of the mode shape 1 with SNR=100 when the beam is divided in 60 points.

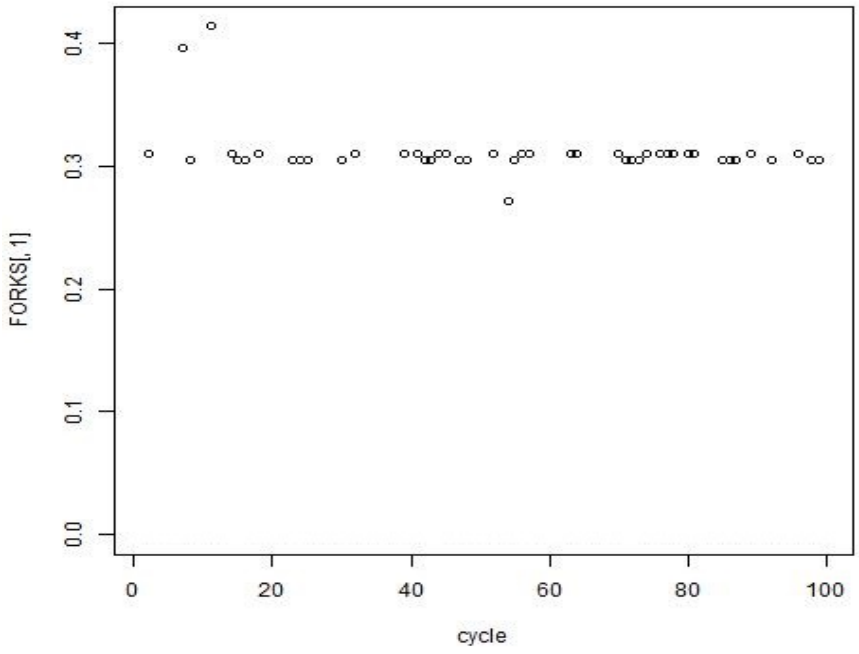

Figure 4.19: Estimation of crack for the first derivatives of the mode shape 6 by means of TGP method with SNR=100 and the beam divided in 60 points.

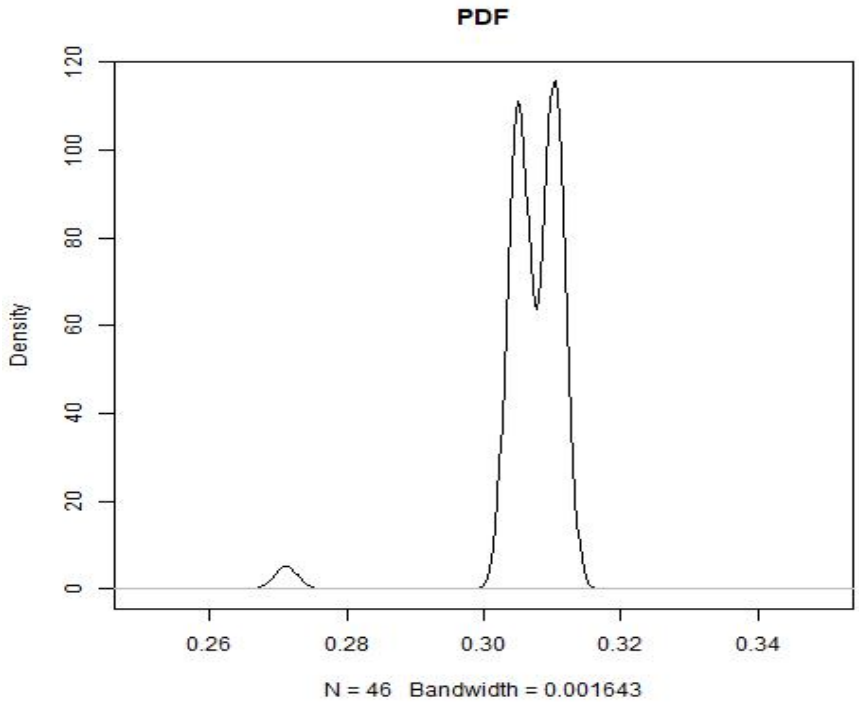

Figure 4.20: Estimated density function for the first derivatives of the mode shape 6 with SNR=100 when the beam is divided in 60 points.

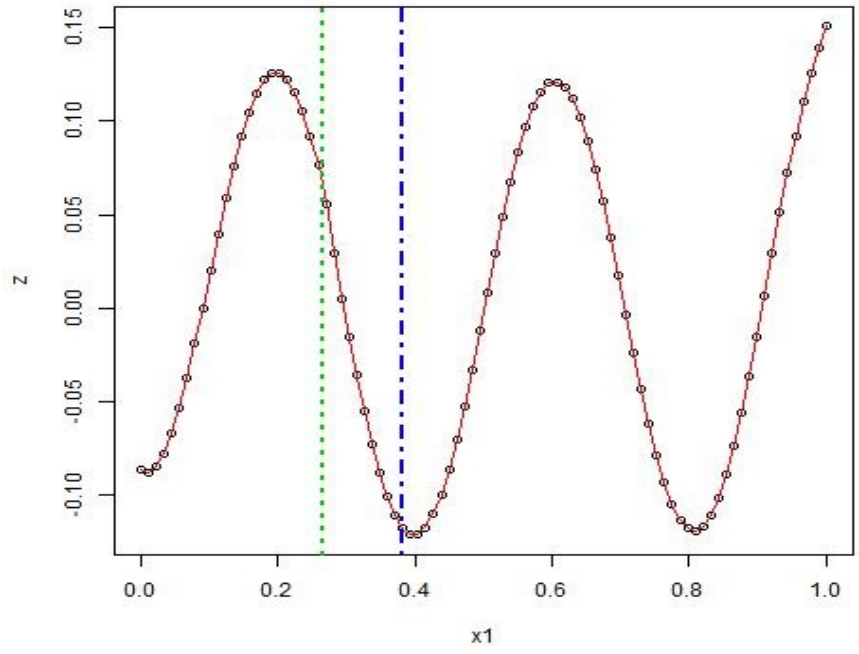

Figure 4.21: Simulation of first derivatives of mode 6 whit SNR=100. The beam is divided in 60 points.

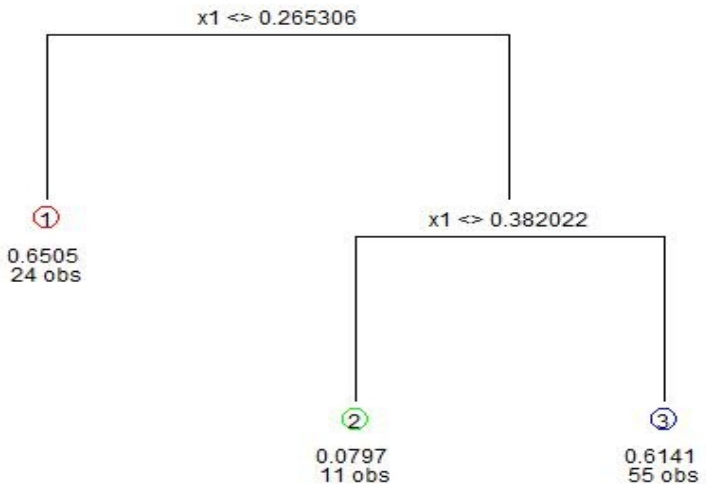

Figure 4.22: Tree partitions of first derivatives of mode 6 whit SNR=100. The beam is divided in 60 points.

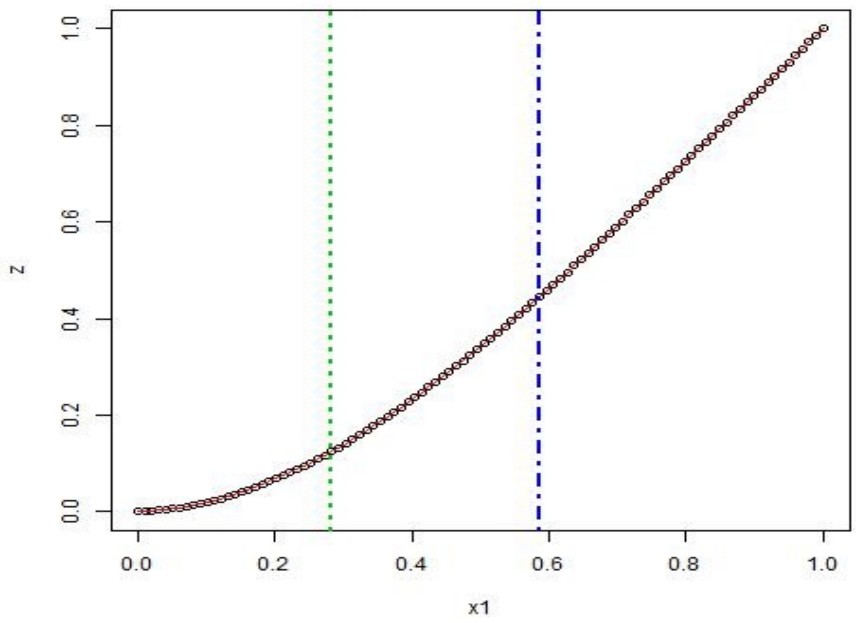

Figure 4.23: Representation of a simulation of mode 1 without adding noise when the beam is divided in 100 points. The measured mode shape ϕ are represented by circle and TGP model regression through red line. The green and blue vertical lines locate the branch points coincident with the two crack positions.

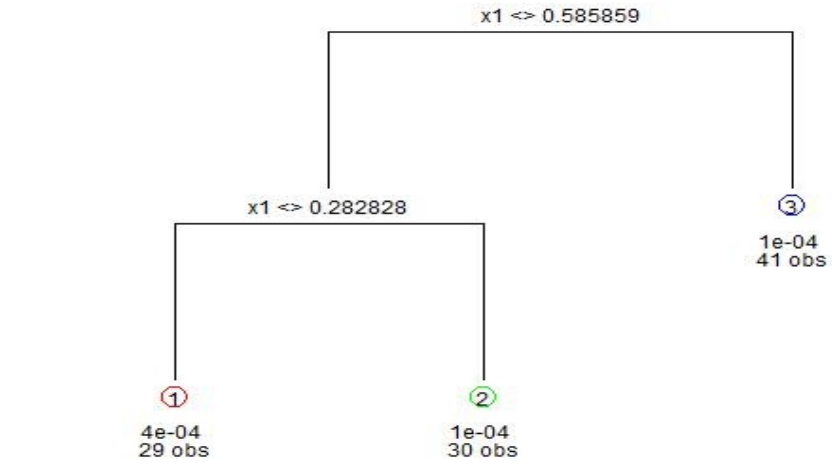

Figure 4.24: Tree partitions of a simulation of measured mode shape 1 without adding noise when the beam is divided in 100 points with 2 cracks at 0.3m and 0.6m.

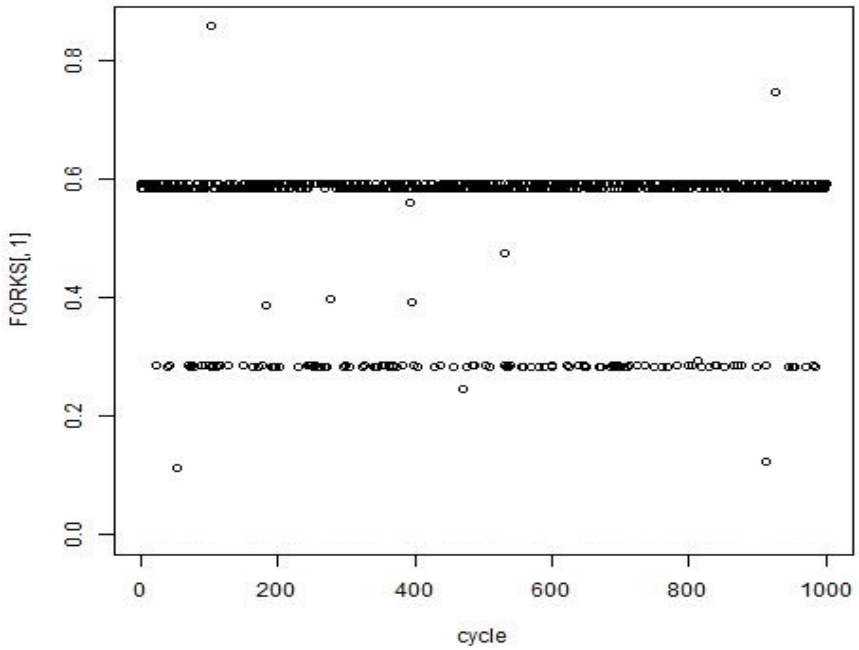

Figure 4.25: Estimation of cracks for the first mode shape without added noise for 100 measured points for 2 cracks case.

Some of the analyzed modes do not identify precisely the position of crack but

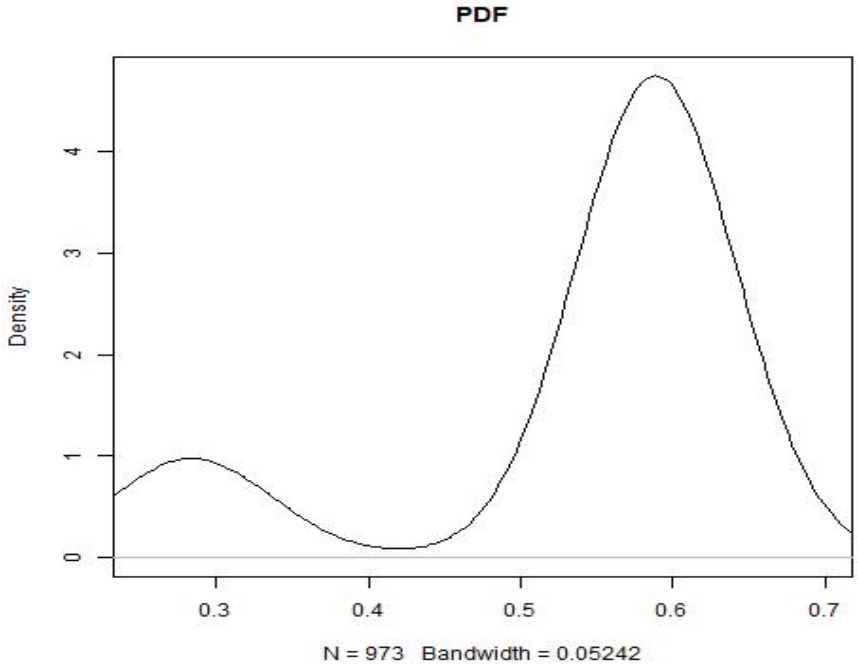

Figure 4.26: Estimated density function for the first mode shape without added noise for 100 measured points for 2 cracks case.

they branch twice. An example of that, as regards the first derivatives of mode 6 with SNR=100, is presented in figure 4.21. The model identifies the three different areas characterized by three different GPs. The two branches, represented by the green and the blue lines, identify a range and not a single point in which the crack is included. The final set of results are presented for mode 1 in the case of 2 cracks; the measurements along the beam are 100. The cracks are localized at 0.3m and 0.6m respectively from the fixed end of the beam. The simulations are run 1000 times in order to have a good chance of adequate performance of the model. The results show (see figures 4.23, 4.24, 4.25 and 4.26) that both the cracks are underestimated. The mode 6 does not produce satisfactory results in this case. The model for the mode 1 forks twice 179/1000 and once 563/1000 in which most times the crack at 0.6m is detected.

Finally, taking a look of the presented results , the number of nodes along the beam conditions the performance.
The first modes are fit better when no noise is not added to the data.
The noise free case is not real, so when the different levels of noise are added on the input data the mean branch point is a value higher than 0.3m.

For 100 measured points a more detailed analysis can be carried out because the available data allow to detect crack in each event.
In this case when noise is added it is clear that the middle modes - for 6 to 10- seems to be producing better estimations of crack. Sometimes no results are produced, as mode 15, the reason of that could be linked to the fact that the node could coincide with the crack, in that case the model is not able to record any result.

4.3 Genetic algorithm applied to the tree parameters

In order to improve the results the tree parameters are optimized by means of the Genetic algorithm. These parameters, as known, rules the size and the shape of the tree.
The tree parameters α and β are defined in a range of [0,1] and [0,2] respectively. According to Chipman [4] the chosen values are 0.5 for α and 2 for β. These values are determined experimentally. Varying them manually, it was noticed an improvement of the results.
This is only a preliminary study due to the large amount of data needed. The simulations are carried out for the mode 2 without adding noise when 100 measured points are distributed along the beam.
The Genetic algorithm, fixing a population size equal to 30 and a number of iterations equal to 15 produced that the best values for α and β are 0.75 and 1.26 respectively. Replacing these values in the tree vector, the tree branch once 94/100, the same result of the default values. The reason of that it is detectable in the low number of generations of the algorithm.
The next step, an improvement, is that to set the genetic algorithm parameters so as to find the correct bond between population size and number of iterations/-generations.

Chapter 5

Conclusions

The aim of this thesis is the detection of crack in beam by means of the method: Treed Gaussian process.

The overall goal has been achieved giving satisfactory results. The TGP results are satisfying when the input data are the measured mode shape. However, the method has also a high percentage of success when the first and the second derivatives are employed.

One of the achievement, compared to the past applied method, is that the "*output data only*" is needed. This means that only the data measured by the sensors are relevant for a good estimation of the model and physic model of the beam is not requested.
The most advantage is that it works well for any number of cracks because it has the potential to find the global minimum of likelihood also when multiple cracks occur, unlike to the GPs.
Another important and positive remark is the Bayesian approach of the method, because it provides a way of combining prior information with data and also to obtain an inference conditioned on data too.

TGP is defined as "*base line free*" because the method is not based on the comparison between damaged and undamaged beam.

The main drawback is the large time consuming for the process because TGP is based on complicated algorithm and a large number of hyperparameters must be set. The computation time is adversely affected by the large input dataset

needed in order to draw consistent results. Moreover, a large amount of input data is possible only if particular instrumentation is employed, e.g. laser vibrometry.
The last observation, looking at the drawn results, is that the model is sensible at presence of noise especially when the number of measured inputs are low.

The results can be improved by means of an optimization algorithm.
It was thought that genetic algorithm was a good choice. It is applied to the prior tree parameters because they are not known a priori. This can be a limit for a good mode of operation of the model but the limitation of a large time consuming during the computation does not allow a more accurate study. This optimization will be accomplished in the future.

Bibliography

[1] H.S. Stern D.B. Dunson A. Vehtari D. B. Rubin A. Gelman, J.B.Carlin. *Bayesian Data Analysis, 2nded.* Chapman and Hall/CRC, 1998.

[2] J. R. Lee C. C. Ciang and H. J. Bang. Structural Health Monitoring for a wind turbine system: a review of damage detection method. . *Measurement Science and Technology*, 2008.

[3] C.E. Williams C. E. Rasmussen. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[4] George E.I. McCulloch R.E. Chipman, H.A. Bayesian CART model search. . *Journal of the American Statistical Association*, 1998.

[5] K.Worden C.R. Farrar. *Structural Health Monitoring: A Machine Learning Perspective.* WILEY, 2012.

[6] N. Cressie. *Statistics for Spatial Data.* John Wiley and Sons, Inc, 1993.

[7] N.GuJ. G.L.QianS. and S.Jiang. The dynamic behaviour and crack detection of a beam with a crack. *Journal of Sound and Vibration*, 1990.

[8] R. B. Gramacy. BAYESIAN TREED GAUSSIAN PROCESS MODELS., 2005.

[9] R. B. Gramacy. tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models., 2007.

[10] C. Surace J. Hensman and M. Gherlone. Detecting mode-shape discontinuities without differentiation- examining a Gaussian process approach. *9th International Conference on Damage Assessment of Structures (DAMAS).*, 2011.

[11] R. A. Olshen C.J. Stone L. Breiman, J. H. Friedman. *Classification and Regression Trees.* Chapman and Hall/CRC, Boca Raton, 1984.

[12] C. Surace M. Civera and K.Worden. Detection of Cracks in Beams Using Treed Gaussian Processes. *Structural Health Monitoring and Damage Detection*, 2017.

[13] H. K. H. Lee R. B. Gramacy. Adaptive design and analysis of supercomputer experiments. 2008.

[14] H. K. H Lee R. Gramacy. Bayesian Treed Gaussian Process Models With an Application to Computer Modeling. . *Journal of the American Statistical Association 2nded*, 2012.

[15] M.A. Taddy R. Gramacy. Bayesian Treed Gaussian Process Models, Package tgp., 2016.

[16] M.A.Taddy R. Gramacy. Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with tgp Version 2, an R Package for Treed Gaussian Process Models. *Journal of Statistical Software*, 2010.

[17] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. 1959.

[18] Luca Scrucca. GA: A package for Genetic Algorithms in R., 2013.

[19] Luca Scrucca. Package "ga"., 2016.

[20] J.Rowson W. Becker, K.Worden. Bayesian sensitivity analysis of bifurcating non linear models. *Mechanical Systems and Signal Processing*, 2012.

List of Figures

List of Tables