



**Escola de Camins**  
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports  
UPC BARCELONATECH

# Surrogate data-driven models for parametric analysis of vehicle structural dynamics for NVH assessment

Treball realitzat per:  
**Toni RIERA ROIG**

Dirigit per:  
**Pedro DIEZ MEJIA**  
**Sergio ZLOTNIK MARTINEZ**

Màster en:  
**Enginyeria de Camins, Canals i Ports**

Barcelona, *26/06/2020*

Departament d'Enginyeria Civil i Ambiental

**TREBALL FINAL DE MÀSTER**

## ABSTRACT

The limitations of numerical simulations by a lack of computational resources has favoured the development of approaches that aim at reducing the computational cost of these simulations, while retaining a high degree of reliability. In the context of automotive structural analysis, we present the development of a surrogate model for linear elasticity through the implementation of Model Order Reduction (MOR) techniques. This document is divided in two main parts: first, an overview of the techniques used, as well as its role in the construction of the surrogate model, and a discussion on its implementation through the commercial software NASTRAN; and second, the analysis of the results obtained for different cases of linear and non-linear MOR implementations, and of the results of the surrogate model itself, with respect to the results offered by a direct NASTRAN simulation.

# TABLE OF CONTENTS

1	INTRODUCTION .....	3
2	DIMENSIONALITY REDUCTION THROUGH PCA, POD AND kPCA. ....	7
2.1	PCA and POD.....	8
2.2	kPCA.....	12
3	SURROGATE MODEL CONSTRUCTION THROUGH PCA .....	17
3.1	Linear elasticity in FEM .....	17
3.2	Application of PCA results.....	18
3.3	Surrogate model formulation .....	19
4	DATA GENERATION .....	20
4.1	NASTRAN model.....	20
4.2	Definition of parameters and data generation.....	27
5	ANALYSIS OF POD AND kPCA RESULTS .....	30
5.1	Implementation case 1 .....	34
5.2	Implementation case 2 .....	37
5.3	Implementation case 3 .....	40
5.4	Implementation case 4 .....	43
5.5	Implementation case 5 .....	46
5.6	Results comparison.....	48
6	SURROGATE MODEL IMPLEMENTATION .....	53
6.1	Analysis of results .....	54
7	CONCLUSION.....	57
8	REFERENCES .....	59

# 1 INTRODUCTION

Numerical analyses have become a standard practice among the scientific and industrial world, to help in the design and optimisation processes. From fluid dynamics to micro-electro mechanical systems, several methods, such as Finite Elements, have proven their usefulness in modelling and computing different natural phenomena that couldn't be solved with traditional analytic methods.

The solutions of certain partial differential equations have been obtained thanks to the development of computational techniques. As time has passed, new simulation techniques have required more and more computational resources, as greater accuracy in the modelling stage has been deemed necessary. However, a greater demand for not only greater accuracy, but also speed and reliability, has emerged, which contrast to the ever increasing computational requirements of modern simulations.

In the same sense, traditional computational techniques may not be suited for approaches based on trial and error, as its computational cost can be just too large to make them viable. This is especially certain in models characterized by several parameters, whose effects in the overall system cannot be assessed by simply carrying out a massive amount of simulations because of this.

On the other hand, another question that has been raised is that of the "democratisation" of simulations, or the push for a higher accessibility where potent computers are not required, in favour of portable devices such as tablets or phones.

Hence, in this situation, an approach favouring the simplification of computational technics, while conserving a high degree of reliability, has been gaining momentum, as a way of increasing the efficiency of current technology. This responds to the need for higher speed and reliability, the high number of simulations required to perform certain optimisation problems, and the push for a simplification on simulation models and its computational requirements.

Several methods have been being developed over the years, from machine learning technics, to dimensionality reduction approaches. Many focus on the idea of dividing the overall necessary work on an *offline* work, and an *online* work, where a large amount of computations are carried out first to generate a model that allows the user to obtain faster and reliable results.

In this sense, reduced models have emerged to answer to this demand. When many possible realizations of a simulation need to be carried out, the computational resources available may not be enough. Hence, reduced models present low-dimensional, efficient, and fast approaches capable of providing a high reliability.

Several parameters are usually needed to compute the results of partial differential equations, such as material properties, system geometry, boundary conditions... The idea behind parametric model reduction is to generate models capable of describing the behaviour of the original one in a certain range of parameters, so that fast results can be obtained. This is a really useful technique in many cases: design, control, optimisation, or uncertainty quantification.

In the same sense, and combined with these techniques, model order reduction (MOR) are techniques whose objective is to reduce the complexity of certain models and simulations, by reducing the degrees of freedom that need to be considered. Examples of these are the Proper Orthogonal Decomposition (POD), and the Proper Generalized Decomposition (PGD), but many more can be mentioned, like the Singular Value Decomposition (SVD), Principal Component Analysis (PCA), kernel-Principal Component Analysis (kPCA)... In the case of the POD, its objective is to extract the most significant elements of a system, and to represent them in a set of basis vectors that can provide a reliable representation of the system's behaviour. Not only that, but it also provides a way to simplifying interpolations and to deal with missing data.

As said by [5], the origin of these techniques can be difficult to define, as many of them are equivalent. For example, the SVD was first independently derived during the XIXth century by scholars like *Beltrami (1873)* and *Jordan (1874)*, though we'll have to wait until the XXth century to find applications such as those of *Fisher and Mackenzie (1923)*. Other methods, like the PCA, appeared early on the XXth century, thanks to the work of

*Pearson (1901)* and *Hotelling (1933)*, who derived it through similar (but different) approaches. Other authors, such as *Frisch (1929)* and *Thurstone (1931)*, were also working in similar methods, but *Pearson's* and *Hotelling's* contribution is regarded as more significant, even some of their work (*Hotelling* in particular) contains material on the factor analysis.

Still, not much progress was seen until a few years later, as these methods required a significant computation power that had yet to be developed. Though *Pearson* claimed that the PCA could be carried out by hand with a small number of parameters, this didn't seem to be a possibility; however, once computers came to be, the application of these techniques became an actual possibility. Other important papers regarding the PCA, that were developed during this period of popularisation, are those of *Anderson (1936)*, *Rao (1964)*, *Gower (1966)*, and *Jeffers (1967)*.

It's important to note, too, that these methods have been used in several fields. For example, a remarkable paper on meteorology and oceanography by *Preisendorfer and Mobley (1988)* also introduces several new ideas related to the PCA. Other examples presented by [6] would be those of *Graham and Kevrekedis (1996)*, that apply the POD on reaction-diffusion chemical processes, or *Epureanu et al. (2001)*, on viscous flow.

Regarding the field of structural dynamics, the first applications are found on the 1990s, dealing with distributed systems [*Fitzsimons and Rui (1993)*], dimensionality studies [*Cusumano et al. (1993-1994)*], or vibration of long torsional strings [*Kreuzer and Kust (1996)*]. Currently, we can find applications in fields such as active control, aeroelastic problems, finite element model updating, any many more.

In this document, several MOR techniques will be presented, for both linear and non-linear cases. Our interest is to assess its implementation on Finite Element Method data, in order to find the underlying dimensions that can be obtained from several approaches. Particularly, we will work on the implementation of a linear elastic model of an automotive structure in the software NASTRAN, in order to assess its particularities and challenges. This software will be used both because of its capabilities, and because we are interested in carrying out this project in a commercial software.

This project needs to be understood in the context of Fabiola Cavaliere's project on static and dynamic global stiffness analysis for automotive pre-design, which aims at devising a computational tool to guide the Body-in-White (phase where the final contours of the car body are worked out) designer in the decision making, through the use of reduced order modelling for linear static and dynamic analysis. This project is supervised by professors and professionals from the UPC, the CIMNE, Swansea University, and SEAT. These supervisors include the tutors of this TFM: Pedro Díez and Sergio Zlotnik.

This document will proceed as follows: First of all, the MOR methods will be explained in detail with their appropriate notation, and then, we will explain how to create a surrogate model from the results obtained from these methods, to solve a FEM problem in a smaller dimension. Once this has been done, we will explain how the data needed to carry out these tasks will be obtained. First, a FEM NASTRAN model, provided by the LaCàN and representing the structure of a car, will be presented. Then, we will specify the parameters that will be subject to variation, in order to define the amount of simulations that will be required. From these results, a set of data will be obtained, and it will be used to implement the MOR techniques. After having discussed the results obtained from the dimensionality reduction methods, they will be used to create a surrogate model according to what will have been explained, which will be tested with a new simulation.

## 2 DIMENSIONALITY REDUCTION THROUGH PCA, POD AND kPCA.

In this section, we will adequately introduce the dimensionality reduction methods that will be used in this project.

As said before, the need for these models comes from the fact that current computational resources can't handle, in a reasonable amount of time, certain large scale simulations that require multiple tries. Hence, in this section, we present several models that respond to this demand, by reducing the size of the data of interest. These models have several applications, from reducing the dimensions of the space of solutions for a certain given problem in order to carry out the calculations, to reducing the size of a given data set so that it can be more easily exploited for other analysis. The former will be developed in section 3, where the conclusions from this section will be applied.

In this section, we will present three different methods, though two of them are equivalent. First, the Principal Component Analysis (PCA) and the Proper Orthogonal Decomposition (POD), which were already introduced in the introduction, and are equivalent linear methods; then, the kernel-Principal Component Analysis (kPCA), which is a non-linear method based on the PCA, with a few variations that need to be considered. We will use a precise notation in order to characterize each method, and to establish their similarities and differences.

Though these models are capable of treating any sort of organized data, and the notation will be general enough to represent this, it is important to note that, in this document, a general "data" vector will always represent the displacement results of a FEM model. Later, this will be explained in greater detail.



## 2.1 PCA and POD

The Principal Component Analysis (PCA) and the Proper Orthogonal Decomposition (POD) are equivalent methods that aim at reducing the dimension of a set of data. The idea behind them is to eliminate a possible correlation between the different sets of data, through a conversion of the original variables into a set of linearly uncorrelated variables.

As previously said, these methods were created and developed during the twentieth century. Many equivalent methods were also created, and some of them will also be discussed on this document, such as the Singular Value Decomposition (SVD). This is because two main reasons: first, the SVD can be viewed as the reason why the PCA and the POD are considered equivalent methods; second, because these transformations help understand the logical evolution of the PCA into the so called kPCA, or kernel Principal Component Analysis: an extension of the PCA when non-linearities need to be taken into account.

The idea behind the PCA and the POD is quite simple, yet smart: Given a set of data, its covariance matrix (or equivalent) is calculated and diagonalised. Diagonalising this matrix carries out a change of basis, from correlated variables to uncorrelated variables. Since the sample's variability will be defined as the diagonal values of the matrix, the largest eigenvalues will be able to represent a large part of the total variability. Hence, a tolerance is established, in order to define the minimum number of orthogonal variables that need to be taken into account to satisfy a given variability, and reduce the dimension of the initial sample without losing valuable information.

In the following sections, the PCA and the POD will be properly explained using the adequate mathematical notation, and the necessary considerations regarding the other equivalent methods will be addressed. Then, the kPCA will be introduced, outlining the differences with the first methods and its particularities. The following paragraphs are based on [3], though further information can be found in [1], [2], [4] and [6].

### 2.1.1 DATA SET AND COVARIANCE MATRIX.

The data will be defined as follows: vectors  $x^1, x^2, \dots, x^{n_s} \in \mathbb{R}^d$  are  $n_s$  samples of size  $d$ , usually with  $n_s \gg d$  so that the set of vectors is representative of the random variable that they represent. Then, this set of vectors is used to construct the following matrix:

$$X = [x^1 \quad x^2 \quad \dots \quad x^{n_s}] \quad (1)$$

Which is of size  $d \times n_s$ . The final objective of the PCA is to reduce the size  $d$  to a size  $k$ , with  $d \gg k$ , by eliminating the intrinsic correlation between sets of data.

Having defined  $X$ , the covariance matrix of the sample (as long as its mean value is zero) is defined as:

$$C = XX^T = \sum_{l=1}^{n_s} x^l (x^l)^T \quad (2)$$

Of size  $d \times d$ . In case the mean value of  $X$  is not zero, it can be easily transformed by subtracting its mean to each one of the vectors  $x^l$  forming the matrix. In any case, the diagonalisation of  $C$  results in the following expression:

$$C = U\Lambda U^T \quad (3)$$

Where  $\Lambda$  contains the eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$  in its diagonal, and  $U$  is composed of the vectors  $U = [u^1 \quad u^2 \quad \dots \quad u^d]$ , that describe an orthonormal basis in  $\mathbb{R}^d$ . It is important to notice that, since the covariance matrix is diagonal in the new basis  $U$ , these variables are uncorrelated in this basis. Also, a vector  $x$  in the original basis, becomes  $z = U^T x$  in the new basis, and a matrix  $X$  becomes  $Z = U^T X$ .

While the PCA follows this path, the POD (Proper Orthogonal Decomposition) proceeds through a different matrix, defined as:

$$G = X^T X = \sum_{l=1}^{n_s} (x^l)^T x^l \quad (4)$$

In this case,  $G$  has a size of  $n_s \times n_s$ . Likewise, it can be diagonalised, obtaining the resulting expression:

$$G = V\tilde{\Lambda}V^T \quad (5)$$

With  $\tilde{\Lambda}$  containing its eigenvalues, and  $V = [v^1 \ v^2 \ \dots \ v^{n_s}]$ , describing an orthonormal basis, as  $U$  did.

At this moment, it's worth mentioning that the Singular Value Decomposition (SVD) provides a decomposition of  $X$  into the form  $X = U\Sigma V^T$ , being  $U$  and  $V$  the matrices previously defined, and  $\Sigma$  a diagonal  $d \times n_s$  matrix that takes the following form:

$$\Sigma = \begin{bmatrix} \sigma_1 & & 0 & \dots & 0 \\ & \ddots & & & \\ & & \sigma_d & 0 & \dots & 0 \end{bmatrix} \quad (6)$$

In fact, the diagonalisation of  $C$  can be seen as a consequence of the SVD, as

$$C = XX^T = U\Sigma V^T V \Sigma^T U^T = U(\Sigma \Sigma^T)U^T \quad (7)$$

given that  $V$  is a basis of orthonormal vectors. Therefore,  $\Lambda = \Sigma \Sigma^T$ , and its eigenvalues  $\lambda_i$  are  $\lambda_i = \sigma_i^2$ . The same can be done for  $G$ , obtaining:

$$G = X^T X = V(\Sigma^T \Sigma)V^T \quad (8)$$

Therefore,  $\tilde{\Lambda}$  has  $d$  eigenvalues that aren't equal to zero, which coincide with the eigenvalues in  $\Lambda$ .

This shows how  $C$  and  $G$  are equivalent matrices, used in different dimensionality reduction methods. While  $C$  is the matrix used in the PCA, the approach used for  $G$  will be used in the kPCA, for reasons that will later be explained.

## 2.1.2 DIMENSIONALITY REDUCTION

Considering the PCA (though this can also be done with the results of the POD), once the matrix  $C$  has been diagonalised, a value  $\varepsilon$  needs to be chosen to perform the dimensionality reduction. Being the trace of  $\Lambda$ , or the sum of  $\lambda_i$ , the total variance of the sample, if the sum of a small number  $k$  of  $\lambda_i$ , with  $k \ll d$ , is already close to the total sum of all the  $d$  eigenvalues, this means that these  $k$  dimensions contain a significant amount of the total variability. Therefore, this value  $\varepsilon$  is chosen such that:

$$\sum_{i=1}^k \lambda_i \geq (1 - \varepsilon) \sum_{i=1}^d \lambda_i \quad (9)$$

The dimensionality reduction is then performed by choosing the first  $k$  dimensions of the diagonalised  $\Lambda$  matrix, and ignoring the  $k + 1$  to  $d$  remaining dimensions. It is easy to see how  $k$  depends on  $\varepsilon$ , as the closer  $\varepsilon$  is to 1, the closer  $k$  will be to  $d$ , while the closer  $\varepsilon$  is to 0, the smaller  $k$  will be, tending to 1.

Once  $\varepsilon$  has been defined, and  $k$  calculated, a new matrix  $U^*$  can be constructed from  $U$ , neglecting  $u_i$  vector for  $i > k$ . Hence,  $U^* = [u^1 \ u^2 \ \dots \ u^k]$  is a  $d \times k$  matrix. Similarly to  $U$ , we can be interested in calculating the projection of a vector  $x$  in the subspace of  $U^*$ . This is given by the expression  $z^* = U^{*T}x$ , or  $Z^* = U^{*T}X$  in the case of dealing with matrices. Here,  $Z^*$  would be a  $k \times n_s$  matrix.

An important final consideration is how to carry out the backward mapping of a  $z^*$  vector in the reduced subspace. Given that in the full dimension case,  $x$  would be calculated from a vector  $z$  as  $x = Uz$ , the following expression is proposed:

$$x = Uz \approx U^*z^* \quad (10)$$

Or, in the case of matrices:

$$X = UZ \approx U^*Z^* \quad (11)$$

This is not an exact expression, and an error is produced, equal to  $X - U^*Z^*$ . The closer  $k$  is to  $d$ , the smaller this error will be, so the closer the tolerance  $\varepsilon$  is to 0, the closer the error will be to 0.

A final remark that needs to be addressed is the fact that the computational cost of diagonalising  $C$  and  $G$  are equivalent, even though their sizes are  $d \times d$  and  $n_s \times n_s$  respectively. Computing  $u^i$  and  $v^i$  for  $i = 1, \dots, d$  is equivalent, and this is in fact at the basis of the kPCA.

From the expression  $Cu^i = \lambda_i u^i$ ,  $u^i$  can be defined as:

$$u^i = \frac{1}{\lambda_i} \sum_{l=1}^{n_s} (x^l x^{lT}) u^i = \sum_{l=1}^{n_s} \frac{1}{\lambda_i} (x^{lT} u^i) x^l = \sum_{l=1}^{n_s} [B]_{li} x^l \quad (12)$$

With:

$$[B]_{li} = \frac{1}{\lambda_i} x^{lT} u^i \quad (13)$$

For  $i = 1, \dots, d$  and  $l = 1, \dots, n_s$ . This matrix  $B$  coincides with the  $d$  first columns of matrix  $V$ , such that  $[B]_{li} = [v^i]_l$ . This also implies that  $U^* = XV^*$ . Therefore, from the expression  $z^* = U^{*T}x$ , and for  $i = 1, \dots, k$  we obtain:

$$[z^{j*}]_i = (u^i)^T x^j = \left( \sum_{l=1}^{n_s} [B]_{li} x^l \right)^T x^j = \sum_{l=1}^{n_s} [B]_{li} ((x^l)^T x^j) = [V^{*T}G]_{ij} \quad (14)$$

Or, in matrix notation:

$$Z^* = V^{*T}G \quad (15)$$

Where  $V^*$  is a  $k \times n_s$  matrix. Then, in a similar fashion as (11), the backward mapping can be rewritten as:

$$X \approx U^*Z^* = XV^*Z^* \quad (16)$$

Which introduces a difficulty compared to (11), as the unknown  $X$  appears in both sides of the equation. This is, in fact, one of the principal steps to overcome in the backward mapping of the kPCA.

## 2.2 kPCA

The kernel Principal Component Analysis, or kPCA, is a dimensionality reduction technique derived from the PCA. Its objective is the same, but it aims at overcoming one of the main limitations of PCA, which is that many variables are actually characterized by a non-linear manifold. In these cases, PCA and POD cannot give a proper answer, as they are based on a linear transformation.

When this happens, the idea behind the kPCA is to apply a transformation  $\Phi$  from  $\mathbb{R}^d$  to  $\mathbb{R}^D$ , where  $D$  is a much larger dimension than  $d$ . This transformation must provide an “untangling” of the data, so that, in this new dimension  $D$ , a linear low-dimension manifold can be calculated through the traditional PCA method.

Therefore, the matrix of variables is, in this case:

$$\underline{X} = [\Phi(x^1) \quad \dots \quad \Phi(x^{n_s})] = [\underline{x}^1 \quad \dots \quad \underline{x}^{n_s}] \quad (17)$$

Which is a  $D \times n_s$  matrix. Given the transformation  $\Phi$ ,  $\underline{X}$  is calculated, and from this point, the PCA is applied. However, some difficulties arise from this method: first, an adequate function  $\Phi$  must be chosen; then, the dimension  $D$  could be so large that the standard application of the PCA could require an unaffordable computational effort.

The first difficulty will be addressed in the following sections. The second one is easily dealt with by proceeding with the  $G$  matrix, instead of using the  $C$  matrix. This is because, in this case,  $C$  would be of size  $D \times D$ , while  $G$  would be of size  $n_s \times n_s$ . Usually, in these cases,  $n_s$  is smaller than  $D$ , so the computational effort required to calculate  $G$  becomes smaller than the one required to calculate  $C$ , which is not the case in the PCA.

### 2.2.1. The kernel

Instead of defining a function  $\Phi(\cdot)$ , such that  $[\underline{G}]_{ij} = \Phi(x^i)^T \Phi(x^j)$ , the idea behind the kernel is to introduce a bivariate symmetric form  $\kappa(\cdot, \cdot)$  that defines  $\underline{G}$  as  $[\underline{G}]_{ij} = \kappa(x^i, x^j)$ . An example of this type of function is the Gaussian kernel:

$$\kappa(x^i, x^j) = \exp\left(-\beta \|x^i - x^j\|^2\right) \quad (18)$$

However, many different kinds of kernels exist, the idea being to select the one that provides the best results which is the lowest final dimension.

An important thing to take into account is that, before using the PCA, the matrix  $\underline{G}$  needs to be centered, which is not guaranteed by a random transformation  $\kappa(x^i, x^j)$ . Once this transformation is known, though, it's easy to center the matrix through the expression:

$$\begin{aligned}
[\tilde{\underline{G}}]_{ij} &= \kappa(x^i, x^j) - \frac{1}{n_s} \sum_{l=1}^{n_s} \kappa(x^i, x^l) - \frac{1}{n_s} \sum_{m=1}^{n_s} \kappa(x^m, x^j) \\
&+ \frac{1}{n_s^2} \sum_{l=1}^{n_s} \sum_{m=1}^{n_s} \kappa(x^l, x^m)
\end{aligned} \tag{19}$$

Which can be rewritten as:

$$\tilde{\underline{G}} = \underline{G} - \frac{1}{n_s} \underline{G} \mathbb{I}_{(n_s \times n_s)} - \frac{1}{n_s} \mathbb{I}_{(n_s \times n_s)} \underline{G} + \frac{1}{n_s^2} \mathbb{I}_{(n_s \times n_s)} \underline{G} \mathbb{I}_{(n_s \times n_s)} \tag{20}$$

Where  $\mathbb{I}_{(n_s \times n_s)}$  is an  $n_s \times n_s$  matrix composed solely of ones. This last expression provides the centred matrix after a simple algebraic manipulation of matrix  $\underline{G}$ .

### 2.2.2. Dimensionality reduction in the kPCA.

Once the matrix  $\tilde{\underline{G}}$  has been calculated, the idea behind the kPCA is to proceed as the POD would. Therefore, this matrix is to be diagonalised as follows:

$$\tilde{\underline{G}} = \underline{V} \tilde{\underline{\Lambda}} \underline{V}^T \tag{21}$$

With the matrix  $\tilde{\underline{\Lambda}}$  containing its eigenvalues,  $\underline{\lambda}_1 \geq \underline{\lambda}_2 \geq \dots \geq \underline{\lambda}_{n_s} \geq 0$  in decreasing order, and matrix  $\underline{V}$  its associated eigenvectors.

Following the procedure explained in the previous sections, a number of dimensions  $k$  is defined, such that they collect a significant amount of the total variability, and then the matrix  $\underline{V}^*$  is defined as a  $n_s \times k$  matrix, comprised of the first  $k$  columns of matrix  $\underline{V}$ . Hence, the  $k \times n_s$  matrix  $Z^*$  of samples in the reduced spaces is defined as:

$$Z^* = \underline{V}^{*T} \tilde{\underline{G}} \tag{22}$$

Which is equivalent to (15). In the case of wanting to map a new element,  $x^{new} \in \mathbb{R}^d$ , that does not belong to the training set, we proceed by defining a vector  $\underline{g}^{new}$  such that:

$$[\underline{g}^{new}]_i = \kappa(x^i, x^{new}) \tag{23}$$

Which will be centered through the expression:

$$\begin{aligned} \underline{\tilde{g}}^{new} = \underline{g}^{new} - \left( \frac{1}{n_s} \mathbb{I}_{(n_s)}^T \underline{g}^{new} \right) \mathbb{I}_{(n_s)} - \frac{1}{n_s} \mathbb{I}_{(n_s \times n_s)} \underline{g}^{new} \\ + \left( \frac{1}{n_s^2} \mathbb{I}_{(n_s)}^T \underline{G} \mathbb{I}_{(n_s)} \right) \mathbb{I}_{(n_s)} \end{aligned} \quad (24)$$

Where  $\mathbb{I}_{(n_s)}$  is a vector containing  $n_s$  elements, all equal to one. Finally, the mapping of  $x^{new}$  into the reduced space is given by:

$$z^{new} = \underline{V}^{*T} \underline{\tilde{g}}^{new} \quad (25)$$

### 2.2.3. Backward mapping.

At this point, the question of the backward mapping rises once again. As mentioned when discussing the backward mapping of the PCA, having only  $\underline{V}^*$  does not allow us to obtain the pre-image  $x^*$  of an element  $z^*$ , so a different strategy needs to be followed to obtain it.

The idea behind the backward mapping in the kPCA is to compute the element  $x^*$  as a linear combination of the elements of the training set:

$$x^* = \sum_{i=1}^{n_s} w_i x^i \quad (26)$$

Then, the unknowns become the different  $w_i$  weights. The criteria followed to calculate them is to take into account the distance between the element of interest  $z^*$ , and the already mapped elements  $z^i$ . This distance is easily computable as  $d^i = \|z^* - z^i\|$ , and the weights will have to be inversely proportional to the value of the distance (the larger the distance, the smaller the weight). Different relationships between distances and weight may provide similar results, but also some discrepancies. Some examples would be:  $w^i = 1/d^i$ ,  $w^i = 1/d^{i^2}$ , or  $w^i = \exp(-d^i)$ .

A more precise method would be the following one: given the vector of unknown weights  $w = [w_1 \ \cdots \ w_{n_s}]^T$ , vector  $\underline{g}^*$  is computed as:



$$[\underline{g}^*]_i = \kappa(x^i, x^*) = \kappa\left(x^i, \sum_{j=1}^{n_s} w_j x^j\right) \quad (27)$$

And then it is centred as in equation (24), with  $\underline{g}^{new} = \underline{g}^*$ , to obtain  $\tilde{\underline{g}}^*$ .

Then, given the discrepancy function  $\mathcal{J}(w)$ , as defined by:

$$\mathcal{J}(w) = \left\| z^* - \underline{V}^{*T} \tilde{\underline{g}}^*(w) \right\|^2 \quad (28)$$

The weights are calculated by minimising  $\mathcal{J}(w)$ , so that:

$$w = \arg \min_{w \in \mathbb{R}^{n_s}} \mathcal{J}(w) \quad (29)$$

### 3 SURROGATE MODEL CONSTRUCTION THROUGH PCA

As previously said, one of the main objectives of implementing dimensionality reduction methods in this project is the creation of surrogate models, in order to swiftly tackle complex problems that would require big amounts of computational power in less time. In this section, we will explain how this will be done through the results obtained by applying the PCA method.

In this case, we are no longer dealing exclusively with sets of data. Now, the nature of the problem needs to be considered, so, in this case, it is especially important to note that we are dealing with a FEM model where the material behaviour is supposed to be linear elasticity. Hence, the formulation presented in this section can only be applied to this kind of problems.

The main objective of generating a surrogate model is to reduce the amount of variables that are being computed on the problem. This will be done through the results of applying the PCA to a set of data, that will then be exported to the actual equations that govern the problem.

#### 3.1 Linear elasticity in FEM

We will start by defining the formulation of the actual mechanical problem, without getting into unnecessary details.

Considering a FEM model computed according to linear elasticity, after having defined the geometry, the grid, the elements, the form functions, and the applied forces, the problem will be reduced to the following expression:

$$f = Kx \tag{30}$$

Where:

- $f$  is the vector containing the forces applied at the nodes.
- $K$  is the stiffness matrix calculated from the geometric and material properties of the model.
- $x$  is the displacement vector, which is the unknown that needs to be assessed.

In order to construct the surrogate model, both  $f$  and  $K$  will have to be calculated. The idea behind this is to transform them into smaller matrices, so that the problem that will be computed has less dimensions, and it's less computationally expensive to solve.

Calculating  $f$  and  $K$  will be done, in this case, through the same software used to carry out the necessary simulations, which is NASTRAN. Generating  $f$  and  $K$  is usually a general feature of any FEM commercial software, though some work might need to be done to generate them in the proper format.

### 3.2 Application of PCA results

Once  $f$  and  $K$  have been calculated, we need to apply the results obtained from applying the PCA.

First of all, it is important to note that, in order to apply the PCA, a large set of data needs to be available. As said in the introduction, this is a case of *online* vs. *offline* work: A large amount of simulations will have to be carried out in advance in order to generate the set of data to apply the PCA. With these results, a surrogate model will be generated, which will decrease the computational time of future simulations. Hence, the need for this kind of models and the possibility of carrying out this large amount of simulations beforehand need to be adequately assessed.

In this case, we will suppose that the results of a PCA analysis are available. This means that, from a large set of results  $[x^1 \ x^2 \ \dots \ x^{n_s}]$  with  $x^1, x^2, \dots, x^{n_s} \in \mathbb{R}^d$  representing the displacements obtained from different parameter combinations, a dimensionality reduction from  $d$  to  $k$  has been carried out in order to obtain a matrix  $U^*$  that transforms a  $x^*$  vector into a  $z^*$ , such that  $x^* = U^* z^*$  (as given in *equation (11)*).

This transformation is applicable to any vector  $x$  inside that respects the range of parameters tested in  $[x^1 \ x^2 \ \dots \ x^{n_s}]$ . Hence, the main interest of carrying out the PCA is obtaining the matrix  $U^*$ , and to be able to apply *equation (11)*.

### 3.3 Surrogate model formulation

The construction of the surrogate model is based in the expressions presented before,  $f = Kx$  and  $x^* = U^*z^*$ . The first thing to note is that the latter is only valid for vectors  $x'$  that have been centred. As said in section 2.1.1, constructing matrix  $C$  requires the matrix of data  $X$  used in the PCA to have its mean equal to zero. Considering  $x^{mean}$  to be the mean value of all the data sets contained in  $X$ ,  $x$  and  $x^*$  are related as follows:

$$x = x^* + x^{mean} \quad (31)$$

If we apply this to  $f = Kx$ , we obtain:

$$f = K(x^* + x^{mean}) \quad (32)$$

And from  $x^* = U^*z^*$ :

$$f = K(U^*z^* + x^{mean}) \quad (33)$$

Be leaving the term with  $z^*$  on the right, and multiplying the expression by  $U^{*T}$  on the left, we obtain:

$$U^{*T}(f - Kx^{mean}) = U^{*T}KU^*z^* \quad (34)$$

Which becomes the reduced problem that will be solved, with  $z^*$  as its unknown. Hence, what first was a problem with  $d$  unknowns, has become a problem with  $k$  unknowns. Instead of a  $d \times d$  matrix  $K$ , a  $k \times k$  matrix  $U^{*T}KU^*$  is present, and instead of a  $d$  size vector  $f$ , we have a vector  $U^{*T}(f - Kx^{mean})$  of size  $k$ .

## 4 DATA GENERATION

Up to this point, we've only presented the formulation of the POD, the PCA, and the kPCA methods, and their role in the generation of a surrogate model for linear elasticity. However, in order to apply them, a set of results is necessary.

In this sections, these results will be computed in the form of node displacements through a FEM model. As said before, to solve the finite element problem, we will use the NASTRAN software, which is commonly used in the LaCàN.

First of all, we will describe the NASTRAN model that will be used, which represents the basic structure of a vehicle. The geometry, and its implementation in NASTRAN, will be described in detail, because it's here where all the parameters that characterize the model will be defined (such as geometric properties, material properties...). This is important because in order to apply the POD, the PCA, and the kPCA methods, several sets of results, coming from models where certain parameters vary between each other, are necessary. Hence, several possibilities for these parameters will be presented in this section, and the final choice will be justified later.

Once the model itself has been defied, we will define the parameters that will be varied between simulations. Indeed, each simulation in the data set used in the dimensionality reduction techniques comes from a set of parameters that differs between simulations. Hence, we will discuss the different possibilities and chose the best option.

### 4.1 NASTRAN model

When considering the NASTRAN model, several things will have to be defined. First of all, the different considered geometries and the final choice will be presented. Then, the loads and the stress state of the structure will be discussed, as well as certain computational aspects. Finally, any particularity introduced by the fact that we will be working with a FEM model will be commented. It is, in fact, in the finite element

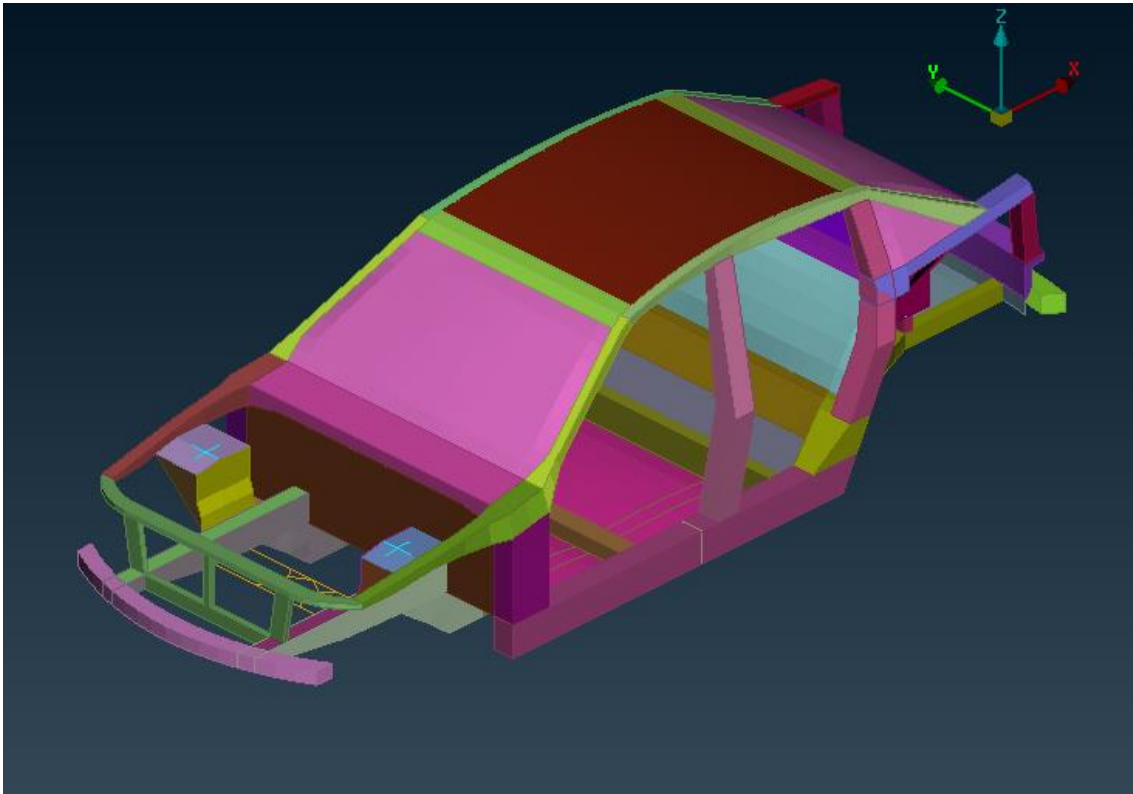
characterisation of the problem, that the parameters that can be subject to variation are introduced.

#### 4.1.1. Geometrical model

In order to carry out the necessary calculations, a geometrical model, that then will be divided into finite elements, is necessary. Two possible options were proposed from existing models in the LaCàN, the first one being a simple representation of the overall shape of the body in white state of a car, and the second one being a more precise characterization of this same state, with more elements present. It was decided from the beginning that the second option would be chosen for the following reasons: First, it would provide better results than the simpler one, as it was a faithful representation of the structure of interest; second, the computing time would also be closer to the real one, which, taking into account that a large amount of simulations were going to be carried out, would indicate how well optimised the code was, and if some improvements were needed; third, a larger amount of elements increased the amount of possible parameters that could be subject of analysis; and fourth, the development of the model needed to be general enough to be adapted to any geometry, so the steps to be followed in order to generate the necessary data were not dependent on the complexity of the geometry. Hence, for these reasons, the second and more complex model was chosen.

Figure 4.1 shows the model in question. As displayed in the image, the model represents a generic car structure, where structural elements are represented. Usually, elements will be connected to each other by means of welding, riveting, clinching, etc. However, these details are not the subject of study of this project; hence, they are not represented.

The metallic elements of the model are made of steel sheets, each one with their own width. It should also be noted that the structure is symmetrical with respect to its longitudinal, vertical plane.



*Figure 4.1: Geometry of the car model*

#### 4.1.2. Loads

Different kinds of loading states could be considered when carrying out the analysis, especially if we consider how important dynamic analysis are in this kind of structures, whether it be vibration analysis or crash analysis.

In this case, however, time dependant solutions are considered unnecessarily complex: first, because our main objective is to obtain a set of results composed of displacements, which can be obtained from a simpler, static analysis; and second, because the rheological behaviour of the materials would have to be considered, which would introduce non-linearity in the actual material behaviour. For these reasons, a static analysis is performed.

The loading state that will be tested in all simulations will be one of torsion. It has been chosen for practical reasons, since other projects in the LaCàN are already studying these kind of loads, and hence synergies between projects could be of interest.

In order to represent this, forces will be applied at both longitudinal extremes of the structure. At each end, two forces will be applied on two different nodes in the same transversal plane. This forces will have the same value, but different direction, therefore generating a moment. The moment generated in each extreme will be of the same value, but of opposite directions; hence, the structure will be globally in equilibrium, but a torsion loading state will be induced on the structure. A representation of this loads is represented in Figure 4.2.

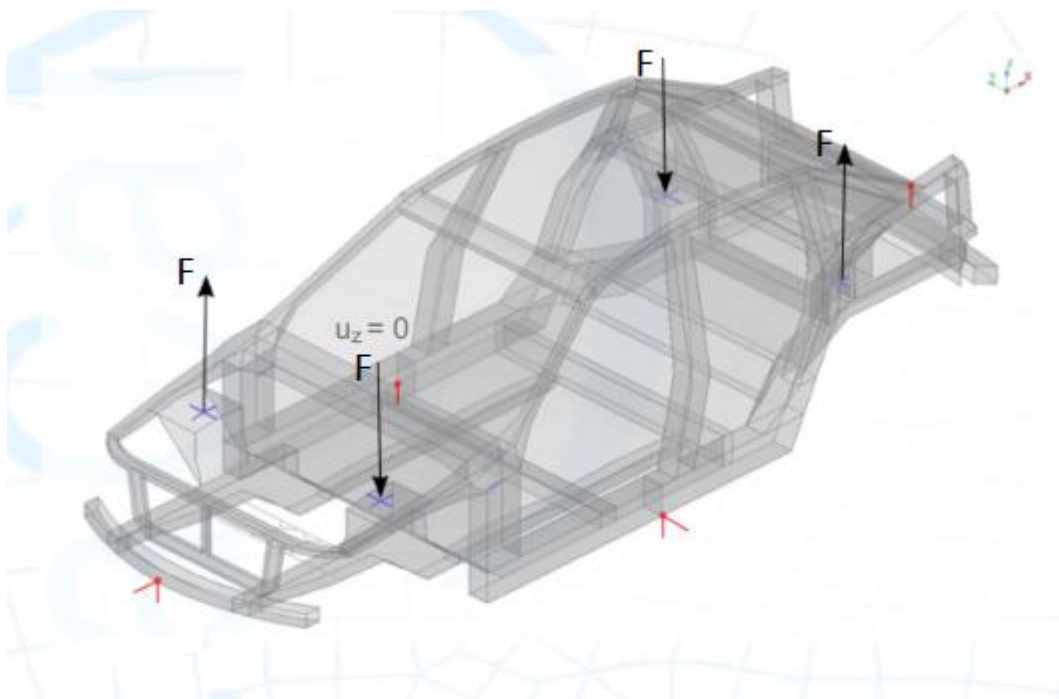


Figure 4.2: Representation of the forces applied to the model.

It's important to notice that the loads applied will not vary between simulations: Both their value, and their application point will remain constant all throughout the set of simulations, because we are interested in the effect of varying geometric and material characteristics of the model.

A final important remark regarding the degrees of freedom of the structure needs to be addressed. As stated before, we are simulating the body in white state of a car, and, contrary to most civil engineering structures, no clear point or surface has clear restraints on its displacement and rotation. In this case, the concept of inertia relief is introduced: when dealing with unconstrained structures, NASTRAN provides this option in order to carry out static analysis on them. The idea behind it is to introduce constant accelerations, in a rigid body state, to balance externally applied forces. It is a really



useful tool used in car, aeronautic, and other engineering branches where dynamic analysis is common. In this case, it will be performed directly by NASTRAN. Further remarks about this technique will be provided when implementing the surrogate model.

### 4.1.3. FEM IN NASTRAN

Once the geometry and the loads have been defined, the next step is to translate this to a finite element model, where the actual computing will be carried out.

The main elements to consider are nodes and elements, with the latter introducing many other variables that will be discussed in this section.

First of all, nodes define the main geometry of the structure, as they are an essential component of the definition of the elements. In NASTRAN, they are defined as shown in Table 4.1:



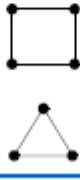
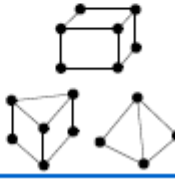

Table 4.1: Grid definition in NASTRAN [7]

1	2	3	4	5	6	7	8	9	10
GRID	ID	CP	X1	X2	X3	CD	PS	SEID	
Field	Contents					Value			
ID	Grid point identification number.					0 < Integer < 100000000			
CP	Identification number of coordinate system in which the location of the grid point is defined.					Integer ≥ 0 or blank			
X1, X2, X3	Location of the grid point in coordinate system CP.					Real; Default = 0.0			
CD	Identification number of coordinate system in which the displacements, degrees of freedom, constraints, and solution vectors are defined at the grid point.					Integer ≥ -1 or blank			
PS	Permanent single-point constraints associated with the grid point.					Any of the Integers 1 through 6 with no embedded blanks, or blank			
SEID	Superelement identification number.					Integer ≥ 0 ; Default = 0			

In our case, the set of grid points is composed of 3856 elements. The results will be obtained as displacements in each one of these points, with six degrees of freedom associated to each one, representing three displacement coordinates (one for each axis), and three rotations (one around each axis).

Regarding the elements, many could be considered, and NASTRAN offers a wide range to choose from, as shown in Table 4.2:

Table 4.2: FEM elements in NASTRAN [7]

Category	Spring Elements	Line Elements	Surface Elements	Solid Elements	Rigid Elements
Physical Behavior	Simple Spring	Rod, Bar, Beam	Membrane, Thin Plate	Thick Plate, Brick	Rigid Bar
MSC Nastran Element Name	CELAS2*	CONROD* CROD CBAR	CQUAD4 CTRIA3	CHEXA CPENTA CTETRA	RBE2*
Associated Property Entry	None Required	PROD PBAR	PSHELL	PSOLID	None Required
					
* A separate property entry is not required. It is built directly into the element entry.					

As shown in Table 4.2, each element is associated to a certain structural element (or physical behaviour). The main reasoning behind these categories is the hierarchy of dimensions behind structural elements, given that, for example, beams can be studied as 1D elements, and membranes as 2D elements. Hence, simplifications that reduce computing time without compromising the results can be introduced. In our case, the model is composed, mainly, of 2D elements, specially CQUAD4 elements, though some CTRIA3 elements are also present. A few 1D elements, such as CBAR, are also part of the model, but they are a minority, and we will focus on the 2D ones. This kind of combination of elements does not pose any kind of problem to the computation.

Regarding surface elements, CQUAD4 elements are defined as shown in Table 4.3, while CTRIA3 elements are defined as shown in Table 4.4:

Table 4.3: CQUAD4 elements definition [7]

1	2	3	4	5	6	7	8	9	10
CQUAD4	EID	PID	G1	G2	G3	G4	THETA or MCID	ZOFFS	
			T1	T2	T3	T4			
Field	Contents		Value						
EID	Element identification number.		Integer > 0.						
PID	Property identification number of a PSHELL or PCOMP entry.		Integer > 0; Default is EID.						
Gi	Grid point identification numbers of connection points.		Integers > 0, all unique.						
THETA	Material property orientation angle in degrees.		Real; Default = 0.0.						
MCID	Material coordinate system identification number. The x-axis of the material coordinate system is determined by projecting the x-axis of the MCID coordinate system (defined by the CORDij entry or zero for the basic coordinate system) onto the surface of the element.		Integer ≥ 0; if blank, then THETA = 0.0 is assumed.						
ZOFFS	Offset from the surface of grid points to the element reference plane.		Real.						
Ti	Membrane thickness of element at grid points G1 through G4.		Real ≥ 0.0 or blank, not all zero.						

Table 4.4: CTRIA3 elements definition [7]

1	2	3	4	5	6	7	8	9	10
CTRIA 3	EID	PID	G1	G2	G3	THETA or MCID	ZOFFS		
			T1	T2	T3				
Field	Contents		Value						
EID	Element identification number.		Integer > 0						
PID	Property identification number of a PSHELL or PCOMP entry.		Integer > 0; Default is EID						
Gi	Grid point identification numbers of connection points.		Integers > 0, all unique						
THE TA	Material property orientation angle in degrees.		Real; Default = 0.0						
MCI D	Material coordinate system identification number. The x-axis of the material coordinate system is determined by projecting the x-axis of the MCID coordinate system (defined by the CORDij entry or zero for the basic coordinate system) onto the surface of the element.		Integer ≥ 0; if blank, then THETA = 0.0 is assumed						
ZOF FS	Offset from the surface of grid points to the element reference plane.		Real						
Ti	Membrane thickness of element at grid points G1, G2, and G3.		Real ≥ 0.0 or blank, not all zero						

From these tables, the main point that needs to be developed is the third property, “PID”. In our case, it is defined as a PSHELL property, which as stated by MSC Software guide on NASTRAN, “defines the membrane, bending, transverse shear, and coupling properties of thin plate and shell elements”. These properties are introduced into the code as shown in Table 4.5:

Table 4.5: PSHELL property definition [7]

1	2	3	4	5	6	7	8	9	10
PSHELL	PID	MID1	T	MID2	12I/T <sup>3</sup>	MID3	TS/T	NSM	
	Z1	Z2	MID4						

Field	Contents	Value
PID	Property identification number.	Integer > 0
MID1	Material identification number for membrane.	Integer ≥ 0 or blank
T	Default value for the membrane thickness.	Real
MID2	Material identification number for bending.	Integer ≥ -1 or blank
12I/T <sup>3</sup>	Bending stiffness parameter.	Real > 0.0; Default = 1.0
MID3	Material identification number for transverse shear.	Integer > 0 or blank; must be blank unless MID2 > 0
TS/T	Transverse shear thickness divided by the membrane thickness.	Real > 0.0; Default = .833333
NSM	Nonstructural mass per unit area.	Real
Z1, Z2	Fiber distances for stress calculations. The positive direction is determined by the right-hand rule, and the order in which the grid points are listed on the connection entry.	Real or blank; Default = ±t/2
MID4	Material identification number for membrane-bending coupling.	Integer > 0 or blank, must be blank unless MID1 > 0 and MID2 > 0, may not equal MID1 or MID2

Two final properties need to be addressed from this table. First of all, “T”, which represents the material thickness, and is one of the main candidates to being varied in order to produce the set of results. Then, MID1, which associates a material to the element. A generic example of concrete is shown in Table 4.6, where it can be seen that an ID is associated to each material, as well of a set of typical parameter (elastic modulus, shear modulus, Poisson modulus...). In our case, a possibility would be to vary the elastic modulus.

Table 4.6: Material definition [7]

MAT1	MID	E	G	NU	RHO	A	TREF	GE	
MAT1	7	30.E6		0.3					

## 4.2 Definition of parameters and data generation

Once the geometry of the model has been presented, as well as the loading state that will be simulated, and the implementation of these decision on NASTRAN, we will

discuss the parameters that will be changed in each simulation, in order to obtain the data set that we need to perform dimensionality reduction methods. We have pointed out some of these parameters as we described the implementation of finite elements on NASTRAN, but the final decision needs to be explained in more detail.

The possible parameters that can be varied are those that represent material and geometrical properties. However, it needs to be pointed out first that it is not possible to vary the actual mesh due to several reasons: First of all, we made clear that we won't vary the point of application of any of the forces, as the stress state of each simulation would be different. These stresses represent forces imposed onto the model, so it's not really logical to vary them, since the objective of developing this project is to facilitate the decision making process. In finite element models, forces are applied on nodes, so the nodes where forces are applied need to be constant throughout the simulations. Furthermore, the results are obtained as displacements and rotations of nodes, and, in order to be able to compare these results between different simulations, they need to refer to the same point in space. For these reasons, no variation of the mesh is considered.

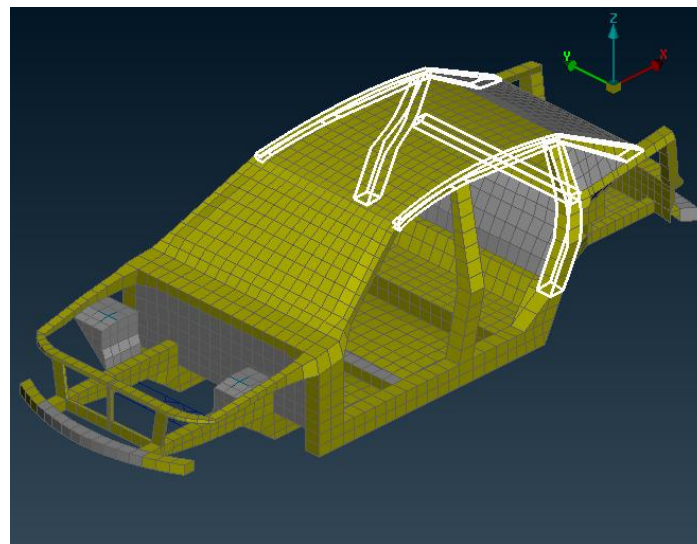
Regarding geometrical properties, this leaves us with only those that can be modified after the mesh is defined. For example, the length of certain elements could not be changed. This brings us to the definition of PSHELL properties; as it can be seen in Table 4.5, the main geometric parameter that can be considered to vary in this case is the thickness of each element, which is represented by the letter  $T$ . This seems a plausible parameter, first, because it does not interfere with the grid definition, but also because the structural parts of the model are composed of thin steel plates, whose thickness can be chosen in order to optimize certain properties, as several possible thicknesses are offered by the industry. Hence, since some decision will have to be taken regarding this parameter for each structural part, it is considered the best parameter for this project.

Another possibility would be the variation of certain material properties of the model, but this idea was finally discarded. No large variety of materials were used in the model, and the only parameter that offered interesting possibilities was the elastic modulus. Still, only two parameters could be varied, and even then, a smaller variety of actual possible values could have been considered. This contrasts with the parameter

described previously, as each structural element had its own thickness, and the industry offered a wider range of possibilities.

It was decided that a certain number of structural parts would have its thickness vary during the simulations, that a certain amount of values for each thickness would be considered, and that, among these values, all possible combinations would be considered. This last point (considering all possible combinations) means that an excessive number of elements or thickness values could exponentially increase the computing time required to obtain all the results. Therefore, it was considered that only the thickness of three elements would vary.

Having taken this decision, it was decided that two out of the three structural elements chosen to vary would have a symmetric counterpart. These were the top lateral element of the car (PSHELL value of 8 and 1008) and its connection to the base and the boot (PSHELL value of 20 and 1020). The last element connects both the other two were the cabin joins the boot (PSHELL value of 29). Figure 4.3 shows these structural elements in the model:



*Figure 4.3: Elements 8, 1008, 20, 1020, and 29 in the model.*

## 5 ANALYSIS OF POD AND kPCA RESULTS

In this section, we will describe the results obtained from implementation of the POD and the kPCA techniques over a large set of data, obtained from NASTRAN simulations based on the model described in the previous sections.

Both the POD and the kPCA will be implemented in five different cases, with five different sets of data. The main objective of these simulations, besides assessing the overall performance of the model, is to study the effects of different computational model parameters. In this case, when we refer these parameters, we are not talking about things such as the thickness of the steel plates or the elastic modulus of the material, since describing the effect of the variability of these two examples could be one of the inherent objectives of dimensionality reduction methods. What we mean, by computational model parameters, are those parameters inherent to the numerical model, that need to be adequately chosen so that the computed results are precise.

The parameters that will be tested between the different POD and kPCA implementations will be those describing the sample of data. Since one of the main requirements of carrying out these dimensionality reduction techniques is to generate a large amount of data through simulations, we will focus on assessing the effect of varying it. Hence, five different implementations will be carried out, each one considering a different amount of simulations.

In order to be able to compare the different results, each case will have to be consistent with one another. The implementations of the POD and the kPCA methods will be as follows:

- For all simulation and all considered elements (8 and 1008, 20 and 1020, and 29), a single range of thicknesses will be chosen, which will be from 0,1 mm to 2,1 mm. This range has been chosen, first of all, because average values of plate thickness of the considered structural elements are close to 1 mm, and because significant non-linear effects will appear when values are closer to zero. The

range of values was finally extended up to 2,1 mm to have the average value of 1 mm close to the centre of the considered range.

- In order to decide the amount of values that will be given to the thickness of the plates in each case, it has been decided to choose a constant increment between values in each case. For example, with an increment of 0,1, the values chosen would be:

[0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1,8 1,9 2,0 2,1]

- The implementation of each different case of POD and kPCA will be based in thickness values characterized by different increments. In each case, all elements (8 and 1008, 20 and 1020, and 29) will take the same possible amount of values. The increment in the first implementation will be of 0.2; in the second one, of 0.4; in the third one, of 0.5; in the fourth one, of 1.0; and in the fifth one, of 2.0. This means that 11 values of thicknesses will be considered in the first implementation, 6 in the second one, 5 in the third one, 3 in the fourth one, and 2 in the fifth one. This is summed up in Table 5.2. This increments have been chosen because they guarantee that the limits of the range of thicknesses (0.1 and 2.1) are both included in all cases, and because they represent five different densities of values in this range.
- In order to define the number of simulations carried out for each implementation of these methods, it has been decided to consider all possible combinations of thicknesses between elements. Since we are considering three different structural elements, the amount of simulations will be of  $n^3$ , where  $n$  is the number of possible thicknesses considered for each one of the three elements. This is also represented in Table 5.2.

In order to illustrate this, the example of the fourth implementation will be explained. As previously said, we will take an increment of 1.0 in this case, and considering a range from 0.1 to 2.1, the possible thickness values for each structural element are three:

[0.1 1.1 2.1]

Since three thicknesses values are considered in this case, a total of  $n^3 = 3^3 = 27$  combinations of values will be considered. Each combination of this parameters will be



used to carry out a simulation, that will result in a different displacement vector that will be used in the POD and the kPCA. The combination of parameters used in each of the 27 simulations are represented in Table 5.1.

TABLE 5.1: Parameters associated to all 27 simulations for the fourth case of POD and kPCA implementation.

Simulation 1	(0.1 0.1 0.1)	Simulation 15	(1.1 1.1 2.1)
Simulation 2	(0.1 0.1 1.1)	Simulation 16	(1.1 2.1 0.1)
Simulation 3	(0.1 0.1 2.1)	Simulation 17	(1.1 2.1 1.1)
Simulation 4	(0.1 1.1 0.1)	Simulation 18	(1.1 2.1 2.1)
Simulation 5	(0.1 1.1 1.1)	Simulation 19	(2.1 0.1 0.1)
Simulation 6	(0.1 1.1 2.1)	Simulation 20	(2.1 0.1 1.1)
Simulation 7	(0.1 2.1 0.1)	Simulation 21	(2.1 0.1 2.1)
Simulation 8	(0.1 2.1 1.1)	Simulation 22	(2.1 1.1 0.1)
Simulation 9	(0.1 2.1 2.1)	Simulation 23	(2.1 1.1 1.1)
Simulation 10	(1.1 0.1 0.1)	Simulation 24	(2.1 1.1 2.1)
Simulation 11	(1.1 0.1 1.1)	Simulation 25	(2.1 2.1 0.1)
Simulation 12	(1.1 0.1 2.1)	Simulation 26	(2.1 2.1 1.1)
Simulation 13	(1.1 1.1 0.1)	Simulation 27	(2.1 2.1 2.1)
Simulation 14	(1.1 1.1 1.1)		

Even though we wanted to test the effect of an increment of 0,1, it has been deemed too computationally expensive: by choosing an increment of 0,1, a total amount of  $21^3 = 9261$  simulations would have been needed, which was estimated to take close to 5 days of available computing time. Not only that, but performing the dimensionality reduction through the kPCA method requires, as said before, a set of matrix multiplications and diagonalisations of  $n^3$  complexity, where  $n$  is the size of the matrices. In this case, the size of the matrix would be the number of simulations, so we would be dealing with  $9261 \times 9261$  matrices. For reference, other tries with  $1000 \times 1000$  matrices have required more than two hours to finish, so it has been decided to choose an increment of 0,2 as the smallest increment considered for the simulations.

All these parameters that characterize each simulation have been summed up in Table 5.2:

*Table 5.2: Summary of parameters used in each implementation case of the POD and kPCA.*

Implementation case	Increment	Number of thickness values	Number of simulations	Thickness values
1	0,2	11	1331	[ 0,1 0,3 0,5 0,7 0,9 1,1 1,3 1,5 1,7 1,9 2,1 ]
2	0,4	6	216	[ 0,1 0,5 0,9 1,3 1,7 2,1 ]
3	0,5	5	125	[0,1 0,6 1,1 1,6 2,1]
4	1,0	3	27	[0,1 1,1 2,1]
5	2,0	2	8	[0,1 2,1]

Another thing that needs to be addressed is the criterion followed to define the tolerance used to carry out the dimensionality reduction (*equation (9)*). As said before, the idea behind this analysis is to reduce the dimension of the problem, so that it can be easily recalculated for other values of certain inputs. In our case, since we are using the thicknesses of the three materials that we are considering, it is logical to think that the problem can be simplified to a three dimensions problem.

Following this reasoning, it has been decided to define a tolerance such that the three first eigenvalues in the kPCA method, representing variabilities, are considered. The kPCA has been chosen, because it will offer more precise results than the POD. From these three values, their overall weight on the total variability will be extrapolated to the POD method, so that we can properly compare a linear approach and a non-linear approach to the problem. What we will obtain is that, for a certain tolerance that reduces the dimension of the problem to three in the case of the kPCA, the POD will reduce this dimension to a certain number higher than 3. The closer it is to three, the more effective a linear method is.

In order to do so, a different approach from the one proposed in the previous section needs to be considered. The idea that we propose is the following:

First of all, the POD and the kPCA methods will have to be carried out until the diagonalisation of  $G$  and  $\tilde{G}$  respectively (considering the notation of section 2). These matrices contain the eigenvalues that represent how the variability of the problem is distributed in independent variables. Therefore, since we are interested in representing the kPCA with three dimension (as this is the number of parameters that we made vary), the first three eigenvalues will be summed, and the tolerance will be calculated as the quotient of this value over the sum of all the eigenvalues that form matrix  $\tilde{G}$ , such that:

$$\text{tol} = \frac{\sum_{i=1}^3 \lambda_i^{kPCA}}{\sum_{j=1}^{n_s} \lambda_j^{kPCA}} \quad (35)$$

Once this tolerance has been defined, the same method proposed earlier to calculate the dimensionality reduction through the POD will be used. Through this method, we will be able to compare the dimensionality reduction provided by the kPCA and the POD for the same reliability.

## 5.1 Implementation case 1

The first set of results is obtained by following the procedure described previously, with the parameters shown in Table 5.3:

Table 5.3: Implementation case 1 parameters.

Simulation	Increment	Number of thickness values	Number of simulations	Thickness values
1	0,2	11	1331	[0,1 0,3 0,5 0,7 0,9 1,1 1,3 1,5 1,7 1,9 2,1]

### - kPCA

The results obtained regarding the kPCA method are shown in Figure 5.1. The values shown are those of the 40 first eigenvalues of matrix  $\tilde{\Lambda}$ .

1522425.88163	6434.57993693	54.1187449497	21.9086160221
1.41215640719	0.784158192745	0.117696162573	0.0302019291511
0.00752449688278	0.00451610564253	0.00187694926015	0.00174270956656
0.000208734172183	4.47524155077e-05	4.2162306183e-05	2.35102304376e-05
2.09371260988e-05	1.34940512968e-05	6.53804329216e-06	4.70439500481e-06
2.70686910641e-06	7.22255520422e-07	6.24696838277e-07	2.28571878234e-07
1.97756666608e-07	1.73143224273e-07	1.30717174582e-07	6.90890411306e-08
2.02176051508e-08	1.44495756953e-08	9.22389585885e-09	8.39554124238e-09
6.23850569538e-09	6.18460227861e-09	2.87401067391e-09	2.68875529937e-09
2.66446774764e-09	1.49503189525e-09	1.23429456037e-09	1.19074075153e-09

Figure 5.1: First 40 eigenvalues obtained in the kPCA.

The total variability contained in matrix  $\tilde{\Lambda}$ , is the following:

$$\sum_{j=1}^{1331} \lambda_j^{kPCA} = 1528938,85$$

From this value, we can obtain the tolerance that we will consider to carry out the POD analysis. Table 5.3 shows the values of the first three eigenvalues, their sum, the total variability, and the tolerance.

Table 5.3: First three eigenvalues, sum, total variability, and tolerance

$\lambda_1^{kPCA}$	1522425,9
$\lambda_2^{kPCA}$	6434,5799
$\lambda_3^{kPCA}$	21,908616
$\lambda_1^{kPCA} + \lambda_2^{kPCA} + \lambda_3^{kPCA}$	1528914,6
Total variability (kPCA)	1528938,9
Tolerance	0,9999841

## - POD

With these values, we can evaluate the data according to the POD method. In this case Figure 5.2 shows the first 40 eigenvalues of matrix  $\tilde{\Lambda}$ .

0.000659415288147	4.76728333699e-05	7.90931175447e-06	3.6353991764e-06
1.83456783879e-06	3.42555753599e-07	2.83858175161e-07	2.19244234978e-07
6.08301443817e-08	4.64947788756e-08	3.03548211334e-08	1.36669379765e-08
1.00170224011e-08	7.53430621255e-09	4.38347512964e-09	2.89700271925e-09
2.2037668945e-09	1.90690701218e-09	1.09896622204e-09	6.77533432258e-10
3.28987390194e-10	2.70490312976e-10	2.26863504109e-10	1.75159033408e-10
1.44487717124e-10	1.13065402972e-10	1.02985189035e-10	8.0342861746e-11
4.18091989229e-11	3.61594187016e-11	2.45273364283e-11	1.75335312915e-11
1.24682454408e-11	9.17215800574e-12	8.62007320297e-12	7.79424522748e-12
5.97990518927e-12	4.68781692585e-12	4.19213602757e-12	3.12402558886e-12

Figure 5.2: First 40 eigenvalues obtained in the POD.

The total variability contained in  $\tilde{\Lambda}$  is the following:

$$\sum_{j=1}^{1331} \lambda_j^{POD} = 7,21496758 \times 10^{-4}$$

From these values, the dimensionality reduction can be performed. Table 5.4 shows the tolerance that needs to be considered to establish the same conditions as in the kPCA method, the total variability of the POD method, the minimum variability that needs to be considered according to the tolerance, and the number of dimensions that will result from the dimensionality reduction.

Table 5.4: Tolerance, total variability, required variability, and resulting number of dimensions.

Tolerance	0,9999841
Total variability (POD)	7,2149675 E-04
Minimum variability	7,2148530 E-04
Dimensions	15

Finally, Table 5.5 shows the minimum variability imposed by the tolerance, the variability of the first dimensions that respects this tolerance, and their sum, which demonstrate that the tolerance is satisfied:

Table 5.5: Minimum variability, eigenvalue of all first 15 dimensions, and sum.

Minimum variability	7,21497E-04
$\lambda_1^{POD}$	5,59E-04
$\lambda_2^{POD}$	4,77E-05
$\lambda_3^{POD}$	7,91E-06
$\lambda_4^{POD}$	3,64E-06
$\lambda_5^{POD}$	1,83E-06
$\lambda_6^{POD}$	3,43E-07
$\lambda_7^{POD}$	2,84E-07
$\lambda_8^{POD}$	2,19E-07
$\lambda_9^{POD}$	6,08E-08
$\lambda_{10}^{POD}$	4,65E-08
$\lambda_{11}^{POD}$	3,04E-08
$\lambda_{12}^{POD}$	1,37E-08
$\lambda_{13}^{POD}$	1,00E-08
$\lambda_{14}^{POD}$	1,83E-06
$\lambda_{15}^{POD}$	7,53E-09
TOTAL	7,21486E-04

## 5.2 Implementation case 2

The first set of results is obtained by following the procedure described previously, with the parameters shown in Table 5.6.

Table 5.6: Implementation case 2 parameters

Simulation	Increment	Number of thickness values	Number of simulations	Thickness values
2	0,4	6	216	[0,1 0,5 0,9 1,3 1,7 2,1]

### - kPCA

The results obtained regarding the kPCA method are shown in Figure 5.3. The values shown are those of the 40 first eigenvalues of matrix  $\tilde{\Lambda}$ .

3.70144429e+04	3.36560188e+02	2.59469671e+00	1.03129240e+00
6.46566763e-02	5.36191085e-02	4.02672902e-03	9.77481007e-04
6.68447900e-04	1.21011678e-04	5.25818113e-05	3.57815283e-05
1.79369374e-05	4.64910973e-06	9.96349414e-07	4.72642858e-07
3.78881780e-07	2.28861770e-07	1.95665908e-07	5.67380035e-08
3.52424303e-08	1.60284114e-08	1.18107638e-08	8.58291933e-09
5.49649950e-09	3.93282231e-09	3.64386381e-09	8.34868681e-10
4.66804446e-10	2.84419876e-10	1.57423366e-10	5.17004131e-11
4.54926300e-11	3.98804608e-11	2.58313807e-11	2.07594810e-11
1.87622718e-11	1.60432811e-11	1.04900490e-11	8.76579705e-12

Figure 5.3: First 40 eigenvalues obtained in the kPCA.

The total variability contained in matrix  $\tilde{\Lambda}$ , is the following:

$$\sum_{j=1}^{216} \lambda_j^{kPCA} = 37354,7533$$

From this value, we can obtain the tolerance that we will consider to carry out the POD analysis. Table 5.7 shows the values of the first three eigenvalues, their sum, the total variability, and the tolerance as a percentage.

Table 5.7: First three eigenvalues, sum, total variability, and tolerance

$\lambda_1^{kPCA}$	37014,443
$\lambda_2^{kPCA}$	336,56018
$\lambda_3^{kPCA}$	2,5946967
$\lambda_1^{kPCA} + \lambda_2^{kPCA} + \lambda_3^{kPCA}$	37353,597
Total variability (kPCA)	37354,753
Tolerance	0,9999691

## - POD

With these values, we can evaluate the data according to the POD method. In this case Figure 5.4 shows the first 40 eigenvalues of matrix  $\tilde{\Lambda}$ .

1.58155381e-04	1.11855526e-05	1.76793214e-06	6.79189289e-07
3.37121879e-07	9.46761170e-08	4.80626152e-08	2.57699886e-08
1.13679864e-08	7.70399716e-09	5.17846421e-09	2.85867732e-09
2.52092202e-09	8.75298338e-10	6.19846237e-10	3.91355370e-10
2.31161990e-10	1.60946745e-10	9.37962838e-11	8.85118233e-11
5.53671638e-11	4.76096621e-11	3.95353730e-11	2.21682282e-11
1.50172322e-11	1.16243697e-11	5.86594295e-12	4.80316799e-12
3.56547647e-12	3.13054198e-12	1.91482616e-12	1.69395611e-12
1.32911972e-12	1.03094316e-12	8.55096445e-13	6.81951293e-13
6.14260861e-13	4.49108311e-13	2.90440029e-13	2.12850479e-13

Figure 5.4: First 40 eigenvalues obtained in the POD.

The total variability contained in  $\tilde{\Lambda}$  is the following:

$$\sum_{j=1}^{216} \lambda_j^{POD} = 1,72320795 \times 10^{-4}$$

From these values, the dimensionality reduction can be performed. Table 5.8 shows the tolerance that needs to be considered to establish the same conditions as in the kPCA method, the total variability of the POD method, the minimum variability that needs to be considered according to the tolerance, and the number of dimensions that will result from the dimensionality reduction.

Table 5.8: Tolerance, total variability, required variability, and resulting number of dimensions.

Tolerance	0,9999691
Total variability (POD)	1,7232079 E-04
Minimum variability	1,7232066 E-04
Dimensions	12

Finally, Table 5.9 shows the minimum variability imposed by the tolerance, the variability of the first dimensions that respects this tolerance, and their sum, which demonstrate that the tolerance is satisfied:



Table 5.9: Minimum variability, eigenvalue of all first 12 dimensions, and sum.

Minimum variability	1,72321E-04
$\lambda_1^{POD}$	1,58E-04
$\lambda_2^{POD}$	1,12E-05
$\lambda_3^{POD}$	1,77E-06
$\lambda_4^{POD}$	6,79E-07
$\lambda_5^{POD}$	3,37E-07
$\lambda_6^{POD}$	9,47E-08
$\lambda_7^{POD}$	4,81E-08
$\lambda_8^{POD}$	2,58E-08
$\lambda_9^{POD}$	1,14E-08
$\lambda_{10}^{POD}$	7,70E-09
$\lambda_{11}^{POD}$	5,18E-09
$\lambda_{12}^{POD}$	2,86E-09
TOTAL	1,72326E-04

### 5.3 Implementation case 3

The first set of results is obtained by following the procedure described previously, with the parameters shown in Table 5.10.

Table 5.10: Implementation case 3 parameters.

Simulation	Increment	Number of thickness values	Number of simulations	Thickness values
3	0,5	5	125	[0,1 0,6 1,1 1,6 2,1]

#### - kPCA

The results obtained regarding the kPCA method are shown in Figure 5.5. The values shown are those of the 40 first eigenvalues of matrix  $\tilde{\Lambda}$ .

1.18501804e+04	1.45367119e+02	1.06586264e+00	4.41438805e-01
2.63723191e-02	1.95985111e-02	1.35140793e-03	4.21929178e-04
2.62726739e-04	4.81223932e-05	1.57012402e-05	9.14330792e-06
5.27425179e-06	2.37669590e-06	3.82431662e-07	1.56226505e-07
5.54302047e-08	4.36413861e-08	2.53680159e-08	1.29999646e-08
1.11148821e-08	5.83094452e-09	3.45724149e-09	2.47316397e-09
1.86376414e-09	4.68786037e-10	3.26985337e-10	1.80179612e-10
9.55508834e-11	3.96874495e-11	1.86214222e-11	7.04970819e-12
6.88041355e-12	5.55433050e-12	4.24491606e-12	4.07056784e-12
3.96557700e-12	3.10140641e-12	2.44301620e-12	2.14633545e-12

Figure 5.5: First 40 eigenvalues obtained in the kPCA.

The total variability contained in matrix  $\tilde{\Lambda}$ , is the following:

$$\sum_{j=1}^{125} \lambda_j^{kPCA} = 11997,1029$$

From this value, we can obtain the tolerance that we will consider to carry out the POD analysis. Table 5.11 shows the values of the first three eigenvalues, their sum, the total variability, and the tolerance as a percentage.

Table 5.11: First three eigenvalues, sum, total variability, and tolerance

$\lambda_1^{kPCA}$	11850,180
$\lambda_2^{kPCA}$	145,36712
$\lambda_3^{kPCA}$	1,0658626
$\lambda_1^{kPCA} + \lambda_2^{kPCA} + \lambda_3^{kPCA}$	11996,613
Total variability (kPCA)	11997,103
Tolerance	0,9999592

## - POD

With these values, we can evaluate the data according to the POD method. In this case Figure 5.6 shows the first 40 eigenvalues of matrix  $\tilde{\Lambda}$ .

1.05461588e-04	7.43591880e-06	1.15656680e-06	3.97570984e-07
1.92041293e-07	6.76084801e-08	2.65765690e-08	1.35501796e-08
5.28532664e-09	4.79995071e-09	2.91479013e-09	1.67739686e-09
1.09015046e-09	4.46551866e-10	3.20558442e-10	2.34820314e-10
1.18056672e-10	5.78770462e-11	4.84812204e-11	3.46335359e-11
2.98211690e-11	2.37729706e-11	1.86481026e-11	9.09172473e-12
7.60187136e-12	5.86804181e-12	2.62661352e-12	2.28427075e-12
1.32522199e-12	1.04006176e-12	6.26689178e-13	5.25833204e-13
4.40355829e-13	3.60690738e-13	2.95473293e-13	2.72414195e-13
1.67915916e-13	1.09287282e-13	7.36442857e-14	6.19051116e-14

Figure 5.6: First 40 eigenvalues obtained in the POD.

The total variability contained in  $\tilde{\Lambda}$  is the following:

$$\sum_{j=1}^{125} \lambda_j^{POD} = 1,14764421 \times 10^{-4}$$

From these values, the dimensionality reduction can be performed. Table 5.12 shows the tolerance that needs to be considered to establish the same conditions as in the kPCA method, the total variability of the POD method, the minimum variability that needs to be considered according to the tolerance, and the number of dimensions that will result from the dimensionality reduction.

Table 5.12: Tolerance, total variability, required variability, and resulting number of dimensions.

Tolerance	0,9999592
Total variability (POD)	1,1476442 E-04
Minimum variability	1,1476387 E-04
Dimensions	11

Finally, Table 5.13 shows the minimum variability imposed by the tolerance, the variability of the first dimensions that respects this tolerance, and their sum, which demonstrate that the tolerance is satisfied:

Table 5.13: Minimum variability, eigenvalue of all first 11 dimensions, and sum.

Minimum variability	1,14764E-04
$\lambda_1^{POD}$	1,05E-04
$\lambda_2^{POD}$	7,44E-06
$\lambda_3^{POD}$	1,16E-06
$\lambda_4^{POD}$	3,98E-07
$\lambda_5^{POD}$	1,92E-07
$\lambda_6^{POD}$	6,76E-08
$\lambda_7^{POD}$	2,66E-08
$\lambda_8^{POD}$	1,36E-08
$\lambda_9^{POD}$	5,29E-09
$\lambda_{10}^{POD}$	4,80E-09
$\lambda_{11}^{POD}$	2,91E-09
TOTAL	1,14769E-04

#### 5.4 Implementation case 4

The first set of results is obtained by following the procedure described previously, with the parameters shown in Table 5.14.

Table 5.14: Implementation case 4 parameters.

Simulation	Increment	Number of thickness values	Number of simulations	Thickness values
4	1,0	3	27	[0,1 1,1 2,1]

#### - kPCA

The results obtained regarding the kPCA method are shown in Figure 5.7. The values shown are those of all 27 eigenvalues of matrix  $\tilde{\Lambda}$ .

4.29245168e+02	1.37444215e+01	1.10915737e-01	3.76263550e-02
2.03800426e-03	3.15184322e-04	9.01623943e-05	3.20884616e-05
5.33158959e-06	2.97671740e-06	1.25741133e-06	2.68464541e-07
1.99079695e-07	1.55017186e-08	3.51604462e-09	5.85271985e-10
4.19535072e-10	1.40015748e-10	1.30360484e-10	4.96690870e-11
1.69019815e-11	5.78432158e-13	1.02665073e-13	6.50970640e-14
8.18597548e-15	4.42408523e-15	2.29480440e-15	

Figure 5.7: All 27 eigenvalues obtained in the kPCA.

The total variability contained in matrix  $\tilde{\Lambda}$ , is the following:

$$\sum_{j=1}^{27} \lambda_j^{kPCA} = 443,140617$$

From this value, we can obtain the tolerance that we will consider to carry out the POD analysis. Table 5.15 shows the values of the first three eigenvalues, their sum, the total variability, and the tolerance as a percentage.

Table 5.15: First three eigenvalues, sum, total variability, and tolerance

$\lambda_1^{kPCA}$	420,24517
$\lambda_2^{kPCA}$	13,744422
$\lambda_3^{kPCA}$	0,1109157
$\lambda_1^{kPCA} + \lambda_2^{kPCA} + \lambda_3^{kPCA}$	443,10051
Total variability (kPCA)	443,14062
Tolerance	0,9999095

## - POD

With these values, we can evaluate the data according to the POD method. In this case Figure 5.8 shows all 27 eigenvalues of matrix  $\tilde{\Lambda}$ .

3.49348196e-05	2.47796936e-06	3.66687999e-07	8.25056464e-08
3.02610431e-08	2.00353582e-08	4.91779422e-09	3.01213937e-09
1.42942427e-09	8.58400755e-10	4.68338681e-10	1.08593756e-10
7.10444930e-11	2.07596120e-11	1.48391696e-11	9.59422334e-12
2.90781648e-12	9.96876163e-13	5.58714908e-13	4.14487888e-13
1.11204742e-13	4.50015887e-14	1.25469831e-14	1.05644164e-14
5.37702900e-15	1.90355295e-16	-2.27798739e-21	

Figure 5.8: All 27 eigenvalues obtained in the POD.

The total variability contained in  $\tilde{\Lambda}$  is the following:

$$\sum_{j=1}^{27} \lambda_j^{POD} = 0,37920209 \times 10^{-4}$$

From these values, the dimensionality reduction can be performed. Table 5.16 shows the tolerance that needs to be considered to establish the same conditions as in the kPCA method, the total variability of the POD method, the minimum variability that needs to be considered according to the tolerance, and the number of dimensions that will result from the dimensionality reduction.

Table 5.16: Tolerance, total variability, required variability, and resulting number of dimensions.

Tolerance	0,999909
Total variability (POD)	3,792021 E-05
Minimum variability	3,791976 E-05
Dimensions	8

Finally, Table 5.17 shows the minimum variability imposed by the tolerance, the variability of the first dimensions that respects this tolerance, and their sum, which demonstrate that the tolerance is satisfied:

Table 5.17: Minimum variability, eigenvalue of all first 8 dimensions, and sum.

Minimum variability	3,7920E-05
$\lambda_1^{POD}$	3,49E-05
$\lambda_2^{POD}$	2,48E-06
$\lambda_3^{POD}$	3,67E-07
$\lambda_4^{POD}$	8,25E-08
$\lambda_5^{POD}$	3,03E-08
$\lambda_6^{POD}$	2,00E-08
$\lambda_7^{POD}$	4,92E-09
$\lambda_8^{POD}$	3,01E-09
TOTAL	3,7923E-05

## 5.5 Implementation case 5

The first set of results is obtained by following the procedure described previously, with the parameters shown in Table 5.18.

Table 5.18: Implementation case 5 parameters.

Simulation	Increment	Number of thickness values	Number of simulations	Thickness values
5	2,0	2	8	[0,1 2,1]

### - kPCA

The results obtained regarding the kPCA method are shown in FIGURE 5.9. The values shown are those of all 8 eigenvalues of matrix  $\tilde{\Lambda}$ .

2.20271061e+01	1.50134353e+00	2.04046809e-02	1.89616758e-03
2.06931336e-04	1.86403851e-05	1.04354958e-06	8.31169124e-08

FIGURE 5.9: All 8 eigenvalues obtained in the kPCA.

The total variability contained in matrix  $\tilde{\Lambda}$ , is the following:

$$\sum_{j=1}^8 \lambda_j^{kPCA} = 23,5509772$$

From this value, we can obtain the tolerance that we will consider to carry out the POD analysis. Table 6.15 shows the values of the first three eigenvalues, their sum, the total variability, and the tolerance as a percentage.

Table 5.19: First three eigenvalues, sum, total variability, and tolerance.

$\lambda_1^{kPCA}$	22,027106
$\lambda_2^{kPCA}$	1,5013435
$\lambda_3^{kPCA}$	0,0204047
$\lambda_1^{kPCA} + \lambda_2^{kPCA} + \lambda_3^{kPCA}$	23,548854
Total variability (kPCA)	23,550977
Tolerance	0,9999098

- **POD**

With these values, we can evaluate the data according to the POD method. In this case Figure 5.10 shows the 8 eigenvalues of matrix  $\tilde{\Lambda}$ .

1.42537555e-05	1.03865581e-06	1.50631507e-07	2.44679910e-08
2.72801004e-09	9.27089748e-10	2.92505296e-11	4.11912070e-22

Figure 5.10: All 8 eigenvalues obtained in the POD.

The total variability contained in  $\tilde{\Lambda}$  is the following:

$$\sum_{j=1}^8 \lambda_j^{POD} = 0,1547119516 \times 10^{-4}$$

From these values, the dimensionality reduction can be performed. Table 6.16 shows the tolerance that needs to be considered to establish the same conditions as in the kPCA method, the total variability of the POD method, the minimum variability that needs to be considered according to the tolerance, and the number of dimensions that will result from the dimensionality reduction.

Table 5.20: Tolerance, total variability, required variability, and resulting number of dimensions.

Tolerance	0,999910
Total variability (POD)	1,547119 E-05
Minimum variability	1,546980 E-05
Dimensions	5



Finally, Table 5.21 shows the minimum variability imposed by the tolerance, the variability of the first dimensions that respects this tolerance, and their sum, which demonstrate that the tolerance is satisfied:

*Table 5.21: Minimum variability, eigenvalue of all first 5 dimensions, and sum.*

Minimum variability	1,54698E-05
$\lambda_1^{POD}$	1,43E-05
$\lambda_2^{POD}$	1,04E-06
$\lambda_3^{POD}$	1,51E-07
$\lambda_4^{POD}$	2,45E-08
$\lambda_5^{POD}$	2,73E-09
TOTAL	1,54702E-05

## 5.6 Results comparison

Once the results have been presented, we will discuss and compare them in order to extract information regarding the tested parameters, and the effectiveness of each set of simulations.

Different elements will be discussed. First of all, as previously said, we are interested in defining the tolerance, in each case, as the percentage that ensures that the dimensionality reduction in the kPCA results in three dimensions, as this is the number of elements that will have their thickness vary. Therefore, we will study if this parameter changes significantly between the five different cases.

Then, we will assess how the dimensionality reduction performs when compared to the number of simulations computed, or the number of thicknesses tested, in each case.

- **Tolerance for three kPCA eigenvalues**

Regarding the weight of the overall variation represented by the three first eigenvalues obtained by the kPCA method, their value will be compared as a percentage of the total variance. This is because the eigenvalues are not normalized, and the obtained result varies between cases, so we need to use a normalized variable, relative to each case, to compare between them.

The obtained results are shown in Table 5.22:

*Table 5.22: Tolerance of the three first kPCA eigenvalues for each implementation case.*

Implementation case	Tolerance
1	0,9999841
2	0,9999691
3	0,9999592
4	0,9999095
5	0,9999098

As shown, these values are really similar. This shows how only three variables are able of representing almost all of the variability of the problem. Regarding its overall evolution between simulations, we see that it remains fairly constant between all five cases, decreasing from the first one to the fourth one, until it stabilizes.

It's important to consider that the fifth case is based on only 8 simulations, which implies that from 8 eigenvalues, 3 will be taken. Since this is the case, the relative weight of one eigenvalue will be higher than in the other cases, where 27, 125, 216 and 1331 eigenvalues were available. Hence, it's not surprising to see that the result from the fifth simulation varies with respect to the overall behaviour of the other cases. This is shown at the first two points on Figure 5.11:

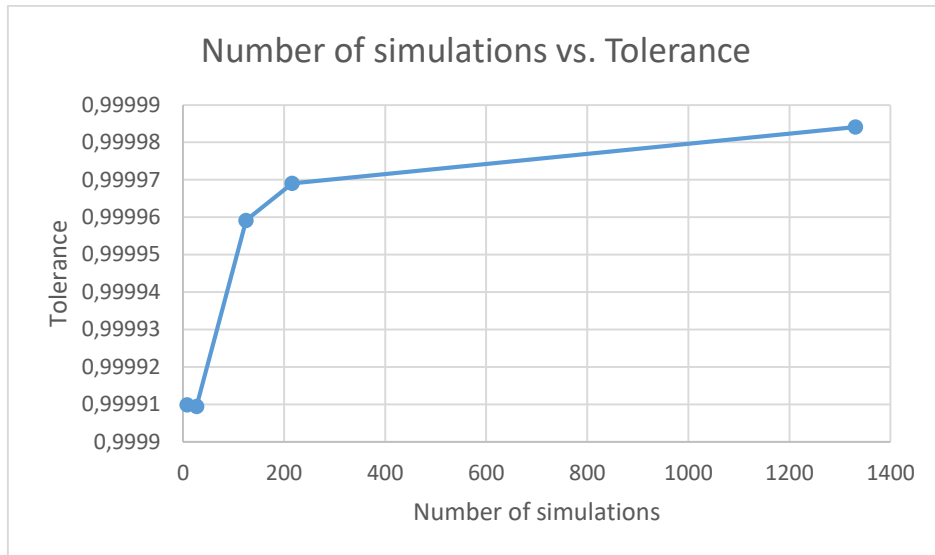


Figure 5.11: Tolerance of the three first kPCA eigenvalues vs. Number of simulations

Regarding the overall behaviour when the number of simulations increase, we see that the tolerance increases rapidly up to the 200 simulations, where it begins to stabilize. It is expected that an implementation of the kPCA with a larger number of simulations (for example, with an increment of 0,1 in each element's thickness, which would result in 9261 simulations) would result in a tolerance close to that of the first implementation case.

- **Dimensionality reduction in POD**

The performance of the dimensionality reduction of the POD method with respect to the tolerance established by the kPCA method will be assessed next. These results are shown in Table 5.23:

Table 5.23: POD dimensionality reduction of each implementation case.

Implementation case	Dimensions after POD reduction
1	15
2	12
3	11
4	8
5	5

In this case, it's clear that the number of dimensions reduces with the thickness increment considered. It's especially interesting to compare the thickness increment to the number of dimensions obtained by the POD, as shown in Figure 5.12:

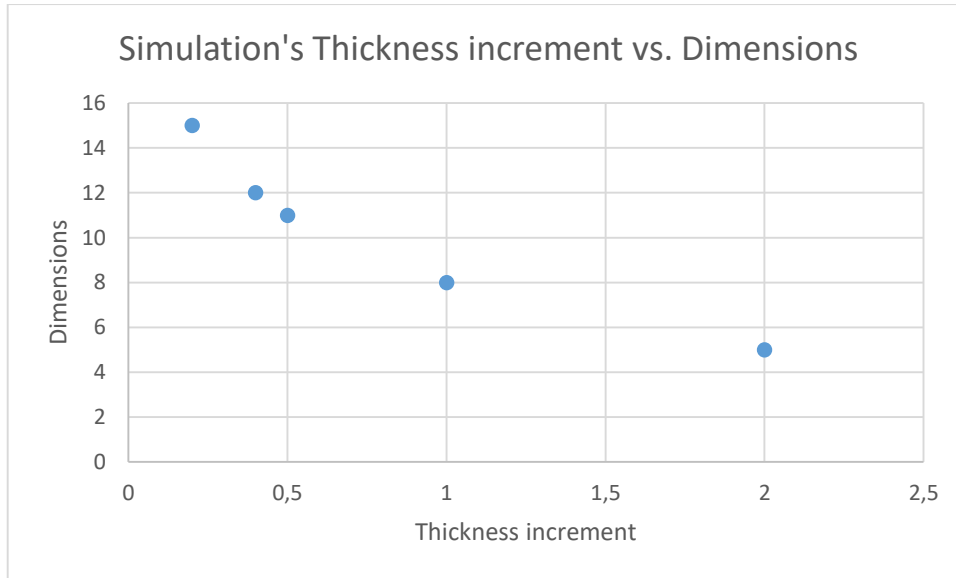


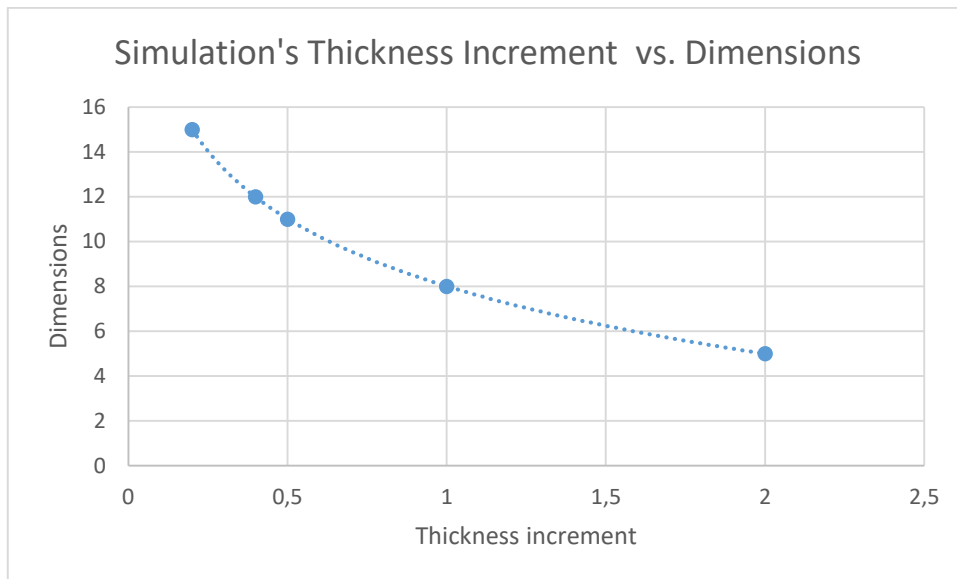
Figure 5.12: Plotting of increment of the implementation case vs the POD dimensions.

It's important to consider that in this figure, the implementation case 1 is the one closest to 0, as it has the smallest thickness increment between its original parameters.

In this case, a logarithmic approach is really successful: Considering the function  $y = m \log(x) + n$ , a linear regression by square minimums can be performed. Table 5.24 shows the value of  $m$  and  $n$ , and Figure 5.13 shows how this function fits the values.

Table 5.24:  $m$  and  $n$  parameters of the linear regression.

$m$	-10,0205
$n$	7,998



*Figure 5.13: Logarithmic fitting of Figure 5.12 points.*

It's important to note, however, that while the logarithmic is continuous, the number of dimensions obtained by the POD is an integer, which means that this approach can never be exact. Still, it provides a really well fitted approximation.

## 6 SURROGATE MODEL IMPLEMENTATION

In this section, the results of the implementation of the surrogate model will be presented and discussed. As said in section 3, this model is based on equations (30) and (11). This results in (34):

$$U^{*T}(f - Kx^{mean}) = U^{*T}KU^*z^*$$

Where the only unknown is  $z^*$ .  $K$  is the stiffness matrix of the model, and  $f$  is the vector of applied loads, and matrix  $U^*$  is obtained from the application of the PCA method.

It has been decided to use the dimensionality reduction results from the implementation case 2, which was the result of 216 different simulations. As stated in section 5.2, the implementation case 2 is capable of reducing the number of dimensions to 12. Considering the case of the PCA instead of the POD (they offer equivalent results), and that the model consisted of 3857 nodes with six degrees of freedom each, this means that these 12 dimensions are the resulting reduction of an initial set of 23142 dimensions.

On the other hand,  $f$  and  $K$  need to be obtained from the FEM software that's being used (NASTRAN, in our case). However, it is important to note the effect of the inertia relief that was commented in section 4.1.2. The model itself didn't have any constraints because of it, because NASTRAN would automatically apply it. However, this is not the case once the matrix  $K$  and the force vector  $f$  are extracted from the software, and since matrix  $U^*$  was constructed with results that did take that into account,  $K$  and  $f$  need to be accordingly modified.

Hence, first of all, the vector  $f$  will have to be modified in order to apply the inertia relief. This is done by considering that the applied force generates a rigid body motion and a relative response. The idea is to modify  $f$  in order to balance its applied forces, so that only a relative motion is generated. In order to do so, the mass matrix  $M$  is also needed, but it is obtained from NASTRAN by the same means as  $K$ . As before, when we first

introduced the concept, we won't go much into detail, but it's important to note that some constraints must be introduced to perform the inertia relief because they are necessary in order to carry out the calculation. In any case, FEM softwares usually provide explanations on its notation and how to implement it; we personally recommend *Abaqus'* approach.

Once this has been done, a new vector  $f'$  will be obtained. The final part will be to take into account the degrees of freedom of the structure in question, by removing six columns and/or rows from the vectors and matrices of the problem. This is due to the fact that, if not, matrix  $K$  will be singular. In this case, six degrees of freedom will have to be removed (one for each displacement direction and rotation direction), so in the final problem,  $K$  will be a  $23136 \times 23136$  matrix,  $U^*$  a  $23136 \times 12$ , and  $f'$  a vector with 23136 components.

From these matrices and vectors, the reduced vector  $z^*$  is computed. If we were interested in computing the associated displacement vector  $x^*$ , we could do this through the following equation:

$$x^* = U^*z^* + x^{mean}$$

Where  $x^{mean}$  is the mean of all displacement vectors used in the computation of  $U^*$  through the PCA. It's important to take it into account because the results obtained by  $U^*z^*$  are centred according to the matrix of data used in the PCA, so  $U^*z^*$  cannot be directly compared to the actual results of a simulation.

## 6.1 Analysis of results

To check the validity of the surrogate model, the computation of the results obtained from the trio of thicknesses [1.0 1.0 1.0] has been performed. Hence,  $K$  will be extracted from a Nastran models that takes this thicknesses into account for the PSHELL elements 8 and 1008, 20 and 1020, and 29.

The results will be analysed in the form of error between the surrogate model calculations and the official results obtained by NASTRAN, since the results itself are displacement vectors of 23142 elements.

The first results to compare will be the relative displacement between the points where the forces are applied in the front of the car, as shown in Figure 4.2. These values correspond to elements 3 and 9 from the displacement vectors. The error is defined as follows:

$$\varepsilon_1 = \frac{\text{abs}[(x^*(3) - x^*(9)) - (x^{Nastran}(3) - x^{Nastran}(9))]}{\text{abs}[x^{Nastran}(3) - x^{Nastran}(9)]}$$

Where  $\text{abs}(\cdot)$  represents the absolute value. The value obtained is the following:

$\varepsilon_1$	0.0069
-----------------	--------

Then, the overall displacement vectors will be compared through the following error:

$$\varepsilon_2 = \frac{\|x^* - x^{Nastran}\|}{\|x^{Nastran}\|}$$

Where  $\|\cdot\|$  represents the norm of the vector, or the square root of the sum of the squares of its elements. The error between the displacement computed from the surrogate model, and the displacement computed from NASTRAN, is the following:

$\varepsilon_2$	0.0569
-----------------	--------

We think that a few considerations need to be taken into account when analysing these results:

First of all, the method used to apply the inertia relief does not coincide in both cases (NASTRAN and surrogate model). While our method is based on manually introducing some boundary conditions, NASTRAN proceeds through what it calls “Automated Inertia Relief Analysis” (AIRA), which does not require constraints because it carries out the stabilization based on characteristics of the model itself. This means that an error will always be introduced in our surrogate model due to this difference in methodology,



since the boundary conditions considered in the second case are unknown, and cannot be replicated.

In fact, the implementation of the inertia relief so that it coincides with the one carried out by NASTRAN is one of the main challenges to obtain successful results in the surrogate model. Still, matrix  $U^*$  has the interesting property of having been generated with a set of results where the inertia relief has been applied, which means that, in the end, the results are projected into a space of solutions. Though this helps with obtaining a realistic result, it also might interfere with the calibration of the boundary conditions for the inertia relief. In our case, this hasn't been a problem because the calibration has been carried out through the solution of  $f=Kx$ .

Still, though  $\varepsilon_2$  seems to be larger than  $\varepsilon_1$ , this is because of a few points that tend to increase the overall error. In fact, when considering relative displacements from a given point as in  $\varepsilon_1$ , around 90% of the cases present an error smaller than 1%, and only 5 of all 23141 cases present an error of 5% or larger.

## 7 CONCLUSION

Reduced computing models are certainly a reliable approach to numerical simulation, offering a simpler implementation and faster results. With the development and popularisation of portable devices such as tablets and smartphones, new possibilities regarding the online construction of surrogate models for offline implementations are becoming a possibility.

In this document, we have presented the implementation of the PCA, POD, and kPCA model order reduction methods in the context of automotive structures, as well as the construction of a reduced, linear elasticity model through the results obtained from these methods.

The data needed in order to carry out the implementation of these different techniques has been obtained through the software NASTRAN. It has been used to generate sets of FEM displacement results associated to different sets of geometric parameters. Though it has been really useful to obtain reliable solutions, it has also introduced several challenges. Nevertheless, this would have been the case for the use any available commercial software.

The POD and the PCA methods have been compared to the kPCA in order to assess the differences between linear and non-linear approaches. Though the dimensionality reduction introduced by the kPCA is more effective, the POD and the PCA can offer the same reliability for a slightly larger number of dimensions (section 5.6). This is especially true for smaller numbers of data; however, it's important to note that, in this document, the selected parameters have only been three, which would not be the case in models with a broader application range. More possible parameters would largely increase the number of simulations (i.e., data sets) required to perform these techniques, and hence, its computational cost would also augment. Still, the logarithmic behaviour of the results presented in section 5.6 should also be applicable to larger cases, and offer valuable information on the calibration the two tested methods. The final decision, however, should consider the required tolerance of the dimensionality reduction, and the

computational time available for the implementation of the model order reduction technique: larger tolerances and a non-linear approach will require a longer implementation time, but might later reduce the time required to use the obtained results in other models.

Regarding the construction of the surrogate model, results shown in section 6 show the validity of its implementation. Still, frictions between the implementation of the inertia relief in NASTRAN and in our case have been detected, so a smaller error could possibly be attained. We believe that changing NASTRAN's automatic approach to one considering a set of degrees of freedom would be the easiest solution, though this would require the construction of a new set of data to perform the MOR techniques from the beginning. Furthermore, the automatic implementation of the inertia relief could respond to other needs not related to this project. Hence, if its manual implementation is not possible, we believe that the calibration of the inertia relief should not be performed using the surrogate model, because, since the set of results used in the dimensionality reduction consists of reliable NASTRAN solutions, the projection over this space could generate a small error even when the actual results are not correct.

Overall, the differences between different implementation cases of the POD and the kPCA have been assessed, and its results have been used in order to prove the validity of a surrogate model to tackle linear elasticity in automotive structures. However, this approach could also be easily implemented in civil engineering structures, where linear elastic analysis is used in most cases. In conclusion, the construction of surrogate models for any case of linear elastic solid mechanics can provide reliable results through simpler and faster models.

## 8 REFERENCES

- [1] Benner, P.; Gugercin, S.; Willcox, K. (2015) *A Survey of Projection – Based Model Reduction Methods for Parametric Dynamical Systems*. SIAM Review 57, no. 4 (January 2015): 483 – 531. Society for Industrial and Applied Mathematics.
- [2] González, D.; Aguado, J. V.; Cueto, E.; Abisset – Chavanne, E.; Chinesta, F. (2016) *kPCA – Based Parametric Solutions Within the PGD Framework*. CIMNE. Barcelona. Spain.
- [3] García – González, A.; Huerta, A.; Zlotnik, S.; Díez, P. (2020). *A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy*. Universitat Politècnica de Catalunya – BarcelonaTech. Barcelona. Spain.
- [4] Izenman, A. I. (2013) *Modern Multivariate Statistical Techniques. Regression, Classification, and Manifold Learning*. Springer. New York.
- [5] Jolliffe, I.T. (2002). *Principal Component Analysis*. Springer. New York.
- [6] Kerschen, G.; Golinval, J.; Vakakis, A. F.; Bergman L. A. (2004) *The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview*. University of Liège. Liège. Belgium.
- [7] MSC Software. (2018) *MSC Nastran 2018. Getting Started Guide*. Newport Beach, USA.