ON THE USE OF SUPPORT VECTOR MACHINES IN STRUCTURAL RELIABILITY

AN UNDERGRADUATE THESIS

BY JUAN CAMILO OSORIO OVIEDO

Supervisor:
Dr. Techn. Diego Andrés Álvarez Marín

Submitted in partial fulfillment of the requirements
for the degree of

Civil Engineer

Department of Civil Engineering
Faculty of Engineering and Arquitecture
Universidad Nacional de Colombia - Sede Manizales

Manizales - Colombia
February - 2022

Who is the happiest of men? He who values the merits of others, and in their pleasure takes joy, even as though it were his own.

— Johann Wolfgang von Goethe

You only live once, but if you do it right,
once is enough.

— Mae West

ACKNOWLEDGMENTS

I want to thank Professor Diego A. Álvarez Marín for his continuous advice and teaching. I also want to thank my parents for their love, patience and support.

Juan Camilo Osorio Oviedo
Manizales, Colombia
February - 2022

CONTENTS

ACRONYMS

CDF	Cumulative Distribution Function
GEV	Generalized Extreme Value
MCM	Monte Carlo Method
MCS	Monte Carlo Simulation
PDF	Probability Density Function
RBF	Radial Basis Function
SVM	Support Vector Machines

1 INTRODUCTION

Now, perhaps more than ever, it is necessary to build trust in the structures since there is skepticism founded on several recent structural collapses although they are still rare events. Structural reliability is thus an area of study that has the potential to save lives. However, to quantify the structural reliability, the probability of failure P_f must be calculated and is usually done through the Monte Carlo Method (MCM) which has the disadvantage that it has a high computational demand.

Not in vain have more efficient alternatives been sought since the early years of the 21st century as polynomial response surfaces adjusted to a design of experiments, chaotic polynomials, multilayer perceptrons, among others. Hurtado and Álvarez (2003)(Hurtado and Alvarez, 2003) proposed a method that employs Support Vector Machines (SVM), in order to convert the computation of the probability of failure into a classification task. In this work, the performance of that algorithm will be verified and compared with the aforementioned Monte Carlo.

1.1 MOTIVATION

Use the concepts learned in the areas of structural reliability, Monte Carlo Simulation (MCS) and Support Vector Machines (SVM) in an eventual master's degree. Additionally and keeping the proportions, have a first experience in writing a document of this type together with its respective research and implementation.

1.2 OBJECTIVE

The idea in this paper is to review some necessary concepts in structural reliability, Monte Carlo simulations and support vector machines.

The python programming language (version 3.8.8) with some libraries such as numpy, matplotlib and sklearn will be used for Monte Carlo and SVM algorithms of which their results will be compared.

IMPORTANT NOTE: The Python codes mentioned in this document as well as the source code of this document in LaTeX can be consulted at the following link:

https://github.com/osvo/SVM_2003

2

THEORETICAL FRAMEWORK

2.1 STRUCTURAL RELIABILITY

2.1.1 *Limit state*

The *limit state function* traces the boundary between the safe set and the failure set in the model domain. It is there where the basic random variables are about to no longer satisfy the design requirements. It is possible to speak of the existence of two main categories of limit states: those of serviceability and those of collapse. The former border on the unacceptable during normal use while the latter border on disaster.

2.1.2 *Probability of failure*

The calculation of the probability of failure is the fundamental objective of structural reliability. Equation 2.1 allows estimating such probability according to whether the response of the structure exceeds a certain threshold or not under the presence of random variables. It is often impossible to solve the integral of equation 2.1 analytically, so there is no choice but to resort to numerical methods such as the Monte Carlo Method. Performing such calculation is computationally expensive, especially the evaluation of the limit state function, then it is an objective to minimize the number of times it is evaluated.

$$P_f = \int_{g(y)\leq 0} p_{\underline{\mathbf{y}}(\mathbf{y})} d\mathbf{y} \tag{2.1}$$

where

$\underline{\mathbf{y}}$	vector of random variables
$p_{\underline{\mathbf{y}}(\mathbf{y})}$	joint density function of $\underline{\mathbf{y}}$
$g(\mathbf{y})$	limit state function

2.2 MONTE CARLO

2.2.1 *History*

It was developed primarily by Stanislaw Ulam and John Von Neumann while they were working on the Manhattan project in Los Alamos. The name of the method was suggested by Nicholas Metropolis and comes from the Monte-Carlo casino located in the principality of Monaco, this due to the fact that Ulam came up with the idea while playing

a game of Canfield solitaire and wanted to calculate the probability of winning a game. He tried using combinatorics but it was getting overly complex, so he thought that numerous games could be played (or simulated) and the proportion of those that were successful could be calculated.

The use of this method was crucial to the success of the Manhattan project and pseudo-random number generators began to be used for it.

2.2.2 *Pseudo-random number generators*

These are algorithms that generate numerical sequences that are close to those of a truly random sequence. Although they appear random, they are in fact completely deterministic, since each value depends on the previous one and the first one is generated from a seed that determines the whole sequence.

The deterministic nature allows sequences to be reproducible but they can have weaknesses such as short periods (sequences are repeated over and over again) and poor distributions. There are different algorithms of this type, in this case the one that comes by default in python and is probably the most popular is used: Mersenne Twister.

2.2.3 *Law of large numbers*

It shows how when performing the same experiment, the average converges to the expected value as the number of times the experiment is performed is increased, as long as it is performed a large enough number of times. In this way it guarantees stable long-term results in random events.

2.3 SAMPLING

2.3.1 *Inverse Transform Sampling*

It is a pseudo-random sampling method that allows samples to be drawn from any Probability Density Function (PDF) whose associated Cumulative Distribution Function (CDF) and its inverse are known.

Given a random variable U that follows a uniform distribution in $[0,1)$ and has an invertible CDF, X can be made to have a distribution F_x if $X = F_X^{-1}(U)$. It is necessary to remember that the relationship between the PDF f_x and the CDF F_x is given by equation 2.2.

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt \tag{2.2}$$

Figure 2.1 is quite illustrative. It shows how from a uniform distribution U in $[0,1)$ in the upper left part and applying the inverse of

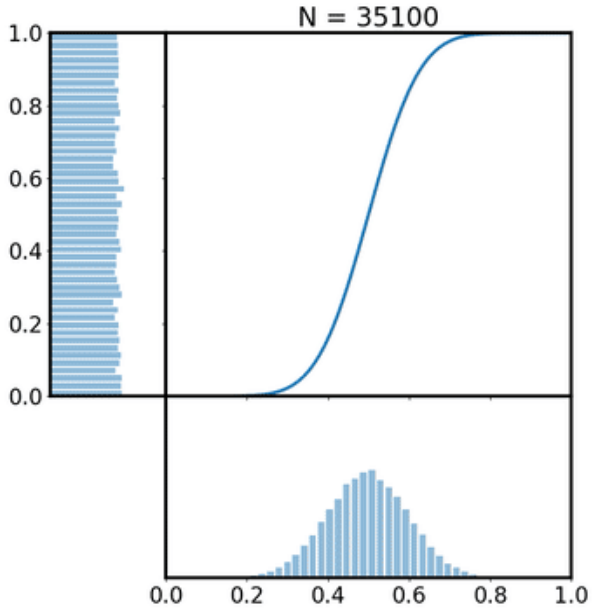

Figure 2.1: Inverse Transform Sampling Example

By Davidjessop, CC BY-SA 4.0, via Wikimedia Commons

Figure 2.2: In solid line is the hyperplane that divides the two categories $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. In dotted line are the lines that limit the margin and pass through the support vectors $\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm 1$.

the CDF, which does the mapping, a sampling that follows the normal PDF (in this case) is formed.

2.4 SUPPORT VECTOR MACHINES

In 1963, Vladimir Vapnik and Alexey Chervonenkis developed the original Support Vector Machines. In the 1990s, it was Vapnik himself who extended the uses of SVM by making use of the kernel trick (see section 2.4.1).

If there are two linearly separable point clouds, by definition it is possible to separate them by at least one hyperplane. It is reasonable to think that the best separating hyperplane is the one that is halfway, being simultaneously away from both clouds so not to be biased. This is called maximizing the margin, defining margin as the distance between the hyperplane and some of the nearest points of each cloud or category. The latter points are called *support vectors*.

It is important to note that once a support vector classifier has been defined, it will only change if samples that are inside the margin are added to it. The hyperplane depends on the support vectors (and their dot product), the vectors further away have no influence, at least in the hard margin SVM (hard and soft margins will be discussed

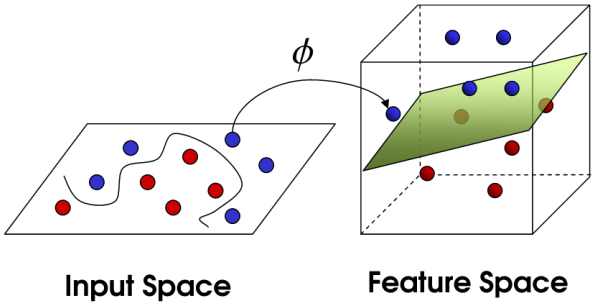

Figure 2.3: Kernel trick

By Shehzadex, CC BY-SA 4.0, via Wikimedia Commons

about shortly). This is key to understand the proposal of Hurtado and Álvarez (2003) Hurtado and Alvarez, 2003.

It is common to encounter data that are not linearly separable so one might think that SVMs are useless in this case. However, a fitting parameter can be introduced to control how much error is admissible. In this case there may be training vectors on the margin and vectors on the wrong side of the hyperplane. This is known as soft margin SVM, as opposed to the one discussed above which is consequently known as hard margin SVM.

The recently mentioned parameter is called C. When C is very small only a few classification errors are allowed making the margin wide and spanning many training vectors, becoming time consuming. In contrast, when C is very large many classification errors are allowed and therefore the margin is smaller.

2.4.1 *Kernels*

A kernel represents a dot product in an output space that is usually of a higher dimension than the input space. A kernel has the purpose of measuring the similarity between two vectors. Equation 2.3 shows the computation of a kernel in which $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$.

This is done because although in a space the data may not be linearly separable, a mapping can be made to a space where it is and there the optimal hyperplane can be found. This is illustrated in figure 2.3.

$$K(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle \tag{2.3}$$

It might be thought that it is necessary to evaluate the vectors in ϕ and then do the dot product, which can be time consuming, especially if it is in a high dimension, and in the end obtain a simple scalar. Actually if a suitable kernel is chosen this is not necessary. The beauty of the kernel trick is that neither ϕ nor the output space m need to be known.

When different kernels are tested, a large number of times common support vectors are found, so their selection is not crucial for the classification, although it is still important.

A disadvantage of kernels is that they may not be intuitive due to the fact that it is generally difficult to properly interpret what they do. They are like a kind of black box.

2.4.1.1 *Radial Basis Function*

Equations 2.4 and 2.5 show equivalent ways of expressing the Radial Basis Function (RBF) kernel which denotes the proximity between two vectors due to its use of the Euclidean norm and its feature space is infinite dimensional.

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right) \tag{2.4}$$

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\gamma\|\mathbf{u}-\mathbf{v}\|^2\right) \qquad \gamma = \frac{1}{2\sigma^2} \tag{2.5}$$

Although there is an expression for γ, it is also considered a free parameter so its value can be adjusted as required.

2.5 ALGORITHM HURTADO AND ALVAREZ, 2003

A databank D is generated as a set of samples of uniformly distributed variables (note that it is neither necessary nor appropriate to use the associated PDFs due to the fact that the uniform distribution scans the space better).

Two samples should be taken, either from D or extras. Of those two points there should be one from each category. The choice of these points can be made under different criteria, a useful one is to place extremely unfavorable conditions that will surely lead to failure for $\mathbf{y_f}$ while for the other point $\mathbf{y_s}$ too favorable conditions are placed.

With the two existing samples a first hyperplane is calculated. On the other hand, points are taken from the dataset D and tested if it is within the current margin. If so, this point is added to the training data set T, removed from D and its corresponding output is calculated. If not, it is still removed from D but will not be taken into account for T. It is worth remembering that in linearly separable classes only the new points that are within the margin have any influence on the classifier, for the rest it is not worth incorporating them into the model.

Eventually all samples in the database have been checked and D is empty. On the other hand, T contains all the necessary samples and the training is finished.

If the size of D is sufficiently large, the margin should be very small and the support vectors should be very close to the decision function on both sides.

It is necessary to mention that scikit-learn assumes that the data is scaled and centered, so in case it is not, the corresponding preprocessing must be done. In section 3.1 a case where the random variables used were normally distributed with mean 0 and standard deviation 1 will be analyzed, so this step was not necessary.

3

IMPLEMENTATION

3.1 TWO-DIMENSIONAL DECISION FUNCTION

Function 3.1 determines the limit state. Both random variables follow a normal distribution with mean 0 and standard deviation 1.

$$g(x_1, x_2) = 2 - x_2 + \exp\left(-\frac{x_1^2}{10}\right) + \left(\frac{x_1}{5}\right)^4 \tag{3.1}$$

The figure 3.2 shows all the samples (10^6). Each ordered pair was evaluated in the limit state function 3.1. If less than zero it was classified as unsafe, otherwise it was classified as safe.

This simulation gives a probability of failure $P_f = 1.7 \times 10^{-3}$. At first glance it would appear that the proportion of points that fail would be much higher. However, it should be noted that a large number of points are overlapping in a very dense area. Therefore, conclusions should not be hastily drawn just by looking at the graph.

On the other hand, 10^4 uniformly distributed samples evaluated on the limit state function were used to train the SVM algorithm although the majority of them were not taken into account as they were not close enough to the margin to be useful.

To initialize the algorithm, one point from each of the two categories was required. The points (x_1, x_2) chosen were $(0, 0)$ which is safe and $(0, 4)$ which is unsafe.

A RBF kernel (see equation 2.5) with $\sigma = \sqrt{5}$ was used. For the calculation of the gamma parameter, the equation 2.5 was slightly modified for a better fit $\left(\gamma = \frac{1}{1.5\sigma^2}\right)$ as it is a free parameter.

Figure 3.2 shows the points that trained the model. They are identified with their actual category and additionally the decision function (a hyperplane in a higher dimension) can be seen. The model ended up with 463 support vectors: 235 safe and 228 unsafe.

To assess the accuracy of the model, 10^4 points different from the training points were evaluated and the confusion matrix shown in Figure 3.3 was formed.

Finally, to calculate the probability of failure, the same 10^6 points used for the MCS (i.e. they employed the same seed) were used and evaluated in the resulting decision function. The result was $P_f = 1.6 \times 10^{-3}$ which is very close to what was originally obtained with far fewer evaluations.

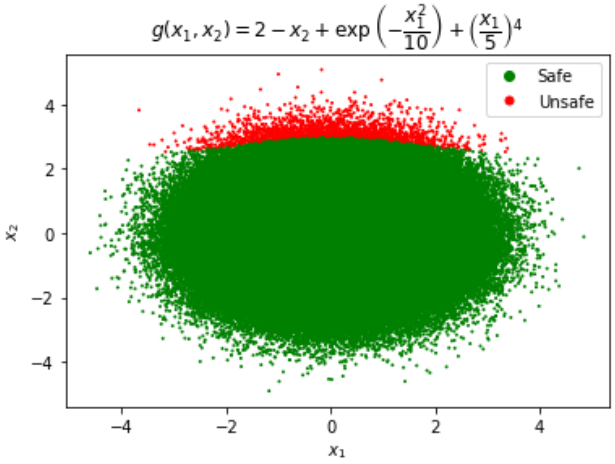

Figure 3.1: Normal random samples and their classification

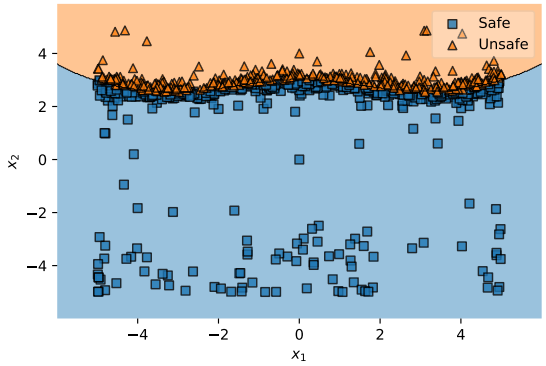

Figure 3.2: SVM implementation with actual categories and decision function

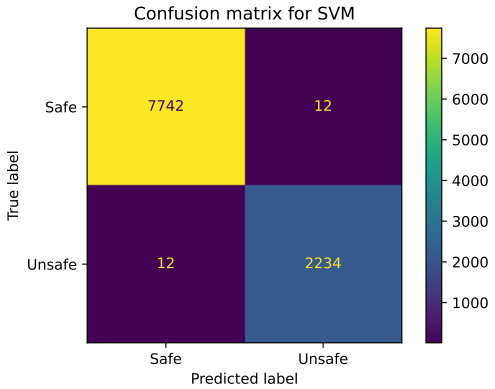

Figure 3.3: Confusion matrix considered in section 3.1

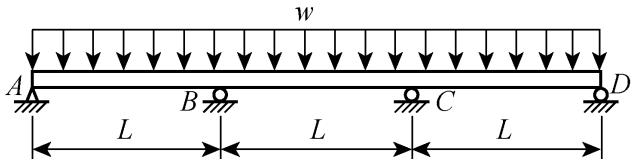

Figure 3.4: Schematic of three-span continuous beam. Taken from Li, Lü and Yue (2006) Li, Lu, and Yue, 2006

VARIABLE	PDF	MEAN	STD	UNITS
q	Normal	10	0.4	$\frac{kN}{m}$
E	Normal	2×10^7	0.5×10^7	$\frac{kN}{m^2}$
I	Normal	8×10^{-4}	1.5×10^{-4}	m^4

Table 3.1: Random variables considered in section 3.2

3.2 RELIABILITY ANALYSIS OF A THREE-SPAN CONTINUOUS BEAM

Consider the three-span beam in figure 3.4. The associated limit state function is expressed in equation 3.2 where the minuend represents the allowable deflection and the subtrahend the maximum deflection. In table 3.1 are the random variables used, which are normally distributed and uncorrelated. The length L is equal to 5 meters.

$$g(q, E, I) = \frac{L}{360} - 0.0069\frac{qL^4}{EI} \tag{3.2}$$

The MCS delivers a probability of failure $P_f = 8.6 \times 10^{-4}$ using 10^6 points. On the other hand, 10^4 points were used to train the SVM model, leaving 836 support vectors: 418 for each category. A value for γ of 30 was chosen resulting in a probability of failure $P_f = 8.1 \times 10^{-4}$.

3.3 A NON-LINEAR DEFLECTION PROBLEM

Figure 3.5 represents a bar whose deflection is expressed by equation 3.4. Table 3.2 shows the random variables together with their PDF including Generalized Extreme Value (GEV), Weibull and Lognormal distributions.

The normal and lognormal probability distributions are strongly related since a random variable X is said to be lognormally distributed if $y = \ln(x)$ has a normal distribution.

It is therefore usual for a lognormal distribution to be defined in terms of its associated normal distribution. In particular, to produce lognormally distributed random variables in the numpy library, the mean and standard deviation of the associated normal distribution shall be passed, for this purpose equations 3.3 are used.

VARIABLE	PDF	MEAN	STD	UNITS
H	GEV	5000	500	N
P	GEV	7000	500	N
I	Weibull	9×10^{-6}	100	$-$
L	Lognormal	4	0.2	m
E	Lognormal	200	10	GPa
φ	Lognormal	80	8	$GN\frac{m}{rad}$

Table 3.2: Random variables considered in section 3.3

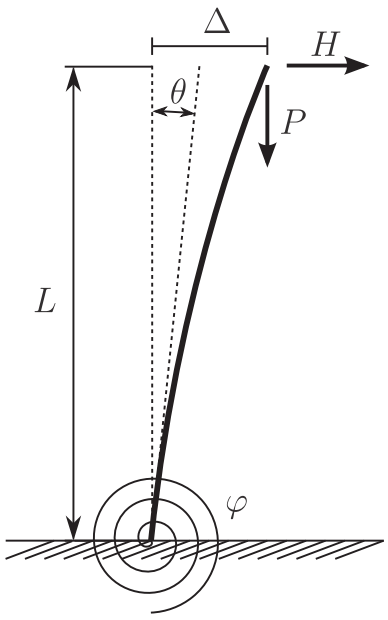

Figure 3.5: Bar studied in section 3.3. Taken from Hurtado and Alvarez, 2013

$$\mu_N = \ln\left(\frac{\mu_{LN}^2}{\sqrt{\mu_{LN}^2+\sigma_{LN}^2}}\right) \qquad \sigma_N = \sqrt{\ln\left(1+\frac{\sigma_{LN}^2}{\mu_{LN}^2}\right)} \tag{3.3}$$

$$\Delta = \frac{H}{EI\left(\frac{P}{EI}\right)^{\frac{3}{2}}}\left[\tan\left(\sqrt{\frac{P}{EI}}L\right) - \sqrt{\frac{P}{EI}}L + \frac{\varphi^2\left(1-\sqrt{1-4HEI\frac{\tan^2\left(\sqrt{\frac{P}{EI}}L\right)}{\varphi^2}}\right)^2}{4HEI\tan\left(\sqrt{\frac{P}{EI}}L\right)}\right] \tag{3.4}$$

The bar is said to fail when the tip displacement is greater than 10 cm. The probability of failure obtained through a Monte Carlo simulation with 10^6 points is $P_f = 1.695 \times 10^{-3}$.

Equations 3.5 and 3.6 describe the GEV and Weibull PDFs, respectively.

$$f(x;\mu,\sigma) = \frac{1}{\sigma}\exp\left(\frac{x-\mu}{\sigma} - \exp\left(\frac{x-\mu}{\sigma}\right)\right) \tag{3.5}$$

$$f(x;\mu,\sigma) = \frac{b}{a^b}x^{b-1}\exp\left(-\left(\frac{x}{a}\right)^b\right)I_{(0,\infty)}(x) \tag{3.6}$$

Equation 3.7 is the result of calculating the cumulative distribution function associated with PDF and writing it as $x = f(y)$ (see section 2.3.1 for more detailed information).

$$x = \sigma\ln\left(\frac{\mu}{\sigma}\right)\ln\left(-\frac{1}{y-1}\right) \tag{3.7}$$

Now, in the implementation of the algorithm described in section 2.5, 10^4 training points were used. The two vectors chosen to initialize the algorithm were taken from the extremes of the population used for the MCS. Then, using 10^6 vectors to evaluate in the resulting decision function, a failure probability $P_f = 1.607 \times 10^{-3}$ was obtained with $\gamma = 0.9$. In total there were 675 support vectors: 339 safe and 336 unsafe.

3.4 DYNAMIC RESPONSE OF A NON-LINEAR OSCILLATOR

Consider the nonlinear oscillator shown in figure 3.6, which has as limit state function 3.8. The random variables are generated according to table 3.3.

$$g(c_1,c_2,m,r,t_1,F_1) = 3r - \left|\frac{2F_1}{m\omega_0^2}\sin\left(\frac{\omega_0 t_1}{2}\right)\right| \tag{3.8}$$

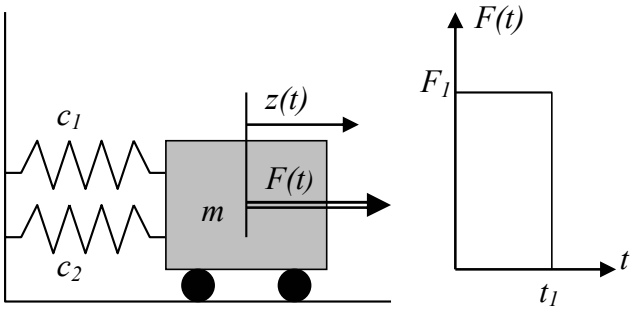

Figure 3.6: Non-linear oscillator – system definition and applied load. Taken from Schueremans and Van Gemert, 2005

VARIABLE	PDF	MEAN	STD
m	Normal	1.0	0.05
c_1	Normal	1.0	0.10
c_2	Normal	0.1	0.01
r	Normal	0.5	0.05
F_1	Normal	1.0	0.20
t_1	Normal	1.0	0.20

Table 3.3: Random variables considered in section 3.4

$$\omega_0 = \sqrt{\frac{c_1 + c_2}{m}} \tag{3.9}$$

The Monte Carlo simulation was performed with 10^6 points and gave a probability of failure $P_f = 2.83 \times 10^{-2}$. The SVM model was trained with 10^4 uniformly distributed points and the points (x_1, x_2) chosen to start the algorithm were taken from the extremes of the MCS. A total of 1414 support vectors were used: 709 safe and 705 unsafe.

In this case a $\gamma = 1/6$ was used. Finally, 10^6 points were evaluated in the decision function and a probability of 2.85×10^{-2} was obtained.

3.5 FINAL THOUGHTS

Support vector machines are simple, elegant and deliver remarkable results using less computational overhead than the Monte Carlo method. The algorithm tested here makes efficient use of samples and avoids unnecessary calls to the limit state function which sometimes requires intensive computation.

BIBLIOGRAPHY

Hurtado, Jorge E. and Diego A. Alvarez (2003). "Classification Approach for Reliability Analysis with Stochastic Finite-Element Modeling." In: *Journal of Structural Engineering* 129.8, pp. 1141–1149. ISSN: 0733-9445. DOI: 10.1061/(asce)0733-9445(2003)129:8(1141).

Hurtado, Jorge E. and Diego A. Alvarez (2013). "A method for enhancing computational efficiency in Monte Carlo calculation of failure probabilities by exploiting FORM results." In: *Computers and Structures* 117, pp. 95–104. ISSN: 00457949. DOI: 10.1016/j.compstruc.2012.11.022.

Li, Hong Shuang, Zhen Zhou Lu, and Zhu Feng Yue (2006). "Support vector machine for structural reliability analysis." In: *Applied Mathematics and Mechanics (English Edition)* 27.10, pp. 1295–1303. ISSN: 02534827. DOI: 10.1007/s10483-006-1001-z.

Melchers, Robert and André Beck (2018). *Structural reliability analysis and prediction*. Third. Wiley. ISBN: 9781119266075.

Nefedov, Alexey (2016). *Support Vector Machines : A Simple Tutorial*.

Schueremans, Luc and Dionys Van Gemert (2005). "Benefit of splines and neural networks in simulation based structural reliability analysis." In: *Structural Safety* 27.3, pp. 246–261. ISSN: 01674730. DOI: 10.1016/j.strusafe.2004.11.001.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/