Dimensionality Reduction and Visualization for Structural Reliability Analysis using Deep Neural Networks

Wellison J. S. Gomes[1]

[1]*Department of Civil Engineering, Center for Optimization and Reliability in Engineering (CORE), Federal University of Santa Catarina*
Rua João Pio Duarte, 205, Córrego Grande, 88037-000, Florianópolis/SC, Brazil
wellison.gomes@ufsc.br

Abstract. Structural reliability analyses may become very computationally demanding, especially when numerical simulations are employed to represent the structural behavior, and/or these analyses are used within structural optimization procedures. To overcome this problem, surrogate models have been widely used in the last decades, helping to avoid evaluations of the demanding parts of the computational code and to reduce the overall computational demand. However, the efficiency of the surrogates is usually compromised when dealing with high dimensional problems. In fact, high dimensionality imposes some difficulties not only to surrogate models but also for some structural reliability methods available in the literature. For these reasons, the present paper proposes to investigate the application of Deep Neural Networks to reduce the dimensionality of structural reliability problems. A proper dimensionality reduction may help visualizing and understanding the problem and may assist surrogate models and reliability methods which would otherwise lose accuracy, precision and/or efficiency when applied to high dimensional problems. The results indicate that: the DNN is indeed capable of performing the dimensionality reduction; the safe and failure classes of samples are clearly distinguishable in all latent spaces considered; even a one-dimensional latent space is enough for the reliability analyses performed herein.

Keywords: structural reliability, dimensionality reduction, deep neural networks.

1 Introduction

Surrogate models, such as kriging and deep neural networks (DNNs), have been widely used in the last decades, in order to alleviate the computational burden associated with many practical applications of, for example, structural optimization and reliability analysis (Forrester, Sóbester and Keane [1], Lima, Evangelista Jr and Guedes Soares [2], Wang *et al.* [3]). This is achieved by employing the surrogate in replacement of the demanding parts of the computational code, commonly those related to the required structural analyses.

In structural reliability analysis, usually the limit state functions are the ones to be replaced by the surrogates, since their evaluations are responsible for a significant amount of the required computational effort. An experimental design (ED), comprising some specific points over the problem domain and the corresponding limit state function responses, is used to construct the surrogate and after that the surrogate is evaluated instead of the true limit state function. To improve accuracy and efficiency, adaptive procedures have been proposed in the literature, such as the so-called AK-MCS, by Echard, Gayton and Lemaire [4], which combines adaptive kriging with Monte Carlo simulation. In the adaptive procedures, the ED is enriched during the analysis, by iteratively selecting points to be included in it, and the surrogate is updated or reconstructed accordingly.

However, a common issue in what concerns surrogate models is the so-called “curse of dimensionality” (Sutton and Barto [5]). As the dimensionality of the problem increases, many surrogates loose efficiency and/or accuracy. A way of dealing with this is by combining the surrogate with dimensionality reductions techniques, such as principal component analysis (Lataniotis, Marelli and Sudret [6]). On the other hand, recent studies on the deep learning area (Poggio and Liao [7]; Hutzenthaler *et al.* [8]) have been indicating that DNNs are capable of avoiding the curse of dimensionality, at least for some kind of problems. In fact, by using adequate numbers of layers and neurons and proper training procedures, the neural network may be able to perform the dimensionality reduction by itself.

Bao *et al.* [9], for example, address high-dimensional reliability analyses by using a deep adversarial autoencoder-based sufficient dimension reduction and an active learning method. The number of dimensions of the reduced space, usually called latent space in the machine learning area, is taken as always equal to two. Good results are obtained when the proposed method is applied to some high dimensional problems and compared to other methods from the literature. Li and Wang [10], on the other hand, also deals with deep learning and high dimensional reliability analyses, but, the DNN is used only for the dimensionality reduction. In a two-dimensional latent space, Gaussian process regression is employed to replace the limit state function and a sampling approach is proposed to update both the neural network and the Gaussian process models.

Dimensionality reduction may also be employed to better understand the problem being solved and the methods used to solve it, as done, for example, in Li and Wang [10] considering the 2D latent space. Taking another example from the literature, Hurtado [11] proposed a transformation of structural reliability problems using polar features and showed that after the transformation, the safe and failure classes of samples were clearly distinguishable, and that it was possible to identify worst-case scenarios, directly related to the samples in the safe domain that are on the verge of the failure domain.

The present paper focuses on the application of DNNs as surrogate models while reducing, at the same time, the dimensionality of the problem. The paper illustrates the capabilities of the DNNs to perform such dimensionality reduction, using a more simple adaptive approach than those usually found in the literature, and also investigates the impact of the number of dimensions of the latent space on the results.

This paper is organized as follows. Section 2 briefly describes the structural reliability problem and its solution via Monte Carlo simulation. Section 3 presents the DNNs and procedures employed herein. In Section 4, examples and results are shown and discussed. Finally, some conclusions and closing remarks are given in Section 5.

2 Structural Reliability

Considering a vector of n_{RV} random variables (RVs), $\boldsymbol{X}$, which represents the uncertainties related to a given structural system, and a vector of realizations of such random variables, $\boldsymbol{x}$. Failure and safety domains, Ω_f and Ω_s respectively, may be defined by employing limit state functions, $g(\boldsymbol{x})$, in such a way that:

$$\begin{aligned} \Omega_f &= \{\mathbf{x} \mid g(\mathbf{x}) \leq 0\}, \\ \Omega_s &= \{\mathbf{x} \mid g(\mathbf{x}) > 0\}. \end{aligned} \tag{1}$$

Each limit state is related to a possible failure mode of the structure, for which a failure probability may be defined as:

$$P_f = P[\mathbf{X} \in \Omega_f] = \int_{\Omega_f} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}, \tag{2}$$

being $f_{\mathbf{X}}(\mathbf{x})$ the joint probability density function of vector $\boldsymbol{X}$.

This multidimensional integral may be solved by structural reliability methods such as FORM, SORM and Monte Carlo simulation (Melchers and Beck [12]), for example. Simple Monte Carlo simulation method (MCS) estimates the failure probability by randomly generating n_{MC} samples of $\boldsymbol{X}$ according to its joint distribution, $f_{\mathbf{X}}(\mathbf{x})$, and by employing an indicator function, $I[\mathbf{x}]$, which is equal to zero if $\mathbf{x}$ belongs to the safety domain and one otherwise. The failure probability is estimated by:

$$P_f = E[I[\mathbf{X}]] \cong \frac{1}{n_{MC}} \sum_{i=1}^{n_{MC}} I[\mathbf{x}_i]. \tag{3}$$

In the context of structural engineering, the number of simulations required to obtain accurate enough estimates of the failure probability is usually high, and the computational burden easily becomes prohibitive.

3 Deep Neural Networks

In structural reliability analyses, artificial neural networks may be used as surrogates of the limit state functions in an attempt to reduce the overall computational effort. To do so, an experimental design is defined, comprising n_{ED} points,

$$\{\mathbf{x}_{ED}^{(1)},\mathbf{x}_{ED}^{(2)},\ldots,\mathbf{x}_{ED}^{(n_{ED})}\}\text{ , with }\mathbf{x}_{ED}^{(i)}\in\mathbb{R}^{n_{RV}}\text{ ,} \tag{4}$$

and the corresponding limit state function values,

$$y_{ED}^{(i)}=g(\mathbf{x}_{ED}^{(i)})\text{ .} \tag{5}$$

The surrogate model is constructed using the ED, and tries to map the relationship between $\boldsymbol{x}$ and $y = g(\boldsymbol{x})$.

Artificial neural networks were introduced by McCulloch and Pitts [13] based on a simplified analogy to the nervous system and have significantly evolved ever since, especially in the area of deep learning (Goodfellow, Bengio and Courville [14]). For the type of DNN used in this paper, see Fig. 1, the relationship is captured by using artificial neurons organized in layers, with weighted connections between neurons of different layers, and previously specified activation functions responsible for the processing within each neuron.

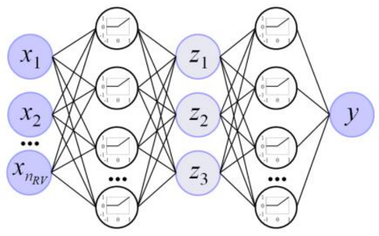

Figure 1. Deep Neural Network.

The activation function f adopted herein is the rectified linear unit (ReLU), defined by

$$f(s)=\begin{cases}s, & \text{if } s\geq 0\\ 0, & \text{otherwise}\end{cases}\text{ .} \tag{6}$$

The mapping is achieved by determining the weights of the connections as well as the other parameters of the neural network, in an iterative process called training. During this process, the inputs are provided to the DNN and their respective outputs are computed propagating the information from the input towards the output layer. The differences between the outputs provided by the DNN and the respective true values are computed and a backpropagation of the error is performed, from the output towards the input layer, updating the parameters of the network. One backpropagation of all the errors, related to entire training data, is called an epoch, which may be divided in batches if the dataset is too large. To do the backpropagation, first a so-called loss function, which is a measure of the error, must be defined.

Here, the DNN is seen as an approximation of the limit state function, so that

$$y_{NET}^{(i)}=\hat{g}(\mathbf{x}_{ED}^{(i)}) \tag{7}$$

and the loss function adopted is the mean squared error, given by:

$$L=\|\mathbf{y}_{ED}-\mathbf{y}_{NET}\|^2 \tag{8}$$

It is also necessary to define a training algorithm to be used during the training process. The limited-memory BFGS is a quasi-Newton method that approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Liu and Nocedal [15]). This algorithm may be used for small networks and data sets that can be processed in a single batch, and is adopted herein.

Considering the previous definitions, the proposed adaptive algorithm consists of the following steps:

1. Randomly generate a Monte Carlo population comprising n_{MC} samples from the joint probability distribution given by $f_{\mathbf{X}}(\mathbf{x})$;

2. Get a sampling pool of n_{POOL} samples from the Monte Carlo population using the farthest apart subset concept. After rescaling the random variables to be in the range $[-1,1]$, using the minimum and maximum respective values obtained from the MCS population, the sample closest to the mean of the RVs is selected and included in the sampling pool. The next selected sample is the defined as the farthest apart sample from all the previously selected ones. The procedure continues, by the criterion of the maximum Euclidean distance between the points, until n_{POOL} samples are selected;

3. Get the first n_{ED} samples from the sampling pool and evaluate the true limit state function on these points. Define the initial ED;

4. Construct or update the surrogate model using the ED:

   4.1 If the ED is not enriched yet, the algorithm is said to be on the initialization phase. Perform n_{RUNS} initializations and trainings of the DNN, using different seeds for the pseudo-random number generator, and choose the DNN with smaller loss function value;

   4.2. If the ED is not the initial one, the algorithm is said to be on the updating phase. The previously trained DNN is updated considering the enriched ED;

5. Estimate the limit state values for the sampling pool, using the DNN. Select n_{ADD} points, taking those with the smallest estimated absolute limit state values among the ones not yet included in the ED. Evaluate the true limit state function on them, include these points in the ED to enrich it;

6. If the number of points in the ED is equal to the maximum number of points defined by the user, compute the failure probability using the surrogate model over the entire MC population, and applying eq. (3). Otherwise go back to step 4.

This adaptive procedure is very simple and relatively easy to implement. It ensures that the points comprising the ED are not too close to each other by employing the farthest apart concept. It further reduces the computational effort by evaluating the surrogate model over the sampling pool instead of over the entire MC population, during the adaptive phase. It tries to include points in the most important regions for reliability analysis, which are those close to the limit state equation ($g(\boldsymbol{x}) = 0$). However, this procedure is expected to be less efficient than those which take into account direct measures of the surrogate model error. Nevertheless, the objective here is to investigate the capabilities of the surrogate model regarding the dimensionality reduction. A simple adaptive procedure seems to be sufficient for this purpose.

4 Numerical Examples

To solve the examples, as depicted in Fig. 1, the DNNs employed have one feature input layer (number of features equal to n_{RV}), fully connected to one hidden ReLU layer with n_{HID} neurons (responsible for the dimensionality reduction), fully connected to the latent space (with n_{LAT} neurons, corresponding to the dimensions of the latent space), fully connected to one hidden ReLU layer with n_{HID} neurons (responsible for representing the limit state function over the latent space) and fully connected to the output layer (with output size equal to one). The number of hidden neurons in each hidden layer is taken as n_{HID}=16 and, for investigation purposes, three different values, 1, 2 and 3, are taken for n_{LAT}. The number of samples in the sampling pool is fixed at n_{POOL}=1000 and the size of the Monte Carlo population changes from example to example. The ED starts with 50 points and is increased, with n_{ADD}=10, until it reaches 200 points.

The computational codes were implemented in MATLAB, using the deep learning toolbox and the corresponding default options, except when otherwise specified. The training algorithm is changed to L-BFGS, with a maximum number of line search iterations equal to 200 and the step and gradient tolerances changed to 10^{-10}. In the initialization phase, n_{RUNS}=5 is adopted with a maximum of iterations of the L-BFGS algorithm equal to 2000. In the updating phase, the maximum number of iterations is changed to 1000.

The computations were performed using an Intel® Core™ I7-9700KF CPU @3.60 GHz processor, with 16GB of RAM, and a NVidia GPU GeForce RTX 4070 Ti Super.

4.1 Example 1: Series System with Four Branches

The first example was proposed in Waarts [16], widely studied in the literature, and consists of a series system with four branches, with two standard normal distributed random variables and the following limit state function:

$$g(x_1, x_2) = \min \left\{ \begin{matrix} 3 + 0.1(x_1 - x_2)^2 - \dfrac{x_1 + x_2}{\sqrt{2}} \\ 3 + 0.1(x_1 - x_2)^2 + \dfrac{x_1 + x_2}{\sqrt{2}} \\ (x_1 - x_2) + \dfrac{6}{\sqrt{2}} \\ (x_2 - x_1) + \dfrac{6}{\sqrt{2}} \end{matrix} \right\}. \quad (9)$$

The final results for this example are presented in Tab. 1, with the reference failure probability being estimated via MCS with n_{MCS}=3·10^5. Figures 2a), b) and c) illustrate the Monte Carlo population divided into failure samples (red squares), and safe samples (blue circles), as well as the initial ED (green diamonds), and the added points (black diamonds). Figures 2d) e) and f) present the results in the latent space.

Table 1. Results for the first example: Series System with Four Branches.

	MCS	DNN-MCS		
		n_{LAT}=1	n_{LAT}=2	n_{LAT}=3
Loss Function	-	2.55E-03	2.49E-04	1.83E-04
P_f	4.43E-03	4.52E-03	4.40E-03	4.45E-03
P_f difference (%)	-	2.03	-0.68	0.38

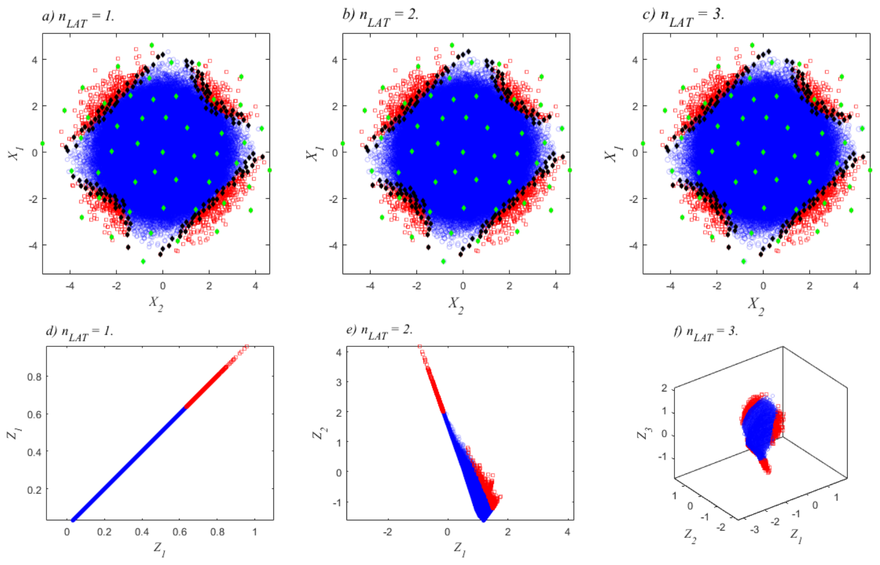

Figure 2. a), b), c) Monte Carlo samples, initial ED and added points in the original space; d) e) f) Monte Carlo samples in the latent spaces.

4.2 Example 2: Dynamic Response of a Nonlinear Oscillator

This example was studied, for example, in Echard, Gayton and Lemaire [4]. It consists of a nonlinear undamped single degree-of-freedom system for which the limit state function is

$$g(c_1, c_2, m, r, t_1, F_1) = 3r - \left| \frac{2F_1}{m\omega_o^2} \sin \frac{\omega_0 t_1}{2} \right|, \quad (10)$$

and $\omega_0 = \sqrt{c_1 + c_2 \ / m}$. The parameters for all six normally distributed random variables are given in Tab. 2.

Table 2. Random variables for the second example: Dynamic Response of a Nonlinear Oscillator.

Variable	Mean	Standard Deviation	Variable	Mean	Standard Deviation
m	1.0	0.05	r	0.5	0.05
c_1	1.0	0.10	F_1	1.0	0.20
c_2	0.1	0.01	t_1	1.0	0.20

The final results for this example are presented in Tab. 3, with the reference *Pf* estimated using n_{MCS}=7·10^4. Figures 3a), b) and c) present the points in the original space, in the plane defined by the two variables with largest absolute values of linear correlations with the limit state function, computed using the initial ED. Figures 3d), e) and f) present the results in the latent space.

Table 3. Results for the second example: Dynamic Response of a Nonlinear Oscillator.

	MCS	DNN-MCS		
		n_{LAT}=1	n_{LAT}=2	n_{LAT}=3
Loss Function	-	2.55E-03	2.49E-04	1.83E-04
P_f	2.79E-02	2.81E-02	2.81E-02	2.84E-02
P_f difference (%)	-	0.82	0.77	1.79

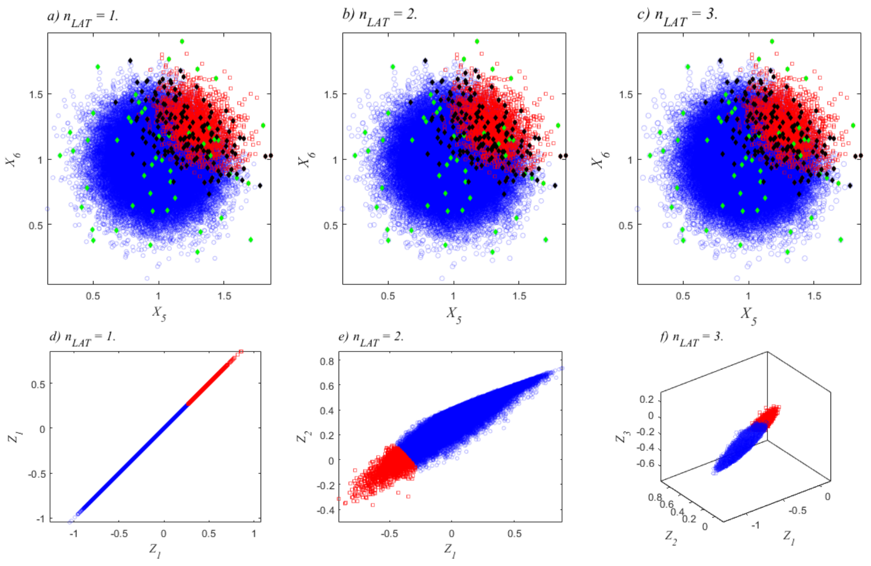

Figure 3. a), b), c) Monte Carlo samples, initial ED and added points in the original space; d) e) f) Monte Carlo samples in the latent spaces.

4.3 Discussions

In all cases the failure probabilities obtained via the adaptive procedure using 200 evaluations of the true limit state functions converged to the reference ones, with differences smaller than about 2%. This indicates that the procedure is indeed inserting points in the vicinity of the limit state equation, as required in structural reliability analyses, and increasing the accuracy of the surrogate model in the most important regions for this kind of analysis.

In what concerns the number of dimensions of the latent space, it is seen that good results were obtained in all cases investigated herein. The best result in the first example was found with n_{LAT}=3, and in the second example with n_{LAT}=2. Even a one-dimensional latent space is enough for these two examples.

It is also seen that the safe and failure classes of samples are clearly distinguishable in all latent spaces, which indicates that the DNN is capable of performing the dimensionality reduction in such a way that it becomes easier to construct the surrogate model for reliability analyses purposes in the reduced space. This is especially true for n_{LAT}=1, case in which it may not even be necessary to construct a surrogate model, but only to define limits for Z_1,

5 Conclusions

The examples showed herein indicated that DNNs are capable of performing the dimension reduction, in such a way that failure and safe classes of samples are clearly distinguishable in the latent space, regardless of the number of dimensions adopted for this space. A one-dimensional latent space is recommended, if it is still enough for more complex problems, because it is easier to deal with and to interpret. A loss function with more focus on the vicinity of the limit state equation could be developed to have a better indication about the quality of the estimated P_f, and to decide if it is necessary to increase the dimensionality of the latent space or not.

On the other hand, the adaptive procedure proposed herein was simple, but enough for the purposes of this paper. It could be replaced by a more efficient one in future studies.

Acknowledgements. This work is supported by the National Council for Technological and Scientific Development (CNPq) via grant 301756/2022-8.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] A. I. J. Forrester, A. Sóbester and A. J. Keane. *Engineering Design via Surrogate Modelling*. John Wiley and Sons, 2008.
[2] J. P. S. Lima, F. Evangelista Jr., C. Guedes Soares. “Hyperparameter-optimized multi-fidelity deep neural network model associated with subset simulation for structural reliability analysis”. Reliability Engineering and System Safety, vol. 239, paper 109492, 2023.
[3] J. Wang, G. Xu, P. Yuan, Y. Li and A. Kareem. “An efficient and versatile Kriging-based active learning method for structural reliability analysis”. Reliability Engineering and System Safety, vol. 241, paper 109670, 2024.
[4] B. Echard, N. Gayton and M. Lemaire. “AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation”. Structural Safety, vol. 33, issue 2, pp. 145-154.
[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2017.
[6] C. Lataniotis, S. Marelli and B. Sudret. Extending classical surrogate modelling to ultrahigh dimensional problems through supervised dimensionality reduction: a data-driven approach. Research Report, ETH Zurich, 2018.
[7] T. Poggio and Q. Liao. “Theory I: Deep networks and the curse of dimensionality.” Bulletin of the Polish Academy of Sciences: Technical Sciences, vol. 66, n. 6, pp. 761-773, 2018.
[8] M. Hutzenthaler, A. Jentzen, T. Kruse and T. A. Nguyen. “A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations”. SN Partial Differential Equations and Applications, vol 1, n. 2, pp. 1–34, 2020.
[9] Y. Bao, H. Sun, X. Guan and Y. Tian. “An active learning method using deep adversarial autoencoder-based sufficient dimension reduction neural network for high-dimensional reliability analysis”. Reliability Engineering and System Safety, vol. 247, paper 110140, 2024.
[10] M. Li and Z. Wang. “Deep learning for high-dimensional reliability analysis”. Mechanical Systems and Signal Processing, vol. 139, paper 106399, 2020.
[11] J. E. Hurtado. “Dimensionality reduction and visualization of structural reliability problems using polar features”. Probabilistic Engineering Mechanics, vol. 29, pp. 16-31.
[12] R. E. Melchers and A. T. Beck. *Structural Reliability Analysis and Prediction*, 3rd ed., Wiley, 2018.
[13] W. McCulloch and W. Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. Bulletin of Mathematical Biophysics, vol. 5, issue 4, pp. 115–133, 1943.
[14] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. The MIT Press, 2016.
[15] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". Mathematical Programming, vol. 45, no. 1, pp. 503-528, 1989.
[16] P. H. Waarts. Structural Reliability Using Finite Element Methods: An Appraisal of DARS—Directional Adaptive Response Surface Sampling. Ph.D. thesis, Technical University of Delft, 2000.