

**APPLICATION OF ARTIFICIAL NEURAL  
NETWORKS TO STRUCTURAL RELIABILITY  
PROBLEMS**

**A Thesis Submitted to  
the Graduate School of  
İzmir Institute of Technology  
in Partial Fulfillment of the Requirements for Degree of**

**MASTER OF SCIENCE**

**In Civil Engineering**

**by**

**Fahri Baran KÖROĞLU**

**June 2023  
İZMİR**

We approve the thesis of **Fahri Baran KÖROĞLU**

**Examining Committee Members:**

---

**Prof. Dr. Engin AKTAŞ**

Department of Civil Engineering, İzmir Institute of Technology

---

**Assoc. Prof. Dr. Çağlayan HIZAL**

Department of Civil Engineering, Ege University

---

**Asst. Prof. Dr. Korhan Deniz DALGIÇ**

Department of Civil Engineering, İzmir Institute of Technology

**22 June 2023**

---

**Prof. Dr. Engin AKTAŞ**

Supervisor

Department of Civil Engineering  
İzmir Institute of Technology

---

**Assoc. Prof. Dr. Marc MAGUIRE**

Co-Supervisor

Durham School of Architectural  
Engineering and Construction  
University of Nebraska-Lincoln

---

**Prof. Dr. Cemalettin DÖNMEZ**

Head of the Department of Civil  
Engineering

---

**Prof. Dr. Mehtap EANES**

Dean of the Graduate School

## ACKNOWLEDGMENT

I would like to express my deepest gratitude to my supervisors Prof. Dr. Engin AKTAŞ and Assoc. Prof. Dr. Marc MAGUIRE for their continuous support in both the motivational, conceptual, and technical aspects of this study. Their vision and knowledge enlightened my path and broadened my horizon during this journey.

This endeavor would not have been possible without the examining committee members Assoc. Prof. Dr. Çağlayan HIZAL and Asst. Prof. Dr. Korhan Deniz DALGIÇ. Their way of thinking and unique comments immensely improved the content of the study.

I am also grateful to Prof. Dr. Murat ALGIN, Asst. Prof. Dr. Muhammet Fethi GÜLLÜ, and Asst. Prof. Dr. Arda Burak EKMEK because of their support to me while I was working in Harran University Civil Engineering Department as a graduate research assistant. Their peerless understanding provided me valuable time to dedicate to the development of ideas that I put forward in this study. We also had fruitful conversations with Asst. Prof. Dr. Arda Burak EKMEK and Asst. Prof. Dr. Muhammet Fethi GÜLLÜ on structural engineering, geotechnical engineering, and scientific programming subjects.

I am also thankful to graduate research assistants Latif Doğan DİNSEVER and Egemen KAYA. We worked with harmony and fun during my time at Harran University.

Thanks should also go to Metin KÖROĞLU for sharing his structural project for the thesis with all the necessary details. Without him, the methodology introduced in this study would not be tested in a real-life RC structure.

Lastly, I would like to extend my sincere thanks to my parents Pınar and İbrahim KÖROĞLU. They always believed in me from my first breath to these days. They raised me with a balance of emotional, ethical, moral, and logical aspects of life. They supported my interest in science and philosophy by approaching me with an open mind and providing the necessary resources since my childhood. I also learned a lot by observing and questioning their daily life and practices.

# ABSTRACT

## APPLICATION OF ARTIFICIAL NEURAL NETWORKS TO STRUCTURAL RELIABILITY PROBLEMS

The contemporary approach in structural engineering indirectly addresses uncertainties arising from load and resistance parameters by using safety factors. To consider these uncertainties in structural engineering, it is necessary to incorporate their statistical properties into the analysis and design process. However, this approach requires the calculation of challenging multi-fold probability integrals. Approximate methods known as FORM and SORM have been developed as an alternative to calculating those integrals. Unfortunately, these methods might have accuracy and convergence problems depending on the problem at hand. Simulation-based structural reliability methods have been developed to overcome the problems associated with approximate methods. The main problem with these methods is that they are often computationally expensive when along with finite element analysis, or it is hard to implement them when a more specific method is chosen to reduce computational costs.

In this study, artificial neural networks have been applied to structural reliability problems to obtain accurate probability estimates with low computational cost. A special type of learning algorithm called Bayesian Regularization was used in the training of artificial neural networks. Additionally, details of the application of artificial neural networks to structural reliability problems are provided.

At the end of the study, the advantages and disadvantages of applying artificial neural networks to structural reliability problems are presented and compared with other known structural reliability methods. Additionally, a new convergence criterion and an adaptive algorithm have been developed. It was observed that applying artificial neural networks to structural reliability problems provides both efficient and accurate probability estimates.



# ÖZET

## YAPAY SİNİR AĞLARININ YAPISAL GÜVENİLİRLİK PROBLEMLERİNE UYGULANMASI

Yapı mühendisliğindeki çağdaş yaklaşım, güvenlik faktörlerini kullanarak talep ve dayanım parametrelerinden kaynaklanan belirsizlikleri dolaylı olarak ele almaktadır. Yapı mühendisliğindeki bu belirsizlikleri göz önünde bulundurmak için, bunların istatistiksel özelliklerinin doğrudan analiz ve tasarım sürecine dahil edilmesi gerekir. Ancak bu yaklaşım, zorlayıcı olan, çok katlı olasılık integrallerinin hesaplanmasını gerektirir. İntegrallerin hesaplanmasına alternatif olarak FORM ve SORM olarak bilinen yaklaşık yöntemler geliştirilmiştir. Ne yazık ki, bu yöntemlerin, eldeki probleme bağlı olarak, doğruluk ve yakınsama sorunları olabilmektedir. Yaklaşık yöntemlerle ilgili problemlerin üstesinden gelmek için benzetim tabanlı yapısal güvenilirlik yöntemleri geliştirilmiştir. Bu yöntemlerle ilgili temel sorun, genellikle ya sonlu elemanlar analizi ile kullanıldıklarında hesaplama maliyetlerinin yüksek olması ya da hesaplama maliyetini azaltmak için daha spesifik bir yöntem seçildiğinde bunların uygulanmasının zor olmasıdır.

Bu çalışmada, yapay sinir ağları düşük hesaplama maliyeti ile doğru olasılık tahminleri elde etmek için yapısal güvenilirlik problemlerine uygulanmıştır. Yapay sinir ağlarının eğitiminde Bayesci Düzenleme adı verilen özel bir öğrenme algoritması türü kullanılmıştır. Ayrıca yapay sinir ağlarının yapısal güvenilirlik problemlerine uygulanmasına ilişkin detaylara da yer verilmiştir.

Çalışmanın sonunda yapay sinir ağlarının yapısal güvenilirlik problemlerine uygulanmasının lehte ve aleyhte noktaları belirtilmiş ve diğer bilinen yöntemlerle karşılaştırılmıştır. Ayrıca uyarlanabilir bir algoritma ve yeni bir yakınsama kriteri geliştirilmiştir. Yapay sinir ağlarının yapısal güvenilirlik problemlerine uygulanmasının hem verimli hem de doğru olasılık tahminleri verdiği gözlemlenmiştir.

*To my dear mom and dad,*

*Pınar & İbrahim KÖROĞLU*

*To my beloved sisters,*

*Pelin Nur & Mira Nur KÖROĞLU*

# TABLE OF CONTENTS

LIST OF FIGURES.....	x
LIST OF TABLES.....	xviii
CHAPTER 1. INTRODUCTION .....	1
1.1. Research Problem and Aim of the Thesis.....	2
1.2. Scope of the Thesis .....	3
1.3. Thesis Outline .....	3
CHAPTER 2. LITERATURE REVIEW .....	4
CHAPTER 3. STRUCTURAL RELIABILITY METHODS.....	11
3.1. Approximate Methods .....	13
3.1.1. Cornell Reliability Index .....	13
3.1.2. Extension to n-Dimensional Linear and Non-Linear Limit State Functions .....	14
3.1.3. Mean Value First Order Second Moment Method (MVFOSM) .....	15
3.1.4. First-Order Reliability Method (FORM).....	16
3.1.4.1. Hasofer-Lind Reliability Index .....	16
3.1.4.2. Hasofer-Lind (HL) Algorithm .....	18
3.1.4.3. Hasofer-Lind Rackwitz-Fiessler (HLRF) Algorithm .....	19
3.1.5. Second-Order Reliability Method (SORM).....	21
3.1.6. Critical Appraisal on Approximate Methods.....	24
3.2. Simulation Methods.....	25
3.2.1. ‘Crude’ Monte Carlo Simulation (CMCS) .....	25
3.2.2. Monte Carlo Simulation with Importance Sampling (IS).....	27
3.2.3. Subset Simulation (SS).....	28
3.2.3.1. Implementation of Subset Simulation .....	29

3.2.4. Directional Simulation (DS) .....	30
3.2.4.1. Importance Sampling in Directional Simulation .....	30
3.2.5. Critical Appraisal on Simulation Methods .....	31
3.3. Surrogate Models.....	32
3.3.1. Response Surface Method (RSM) .....	33
<b>CHAPTER 4. ARTIFICIAL NEURAL NETWORKS.....</b>	<b>36</b>
4.1. Simple Artificial Neuron .....	36
4.2. Architecture of Artificial Neural Networks .....	39
4.3. Learning Paradigms in Artificial Neural Networks .....	39
4.4. Feed-Forward Neural Networks .....	40
4.5. Training Algorithms .....	41
4.5.1. Back Propagation Method and Gradient Descent Algorithm .....	41
4.5.2. Levenberg-Marquardt Algorithm .....	47
4.5.3. Bayesian Regularization .....	49
<b>CHAPTER 5. NUMERICAL EXAMPLES .....</b>	<b>55</b>
5.1. General Framework of Solution Algorithm.....	55
5.1.1. Generation of the Dataset and Analysis with Traditional Reliability Methods .....	55
5.1.2. Training Parameters and Algorithm for ANN .....	56
5.1.3. Determination of ANN Architecture .....	56
5.1.4. Reloading Trained ANN.....	57
5.1.5. Reliability Analysis by using ANN-CMCS.....	57
5.1.6. Repetitive CMCS for the Probability of Failure Estimation.....	57
5.2. Cantilever Beam Example .....	59
5.2.1. Results .....	60
5.3. Simple Portal Frame Example .....	68
5.3.1. Results .....	69
5.4. 12-Story 3-Bay Frame Structure Example.....	76

5.4.1. Results .....	77
5.5. 5-Story 3-Bay Correlated Frame Structure Example.....	85
5.5.1. Results .....	87
5.6. A Real-Life 11-Story Reinforced Concrete Structure.....	99
5.6.1. Lateral Loads and Selection of the Representative Frame.....	109
5.6.2. Determination of Random Variables and Limit State Function .....	114
5.6.3. Modeling of the Frame in OpenseesPy.....	116
5.6.4. Reliability Analysis by Using Pystra and ANN-CMCS .....	120
<b>CHAPTER 6. DISCUSSION.....</b>	<b>127</b>
6.1. 3-Step Convergence Criterion .....	128
6.2. Effect of Different Datasets on the Probability of Failure Estimate .....	130
6.3. Performance of a Neural Network in the Training Phase and Probability of Failure Estimate Capability .....	136
6.4. Comparison of ANN-CMCS Coupling with Other Methods.....	137
6.5. Comparison of the Results with the Results of Previous Studies .....	139
6.5.1. Comparison of the Results for the Cantilever Beam Example .....	139
6.5.2. Comparison of the Results for the Simple Portal Frame Example .....	140
6.5.3. Comparison of the Results for the 12-Story 3-Bay Frame Structure Example .....	140
6.5.4. Comparison of the Results for the 5-Story 3-Bay Correlated Frame Structure Example .....	141
6.6. Adaptive Algorithm for ANN-CMCS Coupling.....	143
<b>CHAPTER 7. CONCLUSION .....</b>	<b>145</b>
<b>REFERENCES .....</b>	<b>147</b>

# LIST OF FIGURES

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 3.1. Cornell reliability index.....	13
Figure 3.2 A limit state function in (a) x-space and (b) u-space.....	17
Figure 3.3. Different design strategies in RSM. Full-factorial design (left), central composite face-centered design (middle), Box-Behnken design (right). ....	33
Figure 4.1. Biological neuron vs. artificial neuron. ....	37
Figure 4.2. Different neural network architectures. ....	39
Figure 4.3. Configuration of single-layer feed-forward neural network and multi-layer feed- forward neural network.....	40
Figure 4.4. An ideal three-layer feed-forward neural network.....	42
Figure 4.5. Simple representation of the flow of inputs in the $j^{\text{th}}$ neuron.....	42
Figure 5.1. Flow chart of the solution algorithm.....	58
Figure 5.2. Configuration of the cantilever beam .....	60
Figure 5.3. Change in the coefficient of variation with respect to the increasing number of simulations for the cantilever beam example.....	61
Figure 5.4. Change in the estimated probability of failure with respect to the increasing number of simulations for the cantilever beam example.....	61
Figure 5.5. Performance metrics of the network that is trained by using 30 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side). ....	62
Figure 5.6. Performance metrics of the network that is trained by using 50 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side) .....	62
Figure 5.7. Performance metrics of the network that is trained by using 100 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side). ....	63
Figure 5.8. Performance metrics of the network that is trained by using 250 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side). ....	63
Figure 5.9. Performance metrics of the network that is trained by using 500 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side). ....	64

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.10. Performance metrics of the network that is trained by using 750 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	64
Figure 5.11. Performance metrics of the network that is trained by using 1000 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	65
Figure 5.12. Performance metrics of the network that is trained by using 1250 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	65
Figure 5.13. Performance metrics of the network that is trained by using 1500 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	66
Figure 5.14. Performance metrics of the network that is trained by using 1750 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	66
Figure 5.15. Performance metrics of the network that is trained by using 2000 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).....	67
Figure 5.16. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the cantilever beam example. ....	67
Figure 5.17. Configuration of the portal frame .....	68
Figure 5.18. Change in the coefficient of variation with respect to the increasing number of simulations for the simple portal frame example. ....	69
Figure 5.19. Change in the probability estimate with respect to the increasing number of simulations for the simple portal frame example. ....	70
Figure 5.20. Performance metrics of the network that is trained by using 30 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	70
Figure 5.21. Performance metrics of the network that is trained by using 50 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	71
Figure 5.22. Performance metrics of the network that is trained by using 100 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	71

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.23. Performance metrics of the network that is trained by using 250 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	72
Figure 5.24. Performance metrics of the network that is trained by using 500 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	72
Figure 5.25. Performance metrics of the network that is trained by using 750 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	73
Figure 5.26. Performance metrics of the network that is trained by using 1000 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	73
Figure 5.27. Performance metrics of the network that is trained by using 1250 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	74
Figure 5.28. Performance metrics of the network that is trained by using 1500 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	74
Figure 5.29. Performance metrics of the network that is trained by using 1750 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	75
Figure 5.30. Performance metrics of the network that is trained by using 2000 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).....	75
Figure 5.31. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the simple portal frame example.....	76
Figure 5.32. Configuration of the 12-story frame .....	77
Figure 5.33. Change in the coefficient of variation with respect to the increasing number of simulations for the 12-story 3-bay frame structure example. ....	78
Figure 5.34. Change in the probability estimate with respect to the increasing number of simulations for the 12-story 3-bay frame structure example. ....	79
Figure 5.35. Performance metrics of the network that is trained by using 30 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	79



<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.36. Performance metrics of the network that is trained by using 50 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	80
Figure 5.37. Performance metrics of the network that is trained by using 100 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	80
Figure 5.38. Performance metrics of the network that is trained by using 250 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	81
Figure 5.39. Performance metrics of the network that is trained by using 500 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	81
Figure 5.40. Performance metrics of the network that is trained by using 750 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	82
Figure 5.41. Performance metrics of the network that is trained by using 1000 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	82
Figure 5.42. Performance metrics of the network that is trained by using 1250 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	83
Figure 5.43. Performance metrics of the network that is trained by using 1500 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	83
Figure 5.44. Performance metrics of the network that is trained by using 1750 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	84
Figure 5.45. Performance metrics of the network that is trained by using 2000 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	84
Figure 5.46. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the 12-story 3-bay frame structure example.....	85
Figure 5.47. The configuration of correlated frame structure .....	87
Figure 5.48. Change in the coefficient of variation with respect to the increasing number of simulations for the 5-story 3-bay correlated frame structure example. ....	88

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.49. Change in the probability estimate with respect to the increasing number of simulations for the 5-story 3-bay correlated frame structure example. ....	88
Figure 5.50. Performance metrics of the network that is trained by using 30 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	89
Figure 5.51. Performance metrics of the network that is trained by using 50 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	89
Figure 5.52. Performance metrics of the network that is trained by using 100 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	90
Figure 5.53. Performance metrics of the network that is trained by using 250 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	90
Figure 5.54. Performance metrics of the network that is trained by using 500 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	91
Figure 5.55. Performance metrics of the network that is trained by using 750 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	91
Figure 5.56. Performance metrics of the network that is trained by using 1000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	92
Figure 5.57. Performance metrics of the network that is trained by using 1250 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	92
Figure 5.58. Performance metrics of the network that is trained by using 1500 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	93
Figure 5.59. Performance metrics of the network that is trained by using 1750 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	93
Figure 5.60. Performance metrics of the network that is trained by using 2000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	94

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.61. Performance metrics of the network that is trained by using 3000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	94
Figure 5.62. Performance metrics of the network that is trained by using 4000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	95
Figure 5.63. Performance metrics of the network that is trained by using 5000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	95
Figure 5.64. Performance metrics of the network that is trained by using 6000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	96
Figure 5.65. Performance metrics of the network that is trained by using 7000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	96
Figure 5.66. Performance metrics of the network that is trained by using 8000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	97
Figure 5.67. Performance metrics of the network that is trained by using 9000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	97
Figure 5.68. Performance metrics of the network that is trained by using 10000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).....	98
Figure 5.69. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the 5-story 3-bay correlated frame structure example.....	98
Figure 5.70. The horizontal response spectrum based on the given soil properties.....	100
Figure 5.71. Section view of the structure in North-East direction.....	102
Figure 5.72. Section of the structure in North-West direction.....	103
Figure 5.73. Ground floor plan.....	104
Figure 5.74. Mezzanine floor plan.....	105
Figure 5.75. Normal floor plan.....	106
Figure 5.76. Ninth floor plan.....	107
Figure 5.77. Roof floor plan.....	108

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.78. Configuration of Grid D. ....	113
Figure 5.79. Equivalent frame model for Grid D. ....	113
Figure 5.80. Deflected shape of shell elements and equivalent frame element. ....	117
Figure 5.81. Frame model built in OpenseesPy. ....	118
Figure 5.82. Deformed shape of the frame.....	118
Figure 5.83. Change in the coefficient of variation with respect to the increasing number of simulations for the real-life 11-story RC frame structure example. ....	120
Figure 5.84. Change in the probability estimate with respect to the increasing number of simulations for the real-life 11-story RC frame structure example. ....	121
Figure 5.85. Performance metrics of the network that is trained by using 50 samples for the real- life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	121
Figure 5.86. Performance metrics of the network that is trained by using 100 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	122
Figure 5.87. Performance metrics of the network that is trained by using 250 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	122
Figure 5.88. Performance metrics of the network that is trained by using 500 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	123
Figure 5.89. Performance metrics of the network that is trained by using 750 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	123
Figure 5.90. Performance metrics of the network that is trained by using 1000 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	124
Figure 5.91. Performance metrics of the network that is trained by using 1250 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	124
Figure 5.92. Performance metrics of the network that is trained by using 1500 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	125
Figure 5.93. Performance metrics of the network that is trained by using 1750 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). ....	125

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 5.94. Performance metrics of the network that is trained by using 2000 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side). .....	126
Figure 5.95. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the real-life 11-story RC frame structure example.....	126
Figure 6.1. Effect of the dataset on the probability of failure estimate for the cantilever beam example. ....	130
Figure 6.2. Effect of the dataset on the probability of failure estimate for the simple portal frame example. ....	131
Figure 6.3. Effect of the dataset on the probability of failure estimate for the 12-story 3-bay frame structure example. ....	131
Figure 6.4. Effect of the dataset on the probability of failure estimate for the 5-story 3-bay frame structure example. ....	132
Figure 6.5. Effect of the dataset on the probability of failure estimate for the real life 11-story RC frame structure example.....	132
Figure 6.6. Flowchart for selecting appropriate method under problem-specific conditions. ..	138
Figure 6.7. Comparison of different results obtained for the cantilever beam example. ....	139
Figure 6.8. Comparison of different results obtained for the simple portal frame structure example. ....	140
Figure 6.9. Comparison of different results obtained for the 12-story 3-bay frame structure example. ....	141
Figure 6.10. Comparison of different results obtained for the 5-story 3-bay correlated frame structure example.....	142
Figure 6.11. Adaptive algorithm for ANN-CMCS coupling. ....	144

# LIST OF TABLES

<b><u>Table</u></b>	<b><u>Page</u></b>
Table 4.1. List of some transfer functions.....	38
Table 5.1. Statistical properties of the cantilever beam .....	59
Table 5.2. Results of FORM, SORM, IS, and CMCS analyses for the cantilever beam example. .....	60
Table 5.3. Statistical properties of the portal frame .....	68
Table 5.4. Results of FORM, SORM, IS, and CMCS analyses for the simple portal frame example. ....	69
Table 5.5. Statistical properties of the 12-story frame structure .....	77
Table 5.6. Results of FORM, SORM, IS, and CMCS analyses for the 12-story 3-bay frame structure example. ....	78
Table 5.7. Statistical properties of the correlated frame structure.....	86
Table 5.8. Results of FORM, SORM, IS, and CMCS analyses for the 5-story 3-bay correlated frame structure example.....	87
Table 5.9. Soil Properties of the Construction Site.....	99
Table 5.10. Material properties. ....	100
Table 5.11. Wind loads acting on the structure.....	109
Table 5.12. Calculated earthquake loads for the structure. ....	110
Table 5.13. Percentage of loads resisted by each grid. ....	111
Table 5.14. Earthquake loads acting on Grid D. ....	112
Table 5.15. The list of random variables for real-life RC frame structure.....	114
Table 5.16. Rotation at both ends of rigid links. ....	119
Table 5.17. Results of FORM, SORM, IS, and CMCS analyses for the real-life 11-story RC frame structure example.....	120
Table 6.1. Results the application of 3-step convergence criterion to the examples.....	129
Table 6.2. Effect of different datasets on the cantilever beam example. ....	133
Table 6.3. Effect of different datasets on the simple portal frame structure example.....	134
Table 6.4. Effect of different datasets on the 12-story 3-bay frame structure example. ....	134
Table 6.5. Effect of different datasets on the 5-story 3-bay correlated frame structure example. .....	135
Table 6.6. Effect of different datasets on the real-life 11-story RC frame structure example. .	135
Table 6.7. Differences in studies in terms of coefficient of correlation.....	142
Table 6.8. Differences in studies in terms of random variables and limit state function. ....	143

# CHAPTER 1

## INTRODUCTION

Structural engineering is the profession of analyzing and designing structures that can withstand the forces of nature, governed by the laws of physics, over long periods of time. A structure is subjected to various uncertain loads imposed by nature throughout its service life, and it must provide adequate resistance against those loads to fulfill its purpose. However, the resistance of a structure is also uncertain, due to factors such as the behavior of materials, the configuration of the structural system, the geometry of its elements, and the assumptions made in mathematical models. These uncertainties are called aleatoric uncertainties. In addition to these, there are also epistemic uncertainties in structural engineering, which stem from the engineer's lack of knowledge. Therefore, a structural engineer must consider all of these uncertainties when analyzing and designing a structure to ensure a sufficient level of safety.

In the contemporary structural analysis and design philosophy, the safety of structures is assured by using load and resistance factors that represent the aleatoric uncertainties in an indirect manner. However, this approach is a deterministic strategy that is in contradiction with the stochastic nature of the loads and resistance parameters. Therefore, more advanced approaches that take into account the uncertainties directly are needed to better represent the uncertainties involved in structural engineering.

The safety of a structure can be evaluated by using the statistical properties of parameters in the structural engineering problem. Regarding those statistical properties, a probabilistic analysis can be performed for the problem at hand, and the effect of the aleatoric uncertainties can be directly imposed on the analysis and design process up to a certain level. However, it is necessary to calculate challenging multi-fold probability integrals in order to obtain a probability estimate if analytical expressions are used. On the other hand, it is possible to calculate those probability integrals approximately to decrease the computational effort. Therefore, over the last fifty years, various structural reliability methods based on the three main approaches—approximate methods,

simulation techniques, and surrogate models—have been developed to evaluate these integrals approximately and obtain a probability estimate.

Chronologically, the first approach developed for structural reliability practice is the approximate methods. In this approach, the governing mathematical expression for a structural reliability problem is approximated by using the first-order or second-order Taylor series expansion, and the probability estimate is made based on a reliability index. Then, the developed simulation techniques are implemented for the structural reliability problems due to the deficiencies observed in the approximate methods. However, in these techniques, it is necessary to generate a large number of samples to perform simulations to obtain a low variance probability estimate.

### **1.1. Research Problem and Aim of the Thesis**

Structural engineering problems are solved using computationally expensive finite element analysis in the present day. Therefore, combining simulation techniques with finite element analysis to perform reliability analysis for a structural engineering problem is generally computationally costly, and in some cases, the cost is prohibitive especially evaluation of the limit state function is expensive. To overcome this problem, surrogate models have started to be used to represent finite element analysis in simulation techniques.

In this study, the aim is to reduce the computational cost of structural reliability analysis by using artificial neural networks (ANN), which are known for their robust function approximation capability and efficiency, to construct a surrogate model for finite element analysis. Then, the trained ANN will be coupled with Crude Monte Carlo Simulation (CMCS) to estimate the probability.

Therefore, the application of the ANN-CMCS coupling will be investigated in detail to determine under which conditions the method is preferable. Furthermore, critical points in the implementation of ANNs for structural reliability problems will also be addressed.



## **1.2. Scope of the Thesis**

The method will be tested by structural reliability problems having implicit limit state functions. For this purpose, 2D frame problems within the linear-elastic range will be used generally. A single exception will be shown to investigate the performance of ANN-CMCS coupling in explicit limit state functions by testing the method on a well-known cantilever beam example. All examples covered in this study will be time-invariant reliability problems and be analyzed by using the linear analysis method.

## **1.3. Thesis Outline**

This thesis is structured into seven chapters. A brief introduction about the research problem and the aim of the thesis is offered to the reader here in Chapter 1. The previous studies that existed on this subject are given in Chapter 2 to provide insight to the reader. The theoretical aspects, mathematical derivations, implementation, and critiques on the most widely used structural reliability methods corresponding to the above-mentioned three main approaches are given in detail in Chapter 3. The fundamentals and extensive mathematical derivations of the artificial neural networks are presented in Chapter 4. The proposed methodology for the application of ANN-CMCS coupling and numerical examples are introduced in Chapter 5 with the results examples. Chapter 6 is dedicated to the discussion of the observations made from the numerical examples. Finally, Chapter 7 summarizes the study and presents the conclusions drawn from the research.

## CHAPTER 2

### LITERATURE REVIEW

Several studies have previously explored the implementation of artificial neural networks for solving structural reliability problems. The most common approach is to use them as surrogates for given limit state functions, typically coupled with a simulation or approximate technique to obtain a probability estimate. However, literature offers diverse perspectives on this subject, with studies combining artificial neural networks with different structural reliability methods or utilizing various types of artificial neural networks. In this section, an overview of previous studies is presented in chronological order.

In the study of Papadrakakis, Papadopoulos, and Lagaros (1996) reliability analysis was performed for steel frame structures. The probability of plastic collapse of the structures was investigated by using artificial neural networks and Monte Carlo Simulation. The proposed methodology was further enhanced by incorporating the Importance Sampling technique to improve the accuracy of the estimates.

Shao and Murotsu (1997) introduced an active learning algorithm by searching the domain of limit state using factorial designs. The algorithm searches the domain of limit state using factorial designs and provides the capability to detect important regions for structural failure with a limited dataset in training. The proposed algorithm was shown to perform well for both single and multiple-limit state functions in reliability analysis.

Artificial neural networks can also be used as classifiers rather than function approximators, and an artificial neural network architecture can be trained using different training algorithms and cost functions. From this point of view, an extensive comparison was performed by Hurtado and Alvarez (2001) by considering different types of neural networks, cost functions, training algorithms, and sampling methods for training data based on the generation of samples uniformly or with respect to statistical properties of random variables. The study revealed that using neural networks as function approximators rather than classifiers yielded more satisfactory results in structural

reliability problems, and there was no significant effect of the sampling method on results. In addition to those observations, the Gauss-Newton algorithm was recommended for training, and as a cost function, it was observed that the sum of square errors performed better than cross-entropy. Even though the radial basis function neural network performed better than the multi-layer feed-forward network, it is observed to have a bifurcation problem.

Goh and Kulhawy (2003) trained artificial neural networks to approximate geotechnical engineering problems. First, an artificial neural network was trained based on the finite element model of the problem at hand, and then the optimal weights and transfer function of the network were used to develop a mathematical expression. Based on the developed explicit expression, it was possible to perform reliability analysis without using the finite element model.

Nie and Ellingwood (2004) used artificial neural networks to identify the shape and location of a limit state function to perform directional simulation by using Fekete point sets. It was noted that using artificial neural networks with directional simulation can provide cost efficiency for reliability problems that have a low effective ratio.

In the study of Gomes and Awruch (2004), the comparison between Response Surface Method and artificial neural networks was presented with respect to the commonly used structural reliability methods such as First-Order Reliability Method, Second-Order Reliability Method, Monte Carlo Simulation, and Monte Carlo Simulation with Adaptive Importance Sampling. The methodology proposed by Shao and Murotsu (1997) was adopted in the study for the neural networks. The results indicated that both Response Surface Method and artificial neural networks can efficiently be employed for structural reliability problems, and they reduce the computation cost in comparison to the other commonly used methods. The study suggested that the methods should be tested on problems with nonlinear high-dimensional systems.

In a later study by H. M. Gomes and Awruch (2005), Response Surface Method and artificial neural networks were applied to the reliability analysis of reinforced concrete structures. Several methods for constructing a response surface and different types of neural networks were compared in the study. Both methods were found to be

suitable for the reliability analysis of reinforced concrete structures. However, it was recommended that the methods should be tested on large scale problems.

Deng et al. (2005) implemented artificial neural network-based FORM, SORM, and Monte Carlo Simulation to structural reliability problems. The gradients to implement FORM were calculated by using artificial neural networks. In a similar manner, the first-order and second-order partial derivatives used to perform SORM were calculated by using artificial neural networks too. The developed methods were tested on a structural reliability problem that has an implicit limit state function. The results demonstrated that the proposed artificial neural network-based FORM and SORM methods are effective for structural reliability problems with implicit and nonlinear limit state functions.

The comparison of the Response Surface Method with artificial neural networks was covered in the study of Hosni Elhewy, Mesbahi, and Pu (2006). The constructed artificial neural network was coupled with Monte Carlo Simulation and First-Order Reliability Method separately and compared with a constructed response surface by using polynomials. It was noted that the reliability analysis by using artificial neural networks was more efficient and accurate than the polynomial-based Response Surface Method even in non-linear limit state functions.

In 2007, Cheng (2007) proposed two methods based on the coupling of genetic algorithms and artificial neural networks to carry out reliability analysis efficiently for problems with implicit limit state functions. For the first method, a generation is generated based on the statistical properties of the random variables involved in the problem. Then, an artificial neural network is used to construct an explicit approximating function for the implicit limit state function, similar to the study of Goh and Kulhawy (2003). After that, new generations are generated by using genetic algorithm operations, namely reproduction, crossover, and mutation, until the convergence criterion for the reliability analysis is achieved. In the second method, Monte Carlo Simulation with Importance Sampling method is used to update the obtained reliability index for further improvement in the results. The proposed methods are suitable for reliability analysis purposes if the problem involves regular-shaped or implicit limit state functions. Additionally, the second method gives a fair estimate for irregular-shaped limit state

functions. It is also observed that the proposed methods are more efficient than Crude Monte Carlo Simulation in terms of computation effort due to requiring fewer samples.

Cardoso et al. (2008) coupled artificial neural networks with Monte Carlo Simulation to perform reliability analysis. The trained network was used in the simulations instead of directly calling the actual limit state function. In the article, it was shown that the neural networks were capable of representing complex structural behaviors and they could be successfully coupled with MCS to perform structural reliability analysis. In addition to that, a numerical example provided in the study showed that the method could be used for reliability-based design optimization purposes.

Cheng and Li (2008) further improved ANN-GA coupling by introducing a method called the “uniform design method” which is denoted as UDM-ANN-GA. This method improves the quality of the selected dataset for training an artificial neural network which leads to better performance. Furthermore, this method also reduces the number of samples to train an artificial neural network. However, it was reported that the method might not converge in some cases. To overcome the convergence issue, a method based on Importance Sampling and denoted as UDM-ANN-GA-MCSIS was suggested. The methods were found effective for structural and non-structural reliability problems in terms of computational cost and more accurate than the traditional ANN-GA method presented in Cheng (2007).

Papadopoulos et al. (2012) combined artificial neural networks and the subset simulation in their study to reduce the variance in probability estimates obtained by using subset simulation method. Basically, samples at each level in the subset simulation were enriched by using a trained neural network based on the random variables generated by Markov chains. It was demonstrated that the combination of artificial neural networks and subset simulation yields a more robust probability estimate with low variance.

The study of Chojaczyk et al. (2015) presented a review and application of artificial neural networks for reliability analysis. The reliability analysis of a stiffened panel was performed in the study. In the review part of the article, the application of artificial neural networks in the structural reliability analysis field was considered, and the related studies were presented in a table form in chronological order. In the second part of the paper, the reliability analysis of a deck-stiffened panel was performed using

three different approaches. The first approach included the application of ANN and MCS combinations to the problem, whereas the combination of ANN and Monte Carlo Simulation with Importance Sampling (MCIS) was employed to the problem as a second approach. In the last approach, the direct coupling of FORM and MCIS with finite element analysis (FEA) was used as a benchmark for ANN-based approaches. The results showed that ANN-based methods could be used in structural reliability analysis instead of traditional reliability methods due to their robustness and efficiency.

Oparaji, Sheu, and Patelli (2017) developed a novel approach to increasing the robustness of artificial neural networks in reliability analysis. They created a set of competing artificial networks and selected the best of them by using model selection and model averaging techniques based on the Bayes theorem. Additionally, they trained each artificial neural network several times to eliminate potential biases and to prevent trapping local minima in the training phase. The decision for the best version of the network was made by using cross-validation. After those steps, a confidence interval for the estimate was determined. It was noted that this approach was computationally expensive due to creating a set of networks and eliminating part of them in the model selection and cross-validation steps. However, it was suggested that parallelization techniques could be used to reduce the cost.

Kroetz, Tessari, and Beck (2017) compared the performances of polynomial chaos expansion (PCE), Kriging, and artificial neural networks. It was observed that all three methods lead to accurate results for structural reliability problems. In terms of computational efficiency, PCE showed the worst performance when a small number of samples were used. Artificial neural networks and Kriging provided approximate results when a small number of samples were used.

In the study conducted by Dudzik and Potrzyszcz-Sut (2019), the reliability analysis of a space truss structure that is susceptible to snap behavior was carried out by using artificial neural networks to create an explicit limit state function based on the finite element analysis results. The explicit limit state function was then coupled with the First-Order Reliability Method to calculate the reliability index. Moreover, the explicit limit state function was also coupled with Second-Order Reliability Methods and Monte Carlo

Simulation to validate the results. Additionally, sensitivity analysis was performed to identify the random variable that has the most impact on the reliability index.

In the study of Beheshti Nezhad, Miri, and Ghasemi (2019) a new type of artificial neural network called the group method of data handling type (GMDH-type) was used along with the response surface method to perform reliability analysis for civil engineering problems. The study compared the GMDH-type neural network with other methods using benchmark problems in the literature. The researchers derived an explicit limit state function for example problems using the parameters of GMDH-type neural networks and the Response Surface Method. Then, the obtained limit state function was used to perform reliability analysis using the FORM, SORM, or Monte Carlo simulation method. The study showed that the introduced method was capable of analyzing high-dimensional, nonlinear, and correlated problems with less computational effort and fair accuracy.

In the study of Li and Wang (2020), a dimension reduction technique was developed and applied to perform reliability analysis in high-dimensional problems. The algorithm of the proposed technique consisted of a deep feedforward network used to reduce the dimension of the problem by converting the actual input parameters into latent variables inspired by high-dimensional data abstraction (HDDA). Then, the Gaussian process was used to create a surrogate model. In the last step, Monte Carlo Simulation was used to perform the reliability analysis based on the developed surrogate model. Moreover, a distance-based iterative sampling strategy was presented in the article to enrich the training data set effectively if more samples were needed to construct a better surrogate model. It was observed that the proposed method could significantly reduce the computation effort without a reduction in accuracy.

In the study of W. J. D. S. Gomes (2020), shallow and deep artificial neural networks were compared for both explicit and implicit limit state functions. The results obtained from the study showed that deep artificial neural networks performed better than shallow ones for reliability problems because their complexity provided them with more refined adaption for the problems at hand. However, an important observation made in the study was that the performance of shallow neural networks was still acceptable even though they were outperformed by deep neural networks.

Lieu et al. (2022) investigated the application of deep neural networks (DNN) to structural reliability problems in their study. In the study, they proposed a two-staged adaptive surrogate model technique based on DNN, and they showed the efficiency of the method by applying it to six example problems. The adaptive surrogate model performed global and local predictions to increase the model's accuracy. In the global predictions, they tried to capture the general behavior of the system by using an initial design of experiments chosen from Monte Carlo Simulation samples randomly. They increased the accuracy by adding more samples until a certain threshold was exceeded. After that, they further increased the model accuracy by performing local predictions by adding more samples from the vicinity of the limit state function and eliminating the unimportant and noisy initial design of experiments. The key point here was that the local prediction stage was not mandatory if the global prediction satisfied the predefined stopping criterion. The example problems revealed that the proposed technique was superior to Monte Carlo Simulation in terms of the number of functional evaluations when the problem was a complex structural system that required finite element analysis, i.e., the problem had an implicit limit state function.



## CHAPTER 3

### STRUCTURAL RELIABILITY METHODS

The failure of a structure can be defined as the exceedance of the demand for the capacity of the structure. The function that defines the relationship between demand and capacity is called the limit state function, and it can be in explicit or implicit form. For a simple R-S problem, R represents the capacity of the structure whereas S represents the demand, the limit state function can be written as:

$$g(R, S) = R - S \quad (3.1)$$

The failure or safety of a structure can be classified based on the sign of the limit state function. The structure is said to be safe if the limit state function is greater than zero. Otherwise, the structure is considered as failed. In classical structural mechanics, these variables are considered deterministic parameters; generally, the capacity and demand can depend on various parameters. If these parameters are denoted as  $X = X_1, X_2 \dots X_n$  then the limit state function is written as  $g(X)$ , and the failure of a structure can be represented by the following equation:

$$g(X) < 0 \quad (3.2)$$

Consequently, the formulation given above yields a single value that represents whether the structure is in a failure or safe situation. Although this deterministic consideration, none of the parameters related to the capacity and demand exhibit deterministic behavior in real life. They include some level of uncertainties in themselves, and the classical approach imposes these uncertainties on parameters by using safety factors. However, such a generalization yields the usage of the same safety factors for different kinds of structures with neglecting their behaviors. Therefore, it is necessary to take into account these uncertainties explicitly by representing the parameters with appropriate random variables. Clearly, this approach cannot eliminate all the uncertainties involved in the problem, however, a more realistic approximation to the nature of the

problem can be achieved in this way. If these parameters are considered random variables, then the probability of failure of a structure can be written as:

$$P_f = P[g(X) < 0] \quad (3.3)$$

The calculation of the probability of failure for an n-dimensional reliability problem is actually the calculation of multifold probability integral over a certain domain which is defined by the range of random variables. The general formulation for a reliability problem is defined as:

$$P_f = P[g(X) < 0] = \int \dots \int_{g(X) < 0} f_X(X) dX \quad (3.4)$$

The term  $f_X(x)$  in equation (3.4) represents the joint probability density function of random variables involved in the problem.

It is also known that the complementary event of a failure situation is equivalent to the safety of a structure. A link between the probability of failure and the structure's safety can be constructed by using the rule of complementary events. Therefore, once the probability of failure of the structure is calculated then, one can easily obtain the safety of the structure by using the following relationship:

$$P_s = 1 - P_f \quad (3.5)$$

As mentioned above, a reliability problem is essentially the calculation of the probability of failure of a structure and the corresponding multifold probability integral. Generally, these types of integrals are challenging and time-consuming to obtain an analytical solution. On the other hand, numerical and approximate methods can be performed to calculate the multifold probability integrals and the probability of failure for a given structure. Therefore, in this chapter, the basic approaches developed for structural reliability problems and related methods will be introduced. Although there are many methods that exist in the literature, a selected few, that are the most widely used ones are included in this chapter.

### 3.1. Approximate Methods

The direct calculation of the probability integral given in equation (3.4) is not preferred in structural reliability analysis due to its challenging nature and the exact solution cannot be obtained in some cases. As an alternative to integration, the actual limit state function is approximated by using the Taylor series, and the probability of failure is calculated by using a reliability index. In this section, the methods based on Taylor series expansion will be introduced and two reliability index definitions that are most widely used will be given. At the end of the section, a critique of the presented methods will be provided in order to reveal the disadvantages of each method and determine which method is applicable to which type of problem. It is hoped that the critique section will be a guideline for the readers in the selection of appropriate method for their analysis.

#### 3.1.1. Cornell Reliability Index

The prior study made on the structural reliability analysis has been done by Cornell (1969) and the reliability index is defined as the distance to the mean value in terms of standard deviation. In Figure 3.1 a simple  $g(R, S) = R - S$  problem which has normally distributed basic variables for both  $R$  and  $S$  is illustrated.

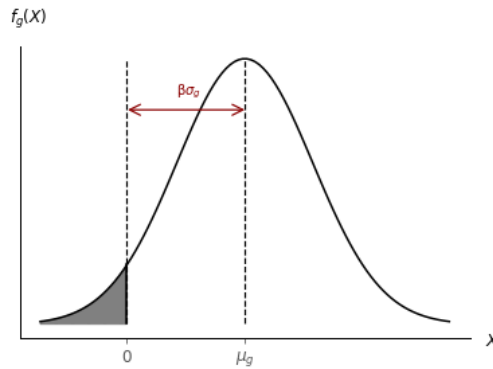


Figure 3.1. Cornell reliability index.

The corresponding probability of failure for this limit state function is equal to:

$$P_f = \Phi(-\beta) \quad (3.6)$$

Based on the definition, the reliability index,  $\beta$ , is equal to:

$$\beta = \frac{\mu_g}{\sigma_g} \quad (3.7)$$

where  $\mu_g$  is the mean value of the limit state function whereas  $\sigma_g$  represents the standard deviation of the limit state function.

The second moments of the limit state function can be determined by using the following relationships if both basic variables are normally distributed:

$$\mu_g = \mu_R - \mu_S \quad (3.8)$$

$$\sigma_g = \sqrt{\sigma_R^2 + \sigma_S^2} \quad (3.9)$$

where  $\mu_R$  and  $\mu_S$  are the mean values of the capacity and demand parameters whereas  $\sigma_R$  and  $\sigma_S$  represents the standard deviations of the capacity and demand parameters.

If equation (3.8) and equation (3.9) are substituted into equation (3.7), one can immediately obtain the reliability index as follows:

$$\beta = \frac{\mu_R - \mu_S}{\sqrt{\sigma_R^2 + \sigma_S^2}} \quad (3.10)$$

### 3.1.2. Extension to n-Dimensional Linear and Non-Linear Limit State Functions

The above-mentioned methodology can easily be extended to problems having n-dimensional limit state function with normally distributed uncorrelated random variables. If the linear limit state function is defined as below:

$$g(X) = g(X_1, X_2 \dots X_n) = a_0 + a_1X_1 + \dots + a_nX_n = a_0 + \sum_{i=1}^n a_iX_i \quad (3.11)$$

Then, the mean value and standard deviation of the given linear limit state function are equal to:

$$\mu_g = a_0 + a_1\mu_{X_1} + \dots + a_n\mu_{X_n} = a_0 + \sum_{i=1}^n a_i\mu_{X_i} \quad (3.12)$$

$$\sigma_g = \sqrt{a_1^2\sigma_{X_1}^2 + \dots + a_n^2\sigma_{X_n}^2} = \sqrt{\sum_{i=1}^n (a_i\sigma_{X_i})^2} \quad (3.13)$$

Therefore, the corresponding reliability index is equal to:

$$\beta = \frac{a_0 + \sum_{i=1}^n a_i\mu_{X_i}}{\sqrt{\sum_{i=1}^n (a_i\sigma_{X_i})^2}} \quad (3.14)$$

Most of the structural reliability problems in structural engineering include non-linear limit state function. Reliability analysis can be made for the problems that have non-linear limit state function by linearizing the existing limit state function at a certain point using first-order Taylor series expansion. The expansion can be made as follows:

$$g(X_1, X_2 \dots X_n) \approx g(x_1^*, x_2^*, \dots, x_n^*) + \sum_{i=1}^n (X_i - x_i^*) \left. \frac{\partial g}{\partial X_i} \right|_{x_1^*, x_2^*, \dots, x_n^*} \quad (3.15)$$

### 3.1.3. Mean Value First Order Second Moment Method (MVFOSM)

It is worth mentioning here that the expansion point has different names in the literature as “checking point”, “design point” and, “most probable point (MPP)”. Although different names are given, all of these names indicate the point where the Taylor series expansion is made. In this study, the design point will be used to indicate the expansion point. One option for the selection of the design point is the usage of mean values of the basic variables involved in the problem. Therefore, the following Taylor series expansion can be written when the mean values are used as the design point:

$$g(X_1, X_2 \dots X_n) \approx g(\mu_{X_1}, \mu_{X_2}, \dots, \mu_{X_n}) + \sum_{i=1}^n (X_i - \mu_{X_i}) \left. \frac{\partial g}{\partial X_i} \right|_{\mu_{X_1}, \mu_{X_2}, \dots, \mu_{X_n}} \quad (3.16)$$

Although this selection is very simple and practical to use, it introduces a serious problem in the result. It is observed that the calculated reliability index changes if the same limit state function is written in an equivalent form. This problem is named the “invariance problem” and it originated from the selection of the mean values as a design point. The invariance problem is solved by Hasofer and Lind (1974) and different algorithms are developed in the literature for the determination of the design point. Those algorithms will be introduced later in this section.

### 3.1.4. First-Order Reliability Method (FORM)

The First-Order Reliability Method (FORM) is a general methodology in the structural reliability analysis field that approximates the limit state function by using the first-order Taylor series. After the approximation is made, the Hasofer-Lind reliability index is used to create the link between the approximate function and the probability of failure. In this subsection, the Hasofer-Lind reliability index will be introduced, and then two similar algorithms to determine the design point will be investigated.

#### 3.1.4.1. Hasofer-Lind Reliability Index

The invariance problem observed in the Cornell reliability index was solved by Hasofer and Lind (1974) by transforming the basic variables involved in the problem to the standard normal distribution form and determining the design point by an iterative algorithm rather than using the mean values of the basic variables involved in the problem. The transformation of the basic variables from the original space, i.e. x-space, to uncorrelated and independent standard normal space, i.e. u-space, can be performed by using different transformation methods. Rosenblatt Transformation (Rosenblatt 1952) or Nataf Transformation (Nataf 1962) are widely used techniques for transformation purposes in reliability analysis.

In Figure 3.2, a two-dimensional limit state function in the x-space is shown on the left whereas the same limit state function in the u-space is illustrated in Figure 3.2 A limit state function in (a) x-space and (b) u-space. on the right.

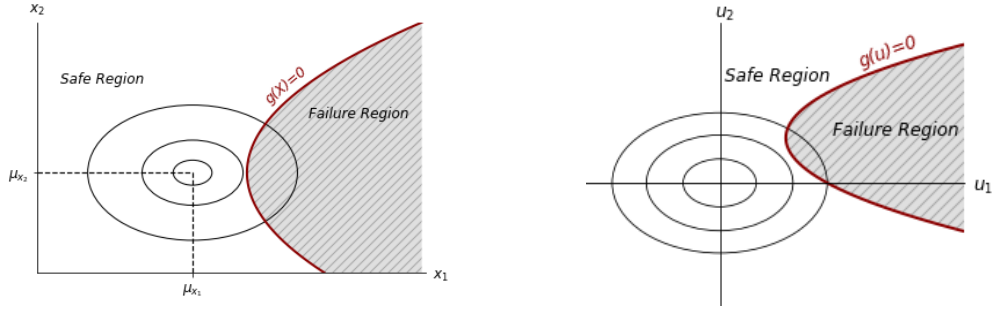


Figure 3.2 A limit state function in (a) x-space and (b) u-space.

For example, consider the limit state function given before for a simple R-S problem having normally distributed uncorrelated random variables. The random variables R and S can be transformed from the original space to standardized normal space can be done as follows if both variables are uncorrelated and independent:

$$u_1 = \frac{R - \mu_R}{\sigma_R} \quad (3.17)$$

$$u_2 = \frac{S - \mu_S}{\sigma_S} \quad (3.18)$$

Equivalently,

$$R = u_1 \sigma_R + \mu_R \quad (3.19)$$

$$S = u_2 \sigma_S + \mu_S \quad (3.20)$$

Then the limit state function becomes in the following form after the transformation:

$$g(u_1, u_2) = u_1 \sigma_R - u_2 \sigma_S + \mu_R - \mu_S \quad (3.21)$$

After the transformation of the basic variables and determining the limit state function in the standard normal space, the Hasofer-Lind reliability index can be determined by calculating the minimum distance from the limit state function and the origin of the standard normal space. Therefore, the Hasofer-Lind reliability index is essentially an optimization problem that minimizes distance, and can be written as follows:

$$\beta = \min_u \sqrt{u^T u} \text{ subjected to } g(u) = 0 \quad (3.22)$$

The optimization problem defined in equation (3.22) to obtain the reliability index can be solved by using programming techniques or other optimization methods. However, in the field of structural reliability, this problem is generally solved by using iterative techniques which will be introduced in the next subsections.

### 3.1.4.2. Hasofer-Lind (HL) Algorithm

This algorithm tries to minimize the distance between the design point and the origin of the standard normal space by successively performing iterations until a predefined tolerance for convergence is achieved. This iterative scheme can be performed by either solving  $2n+1$  simultaneous equations or using matrix algebra. The solution algorithm by using matrix algebra presented in Nowak and Collins (2000) is given below.

Step 1: Determine an arbitrary initial design point to start iterations and say  $\{x^*\} = \{x_1^*, x_2^*, \dots, x_n^*\}$ .

Step 2: Transform the basic variables to standard normal variables by using an appropriate transformation method and calculate the related design point in the standardized normal space.

$$\{u^*\} = \{u_1^*, u_2^*, \dots, u_n^*\} \quad (3.23)$$

Step 3: Determine the partial derivative of the limit state function in terms of standardized normal variables and store them in a vector as follows:

$$\{G\} = \begin{Bmatrix} G_1 \\ G_2 \\ \cdot \\ \cdot \\ G_n \end{Bmatrix} \text{ and } G_i = -\frac{\partial g}{\partial u_i^*} \Big|_{u_1^*, u_2^*, \dots, u_n^*} \quad (3.24)$$

Step 4: Calculate the new estimation for the reliability index by using the partial derivatives and design point.



$$\beta = \frac{\{G\}^T \{u^*\}}{\sqrt{\{G\}^T \{G\}}} \quad (3.25)$$

Step 5: Calculate the direction cosines and store them in a vector.

$$\{\alpha\} = \frac{\{G\}}{\sqrt{\{G\}^T \{G\}}} \quad (3.26)$$

Step 6: Determine a new design point by using the direction cosines and estimated reliability index.

$$\{u^*\} = \{\alpha\}\beta \quad (3.27)$$

Step 7: The location of design point in original space can be calculated by transforming back from the standard normal space if it is necessary.

### 3.1.4.3. Hasofer-Lind Rackwitz-Fiessler (HLRF) Algorithm

Rackwitz and Fiessler (1978) proposed an iterative algorithm to determine the Hasofer-Lind reliability index by considering the type of distribution of each basic variable explicitly. First, the equivalent normal parameters of each non-normal basic variable are calculated at the design point. In order to calculate the equivalent normal value, let's say the corresponding probability density function of the non-normal basic variable is  $f_X(x^*)$  and the cumulative density function is  $F_X(x^*)$  at the design point. The equivalent normal parameters can be calculated by equating the probability density function and cumulative density function of the original distribution to the probability density function and cumulative density function of the standard normal distribution at the design point. This equality can be written in the following mathematical form:

$$F_X(x^*) = \Phi\left(\frac{x^* - \mu_X^e}{\sigma_X^e}\right) \quad (3.28)$$

$$f_X(x^*) = \frac{1}{\sigma_X^e} \phi\left(\frac{x^* - \mu_X^e}{\sigma_X^e}\right) \quad (3.29)$$

From equation (3.28) and equation (3.29), the equivalent normal parameters are calculated as follows:

$$\mu_X^e = x^* - \sigma_X^e [\Phi^{-1}(F_X(x^*))] \quad (3.30)$$

$$\sigma_X^e = \frac{1}{f_X(x^*)} \phi[\Phi^{-1}(F_X(x^*))] \quad (3.31)$$

The only difference between the Hasofer-Lind algorithm and the HLRF algorithm comes from the inclusion of these equivalent normal parameters. Therefore, the following steps can be immediately deduced if this new step is included in the previously introduced algorithm.

Step 1: Determine an arbitrary initial design point to start iterations and say  $\{x^*\} = \{x_1^*, x_2^*, \dots, x_n^*\}$ .

Step 2: Calculate the equivalent normal parameters for each non-normal basic variable by using equation (3.30) and (3.31).

Step 3: Transform the basic variables to standard normal variables by using an appropriate transformation method and calculate the related design point in the standardized normal space.

$$\{u^*\} = \{u_1^*, u_2^*, \dots, u_n^*\} \quad (3.32)$$

Step 4: Determine the partial derivative of the limit state function in terms of standardized normal variables and store them in a vector as follows:

$$\{G\} = \begin{Bmatrix} G_1 \\ G_2 \\ \cdot \\ \cdot \\ G_n \end{Bmatrix} \text{ and } G_i = - \frac{\partial g}{\partial u_i^*} \Big|_{u_1^*, u_2^*, \dots, u_n^*} \quad (3.33)$$

Step 5: Calculate the new estimation for the reliability index by using the partial derivatives and design point.

$$\beta = \frac{\{G\}^T \{u^*\}}{\sqrt{\{G\}^T \{G\}}} \quad (3.34)$$

Step 6: Calculate the direction cosines and store them in a vector.

$$\{\alpha\} = \frac{\{G\}}{\sqrt{\{G\}^T\{G\}}} \quad (3.35)$$

Step 7: Determine a new design point by using the direction cosines and the estimated reliability index.

$$\{u^*\} = \{\alpha\}\beta \quad (3.36)$$

Step 8: The location of the design point in the original space can be calculated by transforming back from the standard normal space if it is necessary.

### 3.1.5. Second-Order Reliability Method (SORM)

Second-Order Reliability Method (SORM) is an improvement on FORM by including the second-order terms in Taylor series expansion in order to increase the accuracy of the probability estimates. Based on the definition, the general mathematical expression of SORM in the standard normal space can be written in matrix form as follows:

$$g(u) \approx g(u^*) + \{u_i - u_i^*\}^T \nabla g(u) \Big|_{u^*} + \frac{1}{2!} \{u_i - u_i^*\} \nabla^2 g(u) \Big|_{u^*} \{u_i - u_i^*\}^T \quad (3.37)$$

In equation (3.37),  $u^*$  represents the vector of design point  $\{u_1^*, u_2^*, \dots, u_n^*\}$  whereas  $\nabla g(u)$  is the vector of the first derivatives and  $\nabla^2 g(u)$  is  $n \times n$  matrix includes the second-order partial derivatives of the limit state function and named as Hessian and, generally denoted by H. The Hessian matrix is, then be, equal to the following expression:

$$\nabla^2 g(u) = H = \begin{bmatrix} \frac{\partial^2 g}{\partial u_1^2} & \frac{\partial^2 g}{\partial u_1 u_2} & \dots & \frac{\partial^2 g}{\partial u_1 u_{n-1}} & \frac{\partial^2 g}{\partial u_1 u_n} \\ \frac{\partial^2 g}{\partial u_2 u_1} & \dots & \dots & \dots & \frac{\partial^2 g}{\partial u_2 u_n} \\ \vdots & \dots & \dots & \dots & \vdots \\ \frac{\partial^2 g}{\partial u_n u_1} & \frac{\partial^2 g}{\partial u_n u_2} & \dots & \frac{\partial^2 g}{\partial u_n u_{n-1}} & \frac{\partial^2 g}{\partial u_n u_n} \end{bmatrix} \quad (3.38)$$

The corresponding probability of failure of the given approximate limit state function can be calculated by using the Hasofer-Lind reliability index concept. However, the solution process of SORM is much more different than FORM due to the existence of the second-order terms. Therefore, the iterative methods introduced earlier cannot be employed herein. On the other hand, a quadratic approximation can be made in the form of a paraboloid, ellipsoid, or hypersphere. After the selection of the form, the probability of failure of the corresponding function should be evaluated. However, it is necessary to rotate the coordinate axes in order to avoid evaluating complicated calculations of the probability content of the selected quadratic form (Fiessler, Neumann, and Rackwitz 1979). The rotation of the axes, in most cases, is performed by coinciding with the last coordinate of the new coordinate system and the location of the design point. For this purpose, orthogonal transformation is used (Choi, Grandhi, and Canfield 2007).

After rotating the coordinate axes, a parabolic approximation can be done in the new rotated space. To calculate the probability content of the approximating parabola, Breitung (1984) developed an asymptotic method based on the main curvatures of the approximated parabola and the reliability index calculated by using FORM. The derived equation is given below:

$$P_f \approx \Phi(-\beta) \prod_{i=1}^{n-1} (1 + k_i \beta)^{-\frac{1}{2}} \quad (3.39)$$

where  $k_i$  is the main curvature of the failure surface at the design point and  $\beta$  is the reliability index calculated by FORM

Please note that the estimated probability of failure is exact when  $\beta$  approaches infinity due to the asymptotic approximation. On the other hand, the method is not applicable when  $k_i \beta$  equals or less than -1. Although the formulation looks simple, the main problem in this method is the calculation of the main curvatures. Because it requires orthogonal transformation and the calculation of the second-order derivatives, these calculations can be prohibitive if the dimension of the problem is large. Additionally, the accuracy of the formulation is highly dependent on the radii of the curvature of the actual limit state function.

Tvedt (1983) proposed the formulation given in equation (3.40) that consists of three-term to improve the accuracy of the approximation. The first term of the formulation is the same as Breitung's formulation and the other two terms are the corrector terms which improve the accuracy. Nevertheless, the computational effort made in the analysis is high compared to Breitung's method due to the corrector terms.

$$P_f \approx A_1 + A_2 + A_3 \quad (3.40)$$

where,

$$A_1 = \Phi(-\beta) \prod_{i=1}^{n-1} (1 + k_i \beta)^{-\frac{1}{2}} \quad (3.41)$$

$$A_2 = [\beta \Phi(-\beta) - \phi(-\beta)] \left\{ \prod_{i=1}^{n-1} (1 + k_i \beta)^{-\frac{1}{2}} - \prod_{i=1}^{n-1} [1 + (\beta + 1)k_i]^{-\frac{1}{2}} \right\} \quad (3.42)$$

$$A_3 = (\beta + 1)[\beta \Phi(-\beta) - \phi(-\beta)] \left\{ \prod_{i=1}^{n-1} (1 + k_i \beta)^{-\frac{1}{2}} - Re \left[ \prod_{i=1}^{n-1} [1 + (\beta + i)k_i]^{-\frac{1}{2}} \right] \right\} \quad (3.43)$$

The above-mentioned formulations are based on curvature fitting and they require complicated processes of orthogonal transformation and calculation of the Hessian matrix in order to coincide with the principal curvatures of the actual limit state function and second-order approximation (Zhao and Ono 1999a). In order to eliminate the calculation of costly Hessian matrix, Kiureghian et al. (1987) developed a new method by constructing a parabola based on the selected points. However, this method is less accurate than the standard curvature fitting methods because of neglecting cross-derivative terms (Kiureghian et al. 1987).

The development of the second-order methods in structural reliability analysis is still open to research and discussion, and more detailed information on SORM can be found in the study of Köylüoğlu and Nielsen (1994), Hu et al. (2021), der Kiureghian and Dakessian (1998), Zhao and Ono (1999b) and Breitung (2021).

### **3.1.6. Critical Appraisal on Approximate Methods**

Approximate methods are extremely useful structural reliability methods in terms of efficiency. In general, only a few iterations are needed for FORM or SORM calculations for low-dimensional reliability problems. However, the main problem with the approximate methods is the level of accuracy due to linear/quadratic approximation to the limit state function. The calculated probability of failure is exact only if the limit state function is linear for FORM and quadratic for SORM, beyond these types of functions these methods can overestimate or underestimate the probability of failure based on the shape of the limit state function.

Moreover, the approximate methods have no convergence guarantee. The iterative process can fail for multiple limit state functions or having highly non-linear limit state functions. On top of that, the limit state function must be differentiable in order to employ an iterative scheme.

Furthermore, the approximate methods are not always feasible with finite element analysis. First of all, the implicit limit state function representing the finite element analysis must be smooth and there must be no detrimental numerical noise. Even under these conditions, the limit state function must not be highly non-linear. If all these three conditions are met, then FORM can be applied for static finite element analysis and even though the conditions are met, FORM is not recommended for dynamic finite element analysis (Koduru and Haukaas 2010). The same issues can also be considered for SORM if the reasons presented above for FORM are extended.

Real-life engineering problems are solved by using finite element analysis these days. Therefore, if we want to apply structural reliability methods in the industry, then the developed methods should be compatible with finite element analysis. As mentioned above, applying FORM to finite element analysis properly requires prior knowledge of the situation of the limit state function. However, it must be recognized that an analyst/engineer has no prior knowledge at the beginning of the analysis.

## 3.2. Simulation Methods

Simulation methods are very powerful tools to perform structural reliability analysis, especially if the limit state function is implicit and highly nonlinear. The simulation methods are immune to the shape of the function. Therefore, having a convex or concave limit state function does not affect the result. However, the performance of the simulation methods is directly related to the number of samples which is related to the magnitude of the probability of failure. The results obtained from the simulation methods can be deceiving in the case of not having enough samples, or the number of samples can be excessively high for a very small probability of failure. Hence, an empirical relationship between the number of samples and the probability of failure will be given in this section and the most widely used sampling strategies will be introduced in order to ease the computational burden of the simulation methods.

### 3.2.1. ‘Crude’ Monte Carlo Simulation (CMCS)

Monte Carlo simulation is based on the successive random sampling of the generated data which is generated by using the statistical properties of the basic variables involved in the problem. The sampled data are then transferred to the limit state function and the evaluation of the limit state function is performed. An indicator function is used to classify the results based on the observation of failure or safe. The value of the indicator function is taken as one if the failure occurs. After that, the number of observations of failure is summed and divided by the total number of samples in order to determine the probability of failure of the given structural reliability problem.

Therefore, the probability integral given in equation (3.4) is approximated by using the following relationship:

$$P_{f_{True}} = \int \dots \int I[g(X) < 0] f_X(X) dX \cong P_f = \frac{1}{N} \sum_{i=1}^N I[g(X_i) < 0] \quad (3.44)$$

where,  $N$  is the total number of samples and  $I[\ ]$  is the indicator function that can take values only one and zero depending on satisfaction of the condition inside.

The exact probability of failure of the given event by using MCS can be achieved if the total number of samples reaches infinity but imposing such a number of samples on analysis is impossible in practice. Nevertheless, an approximate formulation for the required number of samples with a certain variance in results can be achieved by performing repetitive simulations for the problem at hand with the same number of samples. In each simulation, different results will be obtained due to having a finite number of samples. Hence, the probability estimate performed by using MCS is also a random variable having binomial distribution. Therefore, the governing equation for the coefficient of variance,  $V_{P_f}$ , of the results is given in equation (3.45) by using the true probability of failure and the number of samples (Lemaire, Chateauneuf, and Mitteau 2009).

$$V_{P_f} = \sqrt{\frac{1 - P_{fTrue}}{NP_{fTrue}}} \quad (3.45)$$

The required number of samples can be immediately calculated for a certain level of coefficient variation by rearranging equation (3.45) as follows:

$$N = \frac{1 - P_{fTrue}}{V_{P_f}^2 P_{fTrue}} \quad (3.46)$$

If a rough estimation is made for the magnitude of the probability of failure for a given problem as  $10^{-3}$  and the desired coefficient of variation is determined as 0.1, then the number of required samples is equal to 99900. The required number of samples quickly escalates to 999900 for the problems that have a  $10^{-4}$  magnitude of the probability of failure with the same coefficient of variation. Moreover, a further simplification can be made as shown in equation (3.47) for the required number of samples if the magnitude of the probability of failure approaches zero.

$$N = \frac{1}{V_{P_f}^2 P_{fTrue}} \quad (3.47)$$

Please note that if the number of samples used in the simulation is decreased then the variation in results starts to increase. This trade-off can lead to serious problems when the evaluation of the limit state function is costly. Therefore, different sampling strategies



are developed based on this “Crude” Monte Carlo simulation (CMCS) procedure in order to decrease the number of samples by keeping the variance at an acceptable level.

### 3.2.2. Monte Carlo Simulation with Importance Sampling (IS)

One way of reducing the required number of samples by providing a certain variance in results can be achieved by using the Importance Sampling (IS) technique when the magnitude of the probability of failure is low. The basic motivation behind this technique is shifting the sampling region from the whole domain to the region that contributes more to the probability of failure. Then, the estimate for the probability of failure can be calculated by using the following formulation:

$$P_{f_{True}} = \int \dots \int I[g(X) < 0] \frac{f_X(X)}{h_V(X)} h_V(X) dX \quad (3.48)$$

where  $h_V(V)$  is the importance sampling probability density function.

The expression given above can be rearranged as follows in order to provide a discrete form for simulation purposes.

$$P_{f_{True}} \cong P_f = \frac{1}{N} \sum_{i=1}^N I[g(V_i) < 0] \frac{f_X(V_i)}{h_V(V_i)} \quad (3.49)$$

where,  $V$  is the samples taken from the importance sampling probability density function.

There are two key points that exist in this technique that impact the efficiency and accuracy of the obtained results. One of them is the location of the sampling points. Generally, the vicinity of the design point is selected to perform sampling due to its high probability content with respect to the standard sampling. The location of the design point can be found by performing a FORM analysis. The second key point is the distribution of the selected sampling function. The distribution can be made by equating the mean value of the sampling function to the design point and a significant decrease in the required number of samples can be obtained (Harbitz 1983). The derivation for an optimum selection of the sampling function can be found in the book of Melchers and Beck (2018). However, generally Normal distribution function is used as the sampling function due to its simplicity.

### 3.2.3. Subset Simulation (SS)

The required number of samples increases exponentially in CMCS dependent on the magnitude of the probability of failure of the problem at hand. Therefore, for the small probabilities, excessive amount of samples is needed to achieve a stable probability estimation with an acceptable variance. Additionally, even though the CMCS is immune to dimensionality, the IS suffers from the curse of dimensionality indirectly due to the need for prior FORM analysis in order to determine a design point. In order to reduce the required number of samples for those types of problems, Au and Beck (1999) developed Subset Simulation (SS) technique. In this technique, a finite number of intermediate failure events is used to calculate the small probability of failure. Generally, those intermediate failure events are defined as  $F_i = g(X) < b_i$  where  $b_i$  is a certain threshold and the events are ordered in a decreasing fashion as  $F_1 \supset F_2 \supset \dots \supset F_m$  so that any  $k^{th}$  event can be written as  $F_k = \bigcap_{i=1}^k F_i$ . Therefore, the intersection of all intermediate events yields an estimation for the probability of failure of the actual event. Thus, the given equation below is the main idea of the SS.

$$P_f = P(F_m) = P\left(\bigcap_{i=1}^m F_i\right) \quad (3.50)$$

The probability calculation for the intersection of events can be accomplished by using the conditional probabilities as follows:

$$P_f = P(F_m|F_{m-1})P\left(\bigcap_{i=1}^{m-1} F_i\right) \quad (3.51)$$

A further simplification in the equation given above can be performed by benefitting from the multiplication of conditional events:

$$P_f = P(F_1) \prod_{i=2}^m P(F_i|F_{i-1}) \quad (3.52)$$

Consequently, the actual event is divided into intermediate events which are larger probabilities compared to the actual event. However, it is necessary to generate conditional samples to perform the analysis. Fortunately, those conditional samples can

be generated by using the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970) which is a variant of Markov Chain MCS (MCMCS).

### 3.2.3.1. Implementation of Subset Simulation

There are two main important steps that should be handled in the implementation of SS. The first step is the determination of the intermediate events and the second one is the generation of the conditional samples.

The determination of the first intermediate event can be done by predefining a target probability,  $p_0$ . Then, generate  $N$  number of samples and evaluate the limit state function. After that, equate the  $[(1 - p_0)N]$ -th largest value to  $b_1$  which is the threshold for the first intermediate event. Therefore, in total  $p_0N$  samples are belonged to the first intermediate event and based on these samples, new  $(1 - p_0)N$  samples, which are conditional on  $F_1$ , are generated by using the MCMCS. Repeating this method until all of the events yield the probability of failure of the actual event. The noteworthy point here is that all intermediate events will have the probability of  $p_0$  and only the last event might have a different probability if  $p_0$  is kept fixed. Thus, the probability of failure up to  $m - 1$  will be equal to  $p_0^{m-1}$ . As a consequence, the estimate for the probability of failure of the actual event can be written as follows:

$$P_f \cong p_0^{m-1} P(F_m | F_{m-1}) \quad (3.53)$$

In the equation given above  $P(F_m | F_{m-1})$  can be obtained approximately by using the CMCS. In fact, the probability of intermediate events except for the last one is not exactly equal to  $p_0$ . Hence, the probability of failure estimation by this approach is not exact, however, is approximate. Additionally, the target probability and the number of samples might be different for each intermediate step (Papaioannou 2012). The typical interval for the target probability is suggested between 0.1 and 0.3 (Zuev et al. 2012). The MATLAB implementation of SS is provided in the study of H. S. Li and Cao (2016).

### 3.2.4. Directional Simulation (DS)

Directional Simulation (DS) is a technique based on performing simulation by distributing direction vectors on a unit hyper-sphere. It was first developed by Deák (1980) to calculate the multiple normal probability integrals. Then, it is applied to load space by Ditlevsen, Olesen, and Mohr (1987). After that, Bjerager (1988) applied DS for reliability problems and combined it with IS technique to improve its efficiency. Later on, Ditlevsen, Melchers, and Gluwer (1990) proposed a generalized version of DS that is valid for different types of joint distribution functions and complicated limit state functions.

The general procedure for DS given in the study of Bjerager (1988) starts with the generation of  $n$ -dimensional Gaussian vector  $u$  which is defined as  $u = RA$  where  $R^2$  is a chi-squared random variable with  $n$  degrees of freedom and  $\mathbf{A}$  is a random direction vector, on a unit hyper-sphere  $\Omega_n$ . Please notice that  $u$  is a vector in polar coordinates due to the direction vector. In the simplest way,  $\mathbf{A}$  is distributed uniformly, however, more advanced distribution techniques can be found in the study of Nie and Ellingwood (2000). If the condition  $\mathbf{A} = \mathbf{a}$  is set, then the associated probability of failure can be calculated as given below:

$$P_f = \int_{\mathbf{a} \in \Omega_n} P[g(R\mathbf{A}) \leq 0 | \mathbf{A} = \mathbf{a}] f_{\mathbf{A}}(\mathbf{a}) d\mathbf{a} = \int_{\mathbf{a} \in \Omega_n} P[g(R\mathbf{a}) \leq 0] f_{\mathbf{A}}(\mathbf{a}) d\mathbf{a} \quad (3.54)$$

where  $f_{\mathbf{A}}(\mathbf{a})$  is the probability density of  $\mathbf{A}$  on the unit hyper-sphere.

If  $N$  number of  $\mathbf{a}_i$  is generated, then the probability of failure can be estimated as:

$$P_f \cong \widehat{P}_f = \frac{1}{N} \sum_{i=1}^N P[g(R\mathbf{a}_i) \leq 0] \quad (3.55)$$

Thus, the estimated standard deviation is equal to:

$$\sigma_{\widehat{P}_f} = \frac{1}{N(N-1)} \sum_{i=1}^N (P[g(R\mathbf{a}_i) \leq 0] - \widehat{P}_f)^2 \quad (3.56)$$

For a given  $\mathbf{a}$  the probability of  $g(R\mathbf{a}) \leq 0$  can be calculated analytically as follows:

$$P[g(R\mathbf{a}) \leq 0] = P[R \geq r] = 1 - \chi_n^2(r^2) \quad \text{for } r \geq 0 \quad (3.57)$$

where  $r$  is defined as the root which satisfies  $g(r\mathbf{a}) = 0$ .

### 3.2.4.1. Importance Sampling in Directional Simulation

The efficiency of the DS can be further improved by performing importance sampling on  $\mathbf{A}$ . For a selected importance sampling function, the formulation given in equation (3.54) turns into the following form:

$$P_f = \int_{\mathbf{b} \in \Omega_n} P[g(R\mathbf{b}) \leq 0] \frac{f_A(\mathbf{b})}{h_B(\mathbf{b})} h_B(\mathbf{b}) d\mathbf{b} \quad (3.58)$$

Then, the probability estimate is rewritten as:

$$P_f \cong \widehat{P}_f = \frac{1}{N} \sum_{i=1}^N P[g(R\mathbf{b}_i) \leq 0] \frac{f_A(\mathbf{b}_i)}{h_B(\mathbf{b}_i)} \quad (3.59)$$

### 3.2.5. Critical Appraisal on Simulation Methods

Simulation methods generally do not suffer from the accuracy issues that exist in the approximate methods. However, they also have several important drawbacks in their implementation to the structural reliability problems related to computational cost or implementation challenges.

First of all, CMCS is a robust method for any type of problem. It is not affected by the non-linearity of the limit state function. On the other hand, the computational cost is directly related to the magnitude of the probability of failure. For very small probabilities, the computational cost can increase exponentially and the method might become impractical to use for real-life structural engineering problems.

As previously introduced, IS can be used to decrease computational cost. In spite of that, it requires a design point to perform sampling and generally, a prior analysis is performed to find the point. At this point, please remember that FORM is generally used to find the design point but it has convergence issues. Therefore, when FORM does not converge to a design point, implementing IS can be difficult or not feasible. Additionally, if FORM converges to an erroneous design point, then results obtained from IS can be misleading.

Moreover, in order to implement SS to reliability problems, it is necessary to calculate conditional probabilities. As mentioned previously, MCMCS can be used for conditional probabilities. However, performing MCMCS with finite element analysis can be challenging.

Implementing DS to reliability analysis problems requires the distribution of directional vectors on the surface of a unit hypersphere. Even though there are several methods that exist in the literature, the methods generally need special care and attention which makes the implementation challenging.

### **3.3. Surrogate Models**

In the structural reliability context, an expensive limit state function is replaced with a representative function - that is cheap to evaluate - in order to reduce the computation effort in the analysis. The representative function is generally called the surrogate model and it is constructed by using data points generated from the original limit state function, and it can be an explicit or implicit function depending on the used method. Although there are various methods that exist in the literature to construct a surrogate model, only the Response Surface Method (RSM) will be introduced in this study because of its wide usage.

Artificial Neural Network (ANN) can also be used to construct a surrogate model that is used for structural reliability analysis purposes. However, ANN will not be covered in this section. Instead, the next chapter will be devoted to ANN to investigate it in detail.

### 3.3.1. Response Surface Method (RSM)

Response Surface Method (RSM) is a surrogate modeling technique by approximating the limit state function at a local point by using low-order polynomials in order to reduce the computational cost. The main advantage of RSM is that the method can be used with finite element analysis with a relatively low computational with respect to the simulation methods by providing a satisfactory level of accuracy. One of the early applications of the method to structural reliability problems was introduced by Faravelli (1989). In the application, the limit state function was approximated by using a second-order polynomial without cross-terms. Therefore, the polynomial that is used in the analysis has the following form:

$$\bar{g}(x) = a + \sum_{i=1}^n b_i x_i + \sum_{i=1}^n c_i x_i^2 \quad (3.60)$$

where,  $x_i$  from 1 to  $n$  represents the number of random variables involved in the problem and  $a$ ,  $b_i$ , and  $c_i$  coefficients of the polynomial. In order to determine the coefficients of the approximating polynomial, samples obtained from evaluation of the actual limit state function are used.

In the response surface methodology, there are various ways to obtain samples from the limit state function called “*experiment design*”. Examples of design strategies include full-factorial design, central composite face-centered design, and Box-Behnken design. Typical representations of the above-mentioned design strategies are given in Figure 3.3.

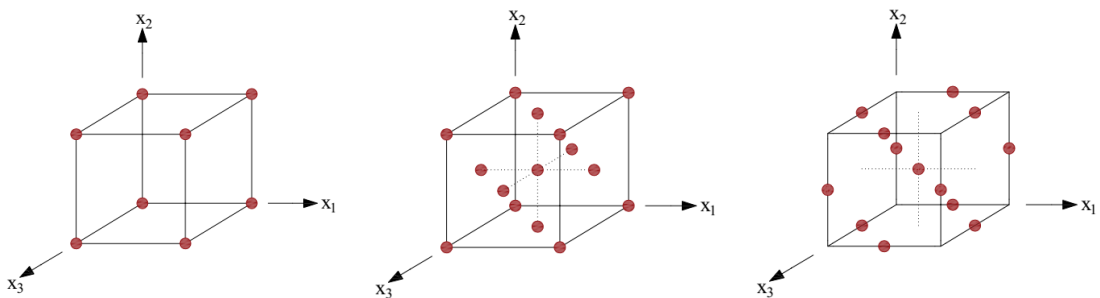


Figure 3.3. Different design strategies in RSM. Full-factorial design (left), central composite face-centered design (middle), Box-Behnken design (right).

The suggested methodology for reliability analysis was further improved by Bucher and Bourgund (1990). In the first step, it is suggested the response surface can be constructed around the mean values,  $\mu_i$ , and points  $\mu_i \pm f_i \sigma_i$  where  $\sigma_i$  is the standard deviation of each random variable and  $f_i$  is an arbitrary factor, generally taken as 3.

Then, an estimate for a design point is determined by using the constructed response surface. The design point is used with the constructed response surface in order to calculate a new center point for constructing an updated response surface. The main idea behind the update is to locate the new response surface close as possible to the point where the exact limit state function is equal to zero in each update in order to cover the failure domain adequately for the probability estimate. The key assumption in the application here is that the random variables are uncorrelated Gaussian variables. The new center point based on the update can be found by using the following relationship.

$$x_M = \bar{x} + (\bar{x}_D - \bar{x}) \frac{g(\bar{x})}{g(\bar{x}) - g(\bar{x}_D)} \quad (3.61)$$

However, this update increases the total number of evaluations from  $2n+1$  to  $4n+3$ . Additional improvement for the method was suggested by introducing the cross-terms to the approximating polynomial as follows:

$$\bar{g}(x) = a + \sum_{i=1}^n b_i x_i + \sum_{i=1}^n c_i x_i^2 + \sum_{\substack{i=1 \text{ and } j=1 \text{ and} \\ i \neq j}}^n \sum_{\substack{j=1 \text{ and} \\ j \neq i}}^n d_{ij} x_i x_j \quad (3.62)$$

Depending on the design method used to construct the response surface, the number of evaluations can increase up to  $2^n + 2n + 1$  function evaluations which makes the method inefficient for large systems.

The ideas suggested by Bucher and Bourgund (1990) are investigated further by Rajashekhar and Ellingwood (1993). In the first place, it is suggested that the arbitrary factor,  $f_i$ , can be reduced for further updates in order to avoid an ill-conditioned system of equations. In addition, the effect of the distribution tails in the samples was explicitly considered. Furthermore, the effect of cross-terms was also regarded.



The main drawback of the response surface method is the local approximation to the limit state function. Therefore, the constructed surface does not represent the behavior of the limit state function for a large range which means that when the interested region in the limit state function is changed, the surface must be constructed all over again in order to perform reliability analysis. Moreover, the constructed response surface is sensitive to the arbitrary factor,  $f_i$ , used in the experimental design and this can lead to fluctuations in the probability estimate (Guan and Melchers 2001).

## CHAPTER 4

### ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) are universal function approximators that mimic biological neuron behavior. They consist of interconnected artificial neurons that receive inputs and transform them into output by using weights, biases, and transfer function. The sequential combination of these neurons creates a network that is capable of approximating complex functions. Their capability of approximating any kind of function is proven by Hornik, Stinchcombe, and White (1989) when certain conditions are met. Although ANNs were originally accepted as function approximators, their capability is not limited only to that. ANNs can be used for classification problems, pattern recognition, language processing, and cluster analysis in many subjects.

In this thesis, feed-forward neural networks will be used to create a surrogate model for the limit state function for the given problem. The supervised learning paradigm will be followed for the training of the networks. Bayesian regularization will be used in this study as the training algorithm of neural networks even though there are different training methods and algorithms that exist for the neural networks.

In this chapter, basic information on neural networks will be given in order to provide insight to the reader. The discussion will start with the definition of a simple neuron and continues until the derivation of Bayesian regularization. The given information is kept limited deliberately due to providing only necessary knowledge about the neural networks for the scope of this thesis and the mathematical derivations are tried to be kept simple as possible while protecting their main purpose.

#### 4.1. Simple Artificial Neuron

Artificial neurons are basic elements of an artificial network and they mimic the biological neuron structure. The biological neurons take inputs via dendrites, then process the inputs in the cell body and generate an output. Artificial neurons take inputs and

manipulate them by using weights, bias, and a transfer function in order to generate an output. Therefore, an artificial neuron basically consists of weights, bias, and a transfer function. The analogy between biological and artificial neurons is illustrated in Figure 4.1.

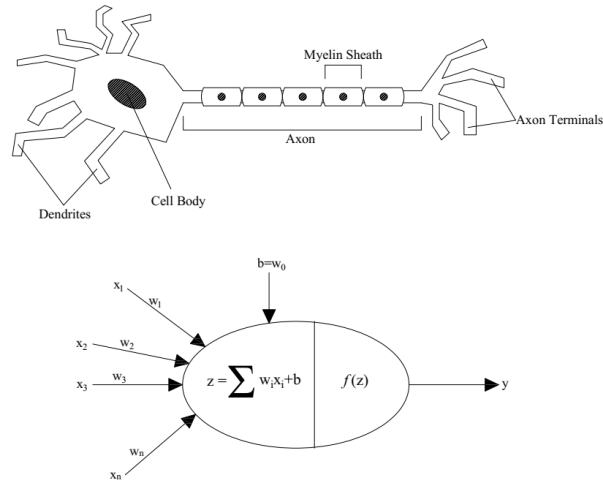


Figure 4.1. Biological neuron vs. artificial neuron.

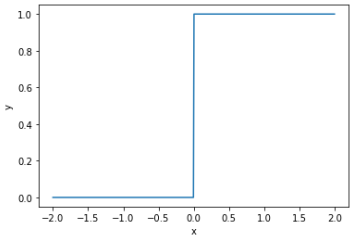
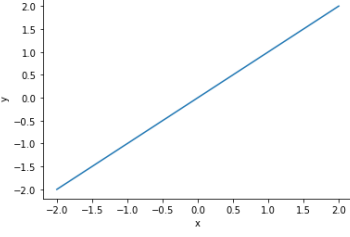
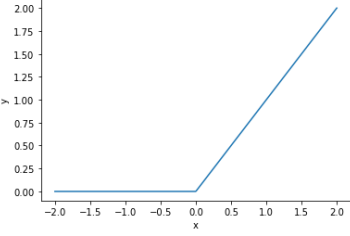
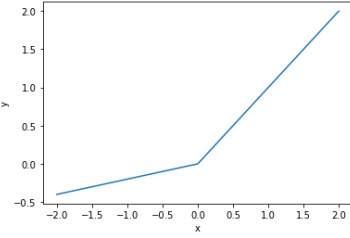
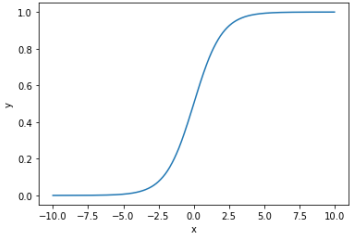
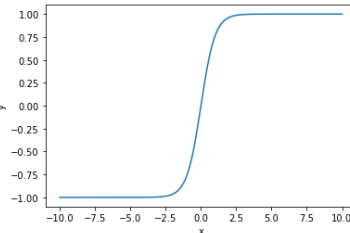
The relationship between the inputs,  $x_i$ , and output,  $y$ , in a simple artificial neuron can be written mathematically as follows:

$$y = f \left( \sum_{i=1}^n w_i x_i + b \right) \quad (4.1)$$

where,  $w$  represents the network weights whereas  $b$  represents the bias terms. The transfer function is represented by  $f(\dots)$ .

The transfer function is a crucial component of an artificial neuron due to its direct impact on the output. Even though any function can be used as a transfer function, there are commonly used transfer functions, some of which are given below.

Table 4.1. List of some transfer functions.

Name	Function	Plot
Binary Step Function	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Pure Linear Function	$f(x) = x$	
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$	
Logistic Sigmoid (Log-sig)	$f(x) = \frac{1}{1 + e^{-x}}$	
Hyperbolic Tangent (Tanh)	$f(x) = \frac{2}{1 + e^{-2x}} - 1$	

## 4.2. Architecture of Artificial Neural Networks

An artificial neural network is basically a system of artificial neurons that are connected to each other. Artificial neural networks can be divided into two categories as feed-forward neural networks and recurrent (feedback) neural networks based on the flow of inputs through the network. These two categories have also sub-branches and some of them are depicted in Figure 4.2.

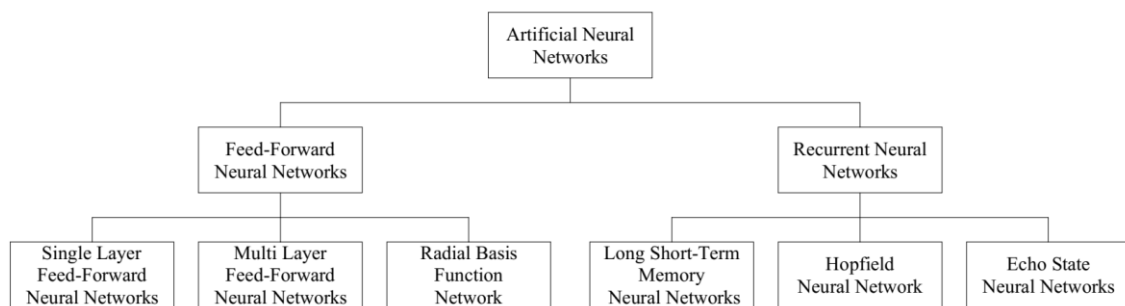


Figure 4.2. Different neural network architectures.

The difference between the feed-forward neural network and the recurrent neural network is the existence of a feedback mechanism for the artificial neurons. There are no feedback connections that exist in the feed-forward neural network and inputs flow through only unidirectional connections whereas recurrent neural networks have such connections and they can have multidirectional connections. Those multidirectional connections provide a dynamic behavior for recurrent neural networks (Jain, Mao, and Mohiuddin 1996).

## 4.3. Learning Paradigms in Artificial Neural Networks

The artificial neural networks are expected to produce desired output based on the provided inputs and they can learn the behavior of a function based on the labeled inputs and/or outputs. There are mainly three paradigms that exist in the literature for the learning paradigm. They are namely, supervised learning, unsupervised learning, and reinforced learning.

In supervised learning, both labeled inputs and outputs are provided to the network in the training phase. Therefore, the network can learn from the dataset at hand. This learning paradigm can be used effectively for function-approximating purposes. In unsupervised learning, only inputs are provided to the network and it is expected from the network to find patterns and matches for inputs and outputs. This type of paradigm is generally used for clustering problems. The reinforced learning paradigm is an intermediate level of supervised and unsupervised paradigms. In the reinforced learning paradigm, inputs are provided to the network, and the network is rewarded or punished based on its performance by the algorithm. This rewarding and punishing algorithm provides a guide to the network to increase its performance.

#### 4.4. Feed-Forward Neural Networks

Feed-forward neural networks consist of artificial neurons in a consecutive layered fashion. There is no interconnection between the neurons of the same layer. Generally, feed-forward neural networks are divided into three groups called single-layer feed-forward neural networks, multi-layer feed-forward neural networks, and radial basis function networks based on the number of layers or type of the function used in the network. The single-layer feed-forward neural networks and multi-layer feed-forward neural networks are also called single-layer perceptron and multi-layer perceptron respectively in the literature. Their schematic view is provided in Figure 4.3.

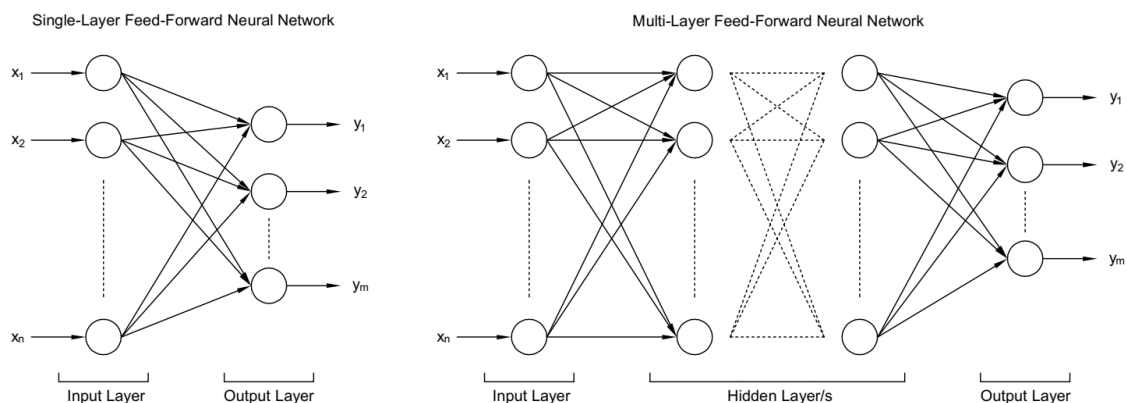


Figure 4.3. Configuration of single-layer feed-forward neural network and multi-layer feed-forward neural network.

The single-layer feed-forward neural networks consist of an input layer and an output layer whereas the multi-layer feed-forward neural networks have intermediate layers called hidden layers. There might be more than one number of hidden layers in the multi-layer feed-forward neural network and the number of hidden layers represents the depth of the network. The number of neurons in each hidden layer can be determined by using the trial-and-error method based on the convergence criterion.

The connection between the layers is provided by weights and biases. These terms are adjusted by using a proper training algorithm in order to obtain a useful neural network.

## **4.5. Training Algorithms**

Training a neural network is basically adjusting the weight vector and bias term of each layer systematically in order to provide a capability to the network to perform reasonable predictions. The adjustment process is generally carried out by minimizing a cost function which is a function that compares predictions generated by the network and actual output in the supervised learning paradigm. Even though various algorithms exist in the literature, the gradient descent algorithm with the back-propagation method is a widely used algorithm for training feed-forward neural networks due to its efficiency in computation and it constitutes the basics of neural network training. Therefore, it must be understood well in order to grasp the more advanced learning algorithms.

### **4.5.1. Back Propagation Method and Gradient Descent Algorithm**

Back-propagation is actually a method to calculate the gradients of a cost function with respect to the network parameters. However, in the machine learning community, the back-propagation term is used loosely like a learning algorithm. In general, the calculated gradients by the back-propagation method are used in the gradient descent algorithm in order to update network parameters.

At the beginning of the training, the weight matrix and bias vector can be randomly assigned small numbers to start the process and the input signal propagates through the network from the input layer to the output layer in the forward direction. This

first stage is called the forward phase due to the direction of the flow of the input signals. At the end of this stage, an error between the predictions and the desired outputs occurs. The cost function can be evaluated by using the error.

In the second stage, the gradients of the cost function are calculated with respect to each weight term in the network by using the chain rule. This stage propagates in the backward direction; therefore, it is called backpropagation.

In the last stage, the weight and bias terms are updated by using the gradient descent algorithm.

Consider an ideal three-layer feed-forward neural network given in Figure 4.4 in order to derive equations for the algorithm. Let's say the network has a fixed structure with  $i$  number of inputs,  $j$  number of neurons in the hidden layer, and yields  $k$  number of outputs.

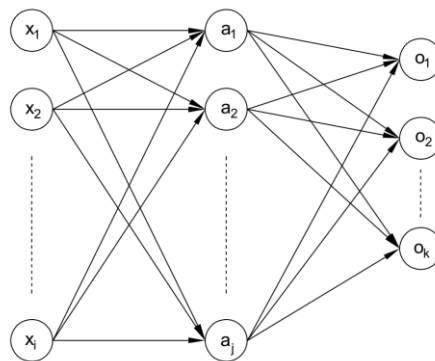


Figure 4.4. An ideal three-layer feed-forward neural network.

The propagation of the input through the network which is the first stage of the algorithm can be illustrated in a simple manner like in Figure 4.5.

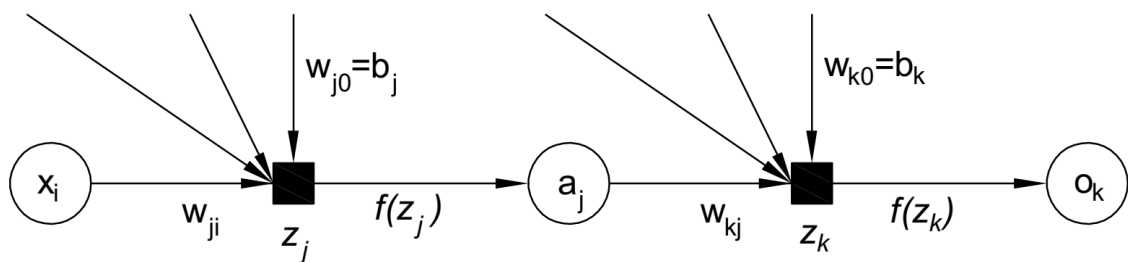


Figure 4.5. Simple representation of the flow of inputs in the  $j^{\text{th}}$  neuron.



Each input parameter from  $x_1$  to  $x_i$  is multiplied with the related weights and, consider that the bias term is embedded into the weight vector. Let say,

$$z_j = \sum_i w_{ji} x_i \quad (4.2)$$

Then, the signal in the  $j^{\text{th}}$  neuron of the hidden layer becomes,

$$a_j = f(z_j) \quad (4.3)$$

where  $f(.)$  represents the transfer function. The signal in the neurons of the hidden layer propagates to the output layer in a similar manner.

$$z_k = \sum_j w_{kj} a_j \quad (4.4)$$

Therefore, the output at the  $k^{\text{th}}$  neuron becomes equal to

$$o_k = f(z_k) \quad (4.5)$$

Now, in the second stage, assume that there are only 2 output neurons. If the cost function is considered as the mean squared error of a certain set of inputs:

$$L = \frac{1}{2} \sum_k (y_k - o_k)^2 \quad (4.6)$$

where  $y_k$  represents the desired or actual output whereas  $o_k$  is the predicted output by the neural network.

Remember that the main effort in training neural networks is to minimize the function  $L$  as much as possible. As it can also be deducted from the first stage, if the type of the transfer function and inputs are kept constant, the function  $L$  can only be changed by changing the layers' weights. Therefore, the change in the cost function,  $L$ , with respect to the weights of the output layer can be calculated by using the chain rule as shown in the equation below.

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}} \quad (4.7)$$

The partial derivative of  $z_k$  with respect to  $w_{kj}$  is nothing but  $a_j$ . Then, the equality becomes in the form of:

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial z_k} a_j \quad (4.8)$$

Use the chain rule again for the partial derivative of the cost function with respect to the  $z_k$ ,

$$\frac{\partial L}{\partial z_k} = \frac{\partial L}{\partial o_k} \frac{\partial o_k}{\partial z_k} \quad (4.9)$$

Please notice that the partial derivative of  $o_k$  with respect to  $z_k$  is equal to the derivative of the transfer function. If the transfer function is considered as logistic sigmoid:

$$\frac{\partial o_k}{\partial z_k} = o_k(1 - o_k) \quad (4.10)$$

Moreover, the partial derivative of the cost function with respect to the output generated by the network can be written as:

$$\frac{\partial L}{\partial o_k} = \frac{\partial}{\partial o_k} \frac{1}{2} \sum_k (y_k - o_k)^2 \quad (4.11)$$

In the equation above, the partial derivative of terms other than  $o_k$  in the summation operation is equal to zero. Therefore,

$$\frac{\partial L}{\partial o_k} = \frac{\partial}{\partial o_k} \frac{1}{2} (y_k - o_k)^2 \quad (4.12)$$

If the necessary arrangements are performed,

$$\frac{\partial L}{\partial o_k} = -(y_k - o_k) \quad (4.13)$$

If equations (4.8), (4.10), and (4.13) are combined,

$$\frac{\partial L}{\partial w_{kj}} = -(y_k - o_k)o_k(1 - o_k)a_j \quad (4.14)$$

Let's say  $-(y_k - o_k)o_k(1 - o_k) = \delta_k$ , then,

$$\frac{\partial L}{\partial w_{kj}} = \delta_k a_j \quad (4.15)$$

The change in the corresponding weight then becomes equal to,

$$\Delta w_{kj} = -\eta \delta_k a_j \quad (4.16)$$

where  $\eta$  stands for the learning rate.

The amount of change or the amount of adjustment for the weights of hidden layer neurons can be calculated by using the same approach. However, the main difference is that the output of the hidden layer affects all neurons of the downstream layer. In this case, the downstream layer is nothing but the output layer. If the  $j^{\text{th}}$  neuron of the hidden layer is considered,

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}} \quad (4.17)$$

Please notice that the partial derivative of  $z_j$  with respect to  $w_{ji}$  is nothing but input  $x_i$ . Then,

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial z_j} x_i \quad (4.18)$$

The cost function is affected by the  $z_j$  via  $a_j$  because  $a_j$  is a component of  $z_k$ . Therefore, the partial derivative of the cost function with respect to  $z_j$  can be expanded further by using the chain rule.

If the chain rule is applied,

$$\frac{\partial L}{\partial z_j} = \sum_{\substack{k \in \text{down} \\ \text{stream of } j}} \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} \quad (4.19)$$

It is previously derived that,

$$\frac{\partial L}{\partial z_k} = \delta_k \quad (4.20)$$

Therefore,

$$\frac{\partial L}{\partial z_j} = \sum_{\substack{k \in \text{down} \\ \text{stream of } j}} \delta_k \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} \quad (4.21)$$

It is known that the partial derivative of  $z_k$  with respect to  $a_j$  is nothing but  $w_{kj}$ . Furthermore, the partial derivative of  $a_j$  with respect to  $z_j$  is equal to the derivative of the transfer function. If logistic sigmoid function is used as transfer function, the equation becomes equal to:

$$\frac{\partial L}{\partial z_j} = \sum_{\substack{k \in \text{down} \\ \text{stream of } j}} \delta_k w_{kj} a_j (1 - a_j) \quad (4.22)$$

Consequently,

$$\frac{\partial L}{\partial w_{ji}} = a_j (1 - a_j) \sum_{\substack{k \in \text{down} \\ \text{stream of } j}} \delta_k w_{kj} x_i \quad (4.23)$$

Let's say,

$$a_j (1 - a_j) \sum_{\substack{k \in \text{down} \\ \text{stream of } j}} \delta_k w_{kj} = \delta_j \quad (4.24)$$

Then the change in the cost function with respect to the weight of  $j^{\text{th}}$  neuron is equal to:

$$\frac{\partial L}{\partial w_{ji}} = \delta_j x_i \quad (4.25)$$

The change in the corresponding weight becomes equal to the following:

$$\Delta w_{ji} = -\eta \delta_j x_i \quad (4.26)$$

where  $\eta$  is the learning rate as in the previous derivation.

Even though the convergence of the algorithm can be guaranteed when the step size, i.e. the learning rate, is selected properly, the algorithm converges very slowly which can create problems when quick convergence is needed. One way of providing quick convergence can be using the Gauss-Newton algorithm instead of the gradient descent. However, the Gauss-Newton algorithm does not guarantee convergence in every case. In order to benefit from the convergence guarantee of the gradient descent algorithm and the speed of the Gauss-Newton algorithm, the Levenberg-Marquardt algorithm was developed.

#### 4.5.2. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm is a minimization algorithm used in non-linear least-squared problems and developed by Levenberg (1944) and Marquardt (1963) in different time periods independently. The application of the algorithm to machine learning problems was presented by Hagan and Menhaj (1994).

As mentioned before, the Levenberg-Marquardt algorithm provides both the Gauss-Newton algorithm's speed and the gradient descent's convergence guarantee. As a very brief introduction to the Gauss-Newton algorithm, the change in the elements of a vector of  $n$  number of parameters is equal to:

$$\Delta \bar{x} = -[\nabla^2 f(\bar{x})]^{-1} \nabla f(\bar{x}) \quad (4.27)$$

where,  $\nabla^2 f(\bar{x})$  is the Hessian matrix of the function and  $\nabla f(\bar{x})$  is the gradient. If  $f(\bar{x})$  is described as:

$$f(\bar{x}) = \sum_{i=1}^k e_i^2(\bar{x}) \quad (4.28)$$

Then, gradient and Hessian of the function are approximated as follows:

$$\nabla f(\bar{x}) = J^T(\bar{x})e(\bar{x}) \quad (4.29)$$

$$\nabla^2 f(\bar{x}) = J^T(\bar{x})J(\bar{x}) + S(\bar{x}) \quad (4.30)$$

where,  $J$  is the Jacobian matrix, and  $S(\bar{x})$  is given in the form of following:

$$J(\bar{x}) = \begin{bmatrix} \frac{\partial e_1(\bar{x})}{\partial x_1} & \frac{\partial e_1(\bar{x})}{\partial x_2} & \dots & \frac{\partial e_1(\bar{x})}{\partial x_n} \\ \frac{\partial e_2(\bar{x})}{\partial x_1} & \frac{\partial e_2(\bar{x})}{\partial x_2} & \dots & \frac{\partial e_2(\bar{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_k(\bar{x})}{\partial x_1} & \frac{\partial e_k(\bar{x})}{\partial x_2} & \dots & \frac{\partial e_k(\bar{x})}{\partial x_n} \end{bmatrix} \quad (4.31)$$

$$S(\bar{x}) = \sum_{i=1}^k e_i(\bar{x})\nabla^2 e_i \quad (4.32)$$

It is generally assumed that  $S(\bar{x}) \cong 0$ . Physically, this assumption can be interpreted by the magnitude of the error. If the calculated errors or the Hessian of the errors are very small, then  $S(\bar{x}) \cong 0$  assumption is valid. If all findings are substituted back in equation (4.27),

$$\Delta\bar{x} = -[J^T(\bar{x})J(\bar{x})]^{-1}J^T(\bar{x})e(\bar{x}) \quad (4.33)$$

The Levenberg-Marquardt is a modified version of the Gauss-Newton algorithm having a positive combination coefficient,  $\mu$ , as follows:

$$\Delta\bar{x} = -[J^T(\bar{x})J(\bar{x}) + \mu I]^{-1}J^T(\bar{x})e(\bar{x}) \quad (4.34)$$

Please notice that the Levenberg-Marquardt algorithm is nothing but the Gauss-Newton algorithm when  $\mu$  is equal to zero but providing a positive combination coefficient assures that the matrix  $J^T(\bar{x})J(\bar{x}) + \mu I$  is always invertible which eliminates the problems of the Gauss-Newton algorithm. On the other hand, the combination

coefficient provides adaptability to the algorithm for example for very large  $\mu$  values the algorithm turns to the gradient descent algorithm (Yu and Wilamowski 2018).

The calculation of the Jacobian matrix is one of the critical issues in the implementation of the Levenberg-Marquardt algorithm to artificial neural networks. The elements of the Jacobian matrix can be calculated by using the standard back-propagation algorithm with minor modifications. Even though some changes have to be made in the back-propagation algorithm, the same logic applies due to the consecutive chain differentiation. One of the drawbacks of the Levenberg-Marquardt algorithm is that the size of the Jacobian matrix is dependent on the size of the network, i.e. the number of neurons and hidden layers. The algorithm loses its advantage of speed if the size of the network is large. Moreover, the algorithm can only be applied to multi-layer feed-forward neural networks. Therefore, the application range of the algorithm is limited. However, within its limits, the Levenberg-Marquardt algorithm is stable and efficient.

### 4.5.3. Bayesian Regularization

The ultimate aim of training neural networks in this thesis is to create a network that can approximate any function based on the data at hand. Regularization is applied to neural networks in order to increase the performance of the network. One of the commonly used regularization algorithms is Bayesian Regularization which was presented by Mackay (1992). A further implementation of Bayesian Regularization to the neural networks was performed by Foresee and Hagan (1997) by implementing the regularization with the Levenberg-Marquardt algorithm.

The classical training approach for neural networks is to minimize a cost function or an error relationship described in the following form:

$$E_D = \sum_{i=1}^n (y_i - o_i)^2 \quad (4.35)$$

As can be deduced from the previous formulations  $y_i$  represents the actual/desired output whereas  $o_i$  represents the output obtained by the network. Due to the error, the relationship between the actual output and the network output can be written as  $y_i = o_i + v_i$  where  $v_i$  is independent Gaussian noise.

In Bayesian Regularization, however, the cost function - or sometimes called the objective function - is written in the following form:

$$F = \beta E_D + \alpha E_W \quad (4.36)$$

where  $E_W$  is the regularizing function which is equal to the sum of squares of the weight terms, and  $\alpha$  and  $\beta$  are called objective function parameters. It is needed a regularizing function in the neural networks to provide smoothness to the network weights when minimizing the errors.

The main purpose of Bayesian Regularization is to minimize the function  $F$  by using the appropriate objective function parameters and the Bayesian approach is used in order to obtain those parameters. In this approach, the network weights are considered random variables and they can be updated based on the data at hand as follows:

$$P(w|D, \alpha, \beta, M) = \frac{P(D|w, \beta, M)P(w|\alpha, M)}{P(D|\alpha, \beta, M)} \quad (4.37)$$

where  $D$  represents the available data,  $M$  is the network configuration, and  $w$  is the network weights. Actually, equation (4.37) represents the Bayes' rule in the following form:

$$Posterior = \frac{(Likelihood)(Prior)}{Normalization Factor} \quad (4.38)$$

Therefore, the term  $P(w|\alpha, M)$  represents the prior density function of weights when there is no data. The likelihood term  $P(D|w, \beta, M)$  indicates the probability of the data occurring given the weights. The normalization term  $P(D|\alpha, \beta, M)$  is used to obtain the total probability as 1.

If the distribution of likelihood and prior density function is assumed Gaussian, then the probabilities can be written in the form of:

$$P(D|w, \beta, M) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D) \quad (4.39)$$



$$P(w|\alpha, M) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \quad (4.40)$$

The equations introduced above can also be written as:

$$\frac{1}{Z_D(\beta)} \exp(-\beta E_D) = \frac{1}{Z_D(\beta)} \exp\left(-\frac{1}{\beta} \sum_{i=1}^n e_i^2\right) \quad (4.41)$$

$$\frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) = \frac{1}{Z_W(\alpha)} \exp\left(-\frac{1}{\alpha} \sum_{i=1}^N w_i^2\right) \quad (4.42)$$

For small noise, in order to normalize the distributions,

$$Z_D(\beta) = \left(\frac{\pi}{\beta}\right)^{\frac{n}{2}} \quad (4.43)$$

$$Z_W(\alpha) = \left(\frac{\pi}{\alpha}\right)^{\frac{N}{2}} \quad (4.44)$$

It is known that  $P(D|\alpha, \beta, M)$  is just the normalizing term. Therefore, the posterior density function is directly related to the likelihood and prior density functions. If all these terms are substituted back into equation (4.37),

$$P(w|D, \alpha, \beta, M) = \frac{\left[\frac{1}{Z_D(\beta)} \exp(-\beta E_D)\right] \left[\frac{1}{Z_W(\alpha)} \exp(-\alpha E_W)\right]}{\text{Normalization Factor}} \quad (4.45)$$

Rearranging the equation yields,

$$P(w|D, \alpha, \beta, M) = \frac{1}{Z_D(\beta) Z_W(\alpha)} \exp(-\beta E_D - \alpha E_W) \quad (4.46)$$

Equivalently,

$$P(w|D, \alpha, \beta, M) = \frac{1}{Z_F(\alpha, \beta)} \exp(-F) \quad (4.47)$$

In order to maximize the posterior probability, it is necessary to minimize the objective function and it can be achieved by determining optimum objective function parameters.

The following equation is used to obtain the optimum objective function parameters by applying Bayes' rule again.

$$P(\alpha, \beta | D, M) = \frac{P(D | \alpha, \beta, M) P(\alpha, \beta | M)}{P(D | M)} \quad (4.48)$$

Please notice that in this case, the prior term is equal to  $P(\alpha, \beta | M)$ , the likelihood term becomes  $P(D | \alpha, \beta, M)$  and the normalization factor is  $P(D | M)$ . If the prior term has uniform density for the objective function parameters, like in the previous case, the posterior term can only be maximized by maximizing the likelihood term. Interesting fact that the normalization factor of the equation (4.37) is nothing but the likelihood term of equation (4.48). If equation (4.37) is solved for the normalization factor, one can obtain the following equation:

$$P(D | \alpha, \beta, M) = \frac{P(D | w, \beta, M) P(w | \alpha, M)}{P(w | D, \alpha, \beta, M)} \quad (4.49)$$

If the previous findings are substituted back into the equation,

$$P(D | \alpha, \beta, M) = \frac{\left[ \frac{1}{Z_D(\beta)} \exp(-\beta E_D) \right] \left[ \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \right]}{\frac{1}{Z_F(\alpha, \beta)} \exp(-F)} \quad (4.50)$$

Equivalently,

$$P(D | \alpha, \beta, M) = \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \frac{\exp(-\beta E_D - \alpha E_W)}{\exp(-F)} \quad (4.51)$$

Please notice that  $\exp(-\beta E_D - \alpha E_W) = \exp(-F)$ . Therefore,

$$P(D | \alpha, \beta, M) = \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_W(\alpha)} \quad (4.52)$$

If  $F(w)$  is approximated by using Taylor series expansion in quadratic form,

$$Z_F(\alpha, \beta) \approx (2\pi)^{\frac{N}{2}} \left[ \det \left( (\mathbf{H}^{\text{MP}})^{-1} \right) \right]^{\frac{1}{2}} \exp \left( -F(w^{\text{MP}}) \right) \quad (4.53)$$

where  $\mathbf{H}$  is the Hessian matrix of the objective function and is equal to  $\beta \nabla^2 E_D + \alpha \nabla^2 E_W$ .

If calculated  $Z_F(\alpha, \beta)$  is substituted into equation (4.52) and derivatives of the equation with respect to objective function parameters are evaluated, one can obtain the optimum objective function parameters as given below.

$$\alpha^{\text{MP}} = \frac{\gamma}{2E_W(w^{\text{MP}})} \quad (4.54)$$

$$\beta^{\text{MP}} = \frac{n - \gamma}{2E_D(w^{\text{MP}})} \quad (4.55)$$

where  $\gamma$  is the effective number of parameters and equals to  $N - 2\alpha^{\text{MP}} \text{tr}(\mathbf{H}^{\text{MP}})^{-1}$ ,  $N$  denotes the total number of parameters in the network.

It is necessary to calculate the Hessian matrix as can be seen from the derived equations. The Levenberg-Marquardt algorithm can be used in the calculation of the Hessian matrix. The procedure introduced in Foresee and Hagan (1997) can be tracked in order to perform calculations. The procedure consists of the following steps:

1. Initialize the objective function parameters and the weights. For the objective function parameters  $\alpha$  and  $\beta$  can be taken as 0 and 1, respectively. In the introduced procedure, the Nguyen and Widrow (1990) method was suggested for the initialization of the weights.
2. The minimization of the objective function,  $F$ , can be performed by using the Levenberg-Marquardt algorithm.
3. In order to calculate the effective number of parameters, the Hessian matrix can be approximated by using the Jacobian matrix and can be solved by using the Levenberg-Marquardt algorithm.

4. From the calculated effective number of parameters, new estimates for the objective function parameters can be obtained.
5. Finally, steps 2 through 4 are repeated until convergence for the objective function parameters is achieved.

## CHAPTER 5

### NUMERICAL EXAMPLES

This chapter of the thesis will cover five different numerical examples. Four of them are well-known benchmark problems from literature and the last one is a new problem that is developed from a real-life reinforced concrete structure. A general framework followed in this thesis for reliability analysis of structural engineering problems by using artificial neural networks will also be introduced herein.

#### **5.1. General Framework of Solution Algorithm**

The solution algorithm applied to the numerical examples consists of several sequential steps that start from the generation of the dataset and end with CMCS by using ANN instead of calculating the actual limit state function by finite element analysis. In this subsection, those steps will be clearly explained as possible and it is hoped that this subsection will provide insight to the reader about the presented method.

##### **5.1.1. Generation of the Dataset and Analysis with Traditional Reliability Methods**

The solution algorithm consists of a combination of Python and MATLAB scripts. The structure is modeled and analyzed in Python by using OpenSeesPy. FORM, SORM, CMCS, and IS analyses are performed by using Pystra which is an open software that works with OpenSeesPy in order to perform reliability analysis. Additionally, the necessary dataset for training ANN is generated in Python by using the previously constructed OpenSeesPy model. Then the dataset is written into a spreadsheet file in a tabular fashion.

### **5.1.2. Training Parameters and Algorithm for ANN**

The artificial neural networks used for the reliability analysis are trained by using the Bayesian Regularization algorithm in this study. The main advantage of the Bayesian Regularization algorithm is not its efficiency in computation time but its increased prediction capability with respect to Levenberg-Marquardt algorithm. The algorithm optimizes the Levenberg-Marquardt algorithm to increase the accuracy of predictions and obtain a generalized artificial neural network by minimizing error.

No validation checks are performed for the default Bayesian Regularization algorithm function in MATLAB. However, it has been observed that refusing validation checks leads to overfitting in artificial neural networks due to consideration of the performance of the training set as one of the main stopping criteria. Therefore, a maximum number of failures for validation checks are also defined in this study for training networks. In this case, however, the networks are trained based on the performance of the validation set. Fortunately, it has also been observed that the performance of the validation set and training set have a tendency to behave similarly which means that the starting of overfitting in the training phase affects negatively both the performance of validation and test sets. This common behavior enables the use of the performance of the validation set as one of the main stopping criteria in the training phase.

The dataset in the training phase was partitioned by ratios of 70%, 15%, and 15% for training, validation, and test sets.

### **5.1.3. Determination of ANN Architecture**

The main function in MATLAB imports the tabulated dataset and turns the table format into an array format in order to create a compatible dataset for the Neural Network Toolbox. After that, it sends the dataset to a function called “OptArch” which further sends the dataset to sub-functions such as “OneHiddenLayer”, “TwoHiddenLayers”, or “ThreeHiddenLayers” based on the raised flags. Those flags have been used as indicators that indicate the optimum architecture is found for the specified number of hidden layers by satisfying user-defined criteria. For example, if an optimum architecture - based on the coefficient of regression of the test set - for one hidden layer is not achieved then the script raises “flag=0” and increases the number of hidden layers to two hidden layers

automatically. However, if the criterion is satisfied then it raises “flag=1” and the sub-function stops the search and saves the trained network.

#### **5.1.4. Reloading Trained ANN**

After saving the network, the “OptArch” function imports back the saved network and sends it to the main MATLAB function. However, in MATLAB, the networks are saved as structure-type arrays and therefore they cannot be imported directly. It is necessary to further import the network among the fields of the structure array. Then, they can be directly used for simulations. Alternatively, the trained network can be transferred to the main function directly, however, the above procedure is more useful if an existing network in the folder is to be used.

#### **5.1.5. Reliability Analysis by using ANN-CMCS**

New random variables are generated independently from the previously generated dataset for the training of neural network in order to create a dataset for CMCS. Finally, the main function sends the new dataset and trained network to the CMCS function to perform CMCS. In the CMCS function, the trained network is used as a surrogate instead of the OpenSeesPy model. Those analysis steps can be illustrated as a flow chart given in Figure 5.1.

#### **5.1.6. Repetitive CMCS for the Probability of Failure Estimation**

CMCS for each problem is repeated 10 times due to the small volatility in the results in each run because of the generation of new CMCS samples which are completely different from each other and the dataset used in the training. This process prevents obtaining a single optimistic result for just for one run and leads to a more general probability estimate. The mean value of the probability of failure obtained from all runs is accepted as the final probability of failure estimate. Repetitions increase the CPU time of analyses but it was observed that the increase in time is just a few seconds and therefore the increase is negligible.

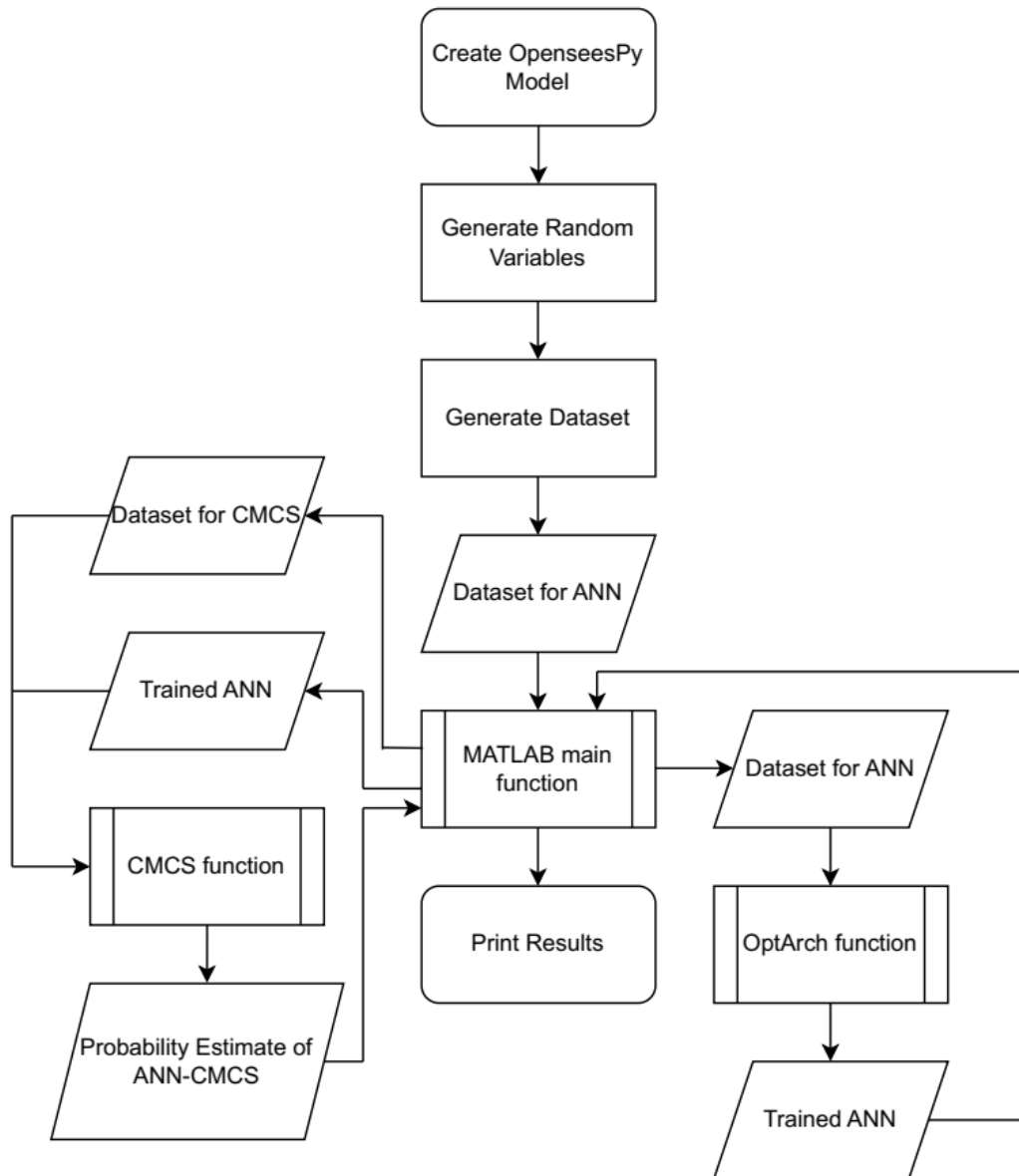


Figure 5.1. Flow chart of the solution algorithm.



## 5.2. Cantilever Beam Example

This example is taken from the study of Rajashekhar and Ellingwood (1993) in order to present the efficiency of artificial neural networks in low-dimensional reliability problems. In the problem, a cantilever beam subjected to a uniformly distributed vertical load given in Figure 5.2 is considered. The beam has a rectangular cross-section having a depth-to-width ratio equal to two. The modulus of elasticity and length of the beam are considered deterministic variables equal to  $2.6 \times 10^4$  MPa and 6 m respectively. The limit state function that represents the maximum serviceability deflection at the free end of the beam is defined explicitly in the equation below:

$$g = -\frac{(wb)l^4}{8EI} + \frac{l}{325} \quad (5.1)$$

If the modulus of elasticity and length of the beam are substituted to the equation above and the width of the beam is written in terms of the depth, the equation simplifies to the following:

$$g(w, h) = 18.46154 - (7.476923 \times 10^{10}) \frac{w}{h^3} \quad (5.2)$$

The statistical properties of the uniformly distributed load and depth of the beam are provided in Table 5.1.

Table 5.1. Statistical properties of the cantilever beam

Random Variable	Mean Value	Coefficient of Variation	Distribution Type
w	1000 N/m <sup>2</sup>	0.20	Normal
h	250 mm <sup>2</sup>	0.15	Normal

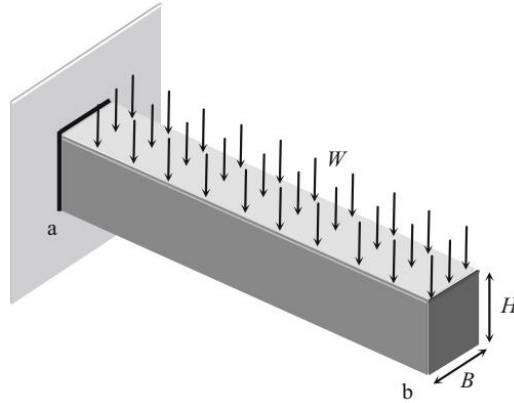


Figure 5.2. Configuration of the cantilever beam (Source: Beheshti Nezhad, Miri, and Ghasemi 2019a).

### 5.2.1. Results

FORM, SORM, CMCS, and IS methods have been applied to the example given above before the application of ANN-CMCS coupling. The results obtained from FORM, SORM, CMCS, and IS are given in the table below:

Table 5.2. Results of FORM, SORM, IS, and CMCS analyses for the cantilever beam example.

Method	Estimated Probability of Failure
FORM	0.00988
SORM (Breitung)	0.00957
SORM (Breitung HR)	0.00952
IS (7000 Samples with 0.02 CoV)	0.00964
CMCS (42000 Samples with 0.05 CoV)	0.00964

The change in coefficient of variation (CoV) and estimated probability of failure with respect to the number of simulations for CMCS are given in Figure 5.3 and Figure 5.4.

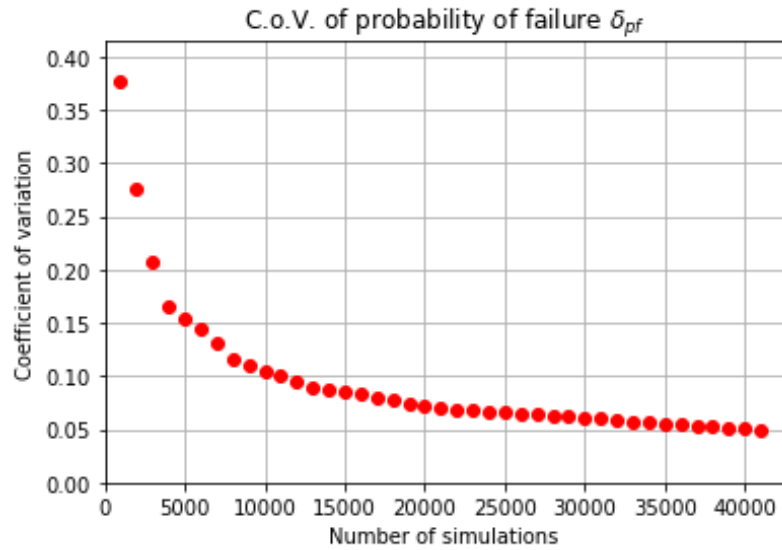


Figure 5.3. Change in the coefficient of variation with respect to the increasing number of simulations for the cantilever beam example.

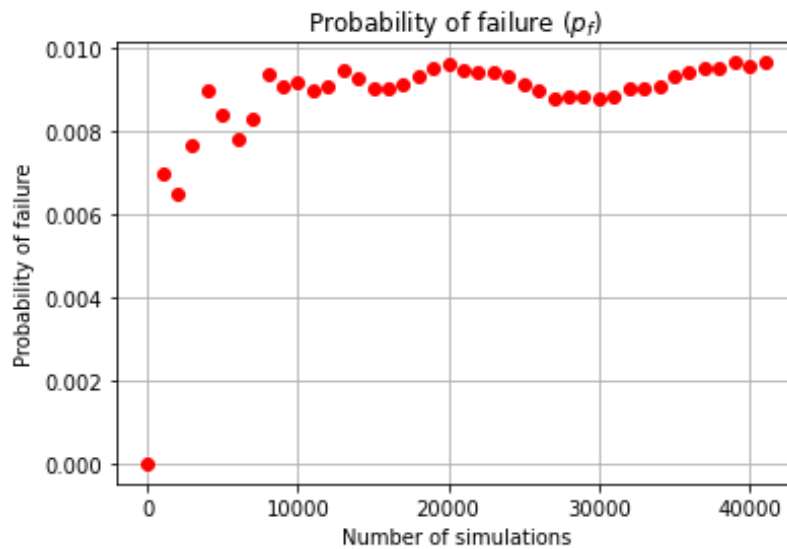


Figure 5.4. Change in the estimated probability of failure with respect to the increasing number of simulations for the cantilever beam example.

Then, neural networks have been trained with different sizes of training samples in order to observe the effect of training samples on the probability estimate. For this purpose, there are 11 different datasets have been generated with a sample size of 30, 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, and 2000. The performance metrics for each trained network are given in the figures below indicating that the networks for each dataset trained well:

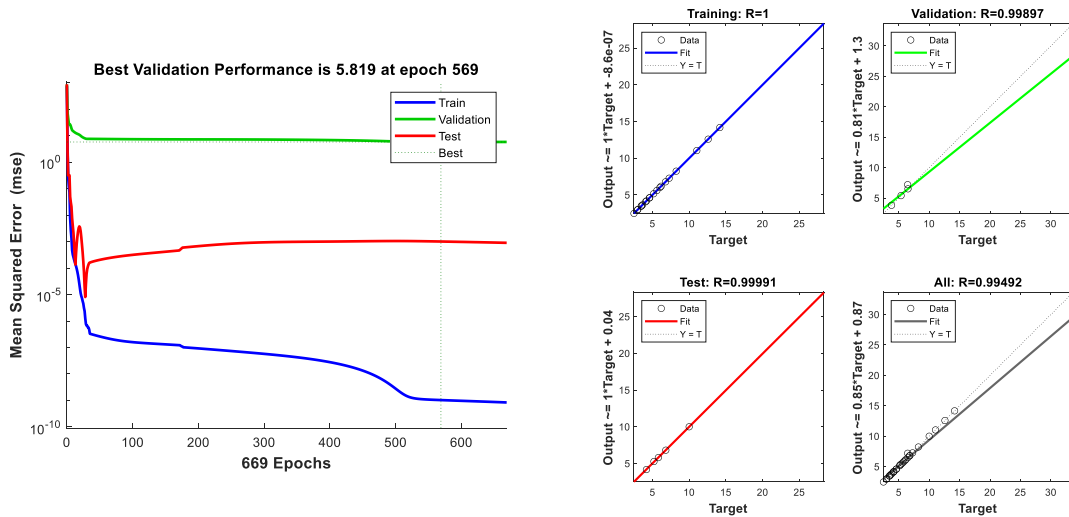


Figure 5.5. Performance metrics of the network that is trained by using 30 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

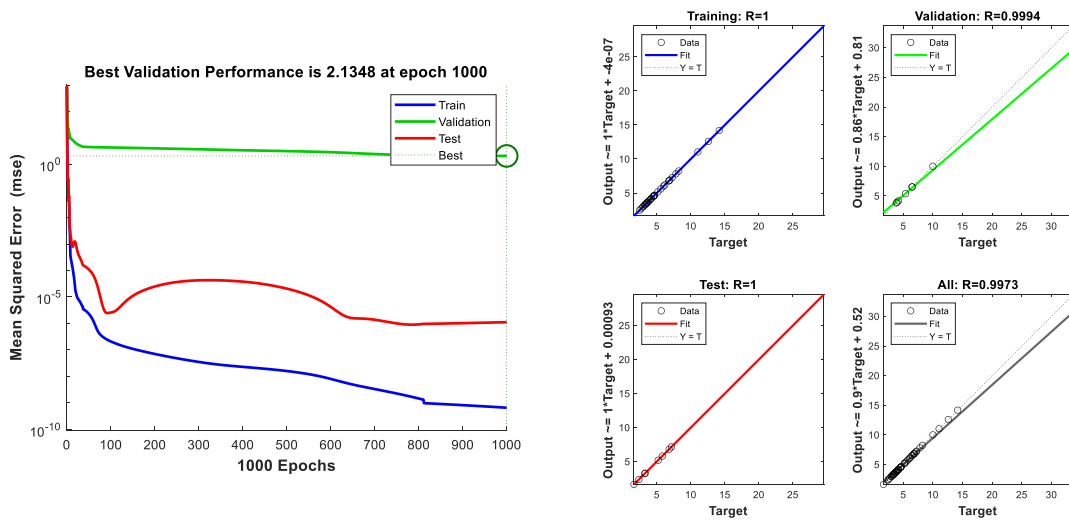


Figure 5.6. Performance metrics of the network that is trained by using 50 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

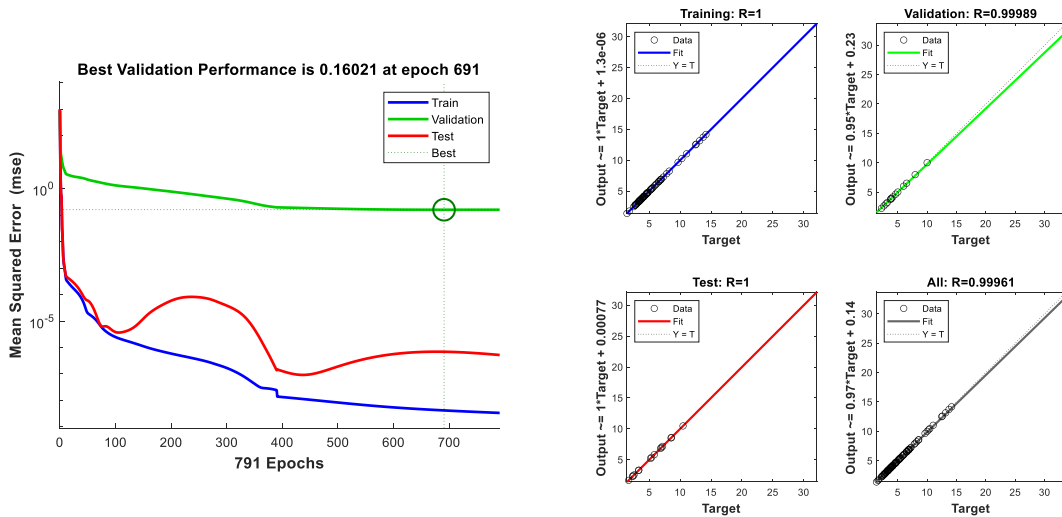


Figure 5.7. Performance metrics of the network that is trained by using 100 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

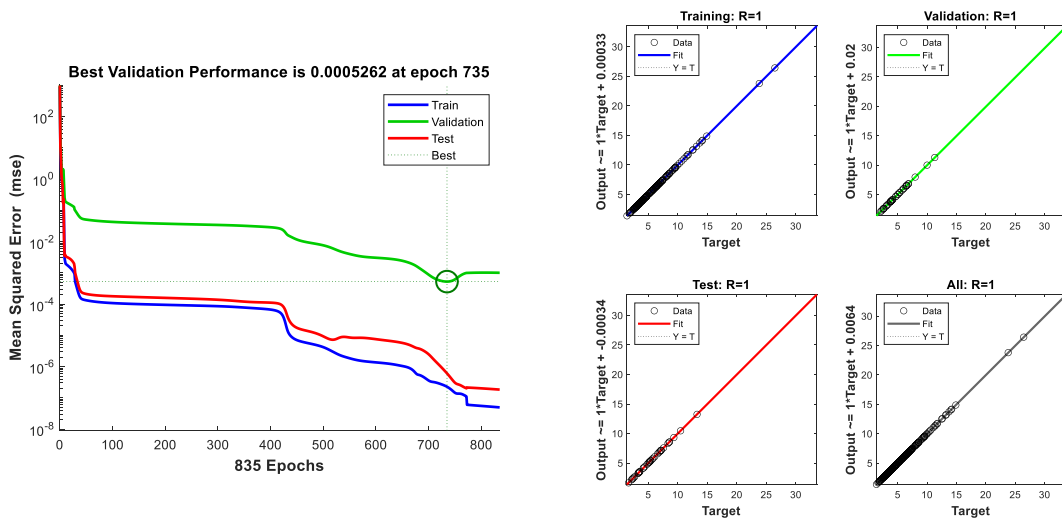


Figure 5.8. Performance metrics of the network that is trained by using 250 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

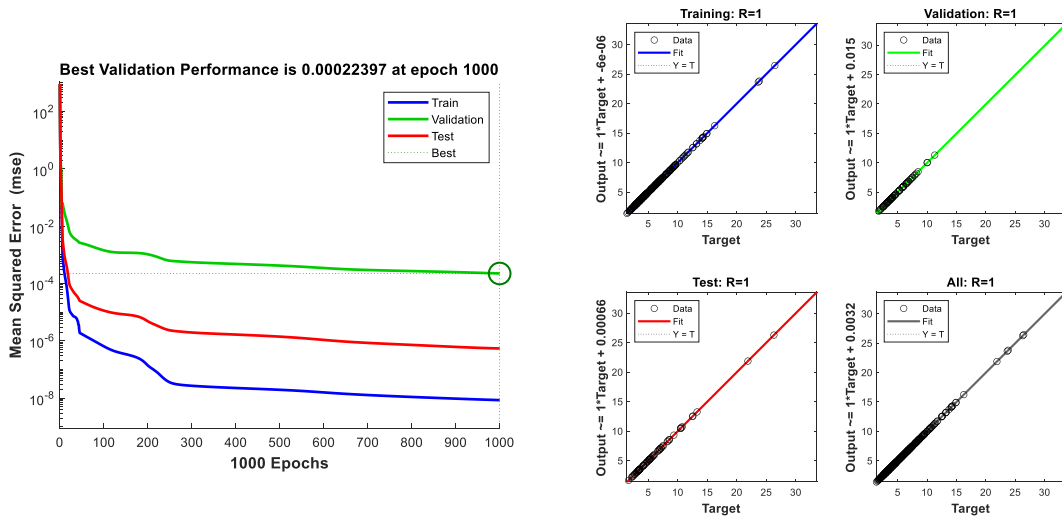


Figure 5.9. Performance metrics of the network that is trained by using 500 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

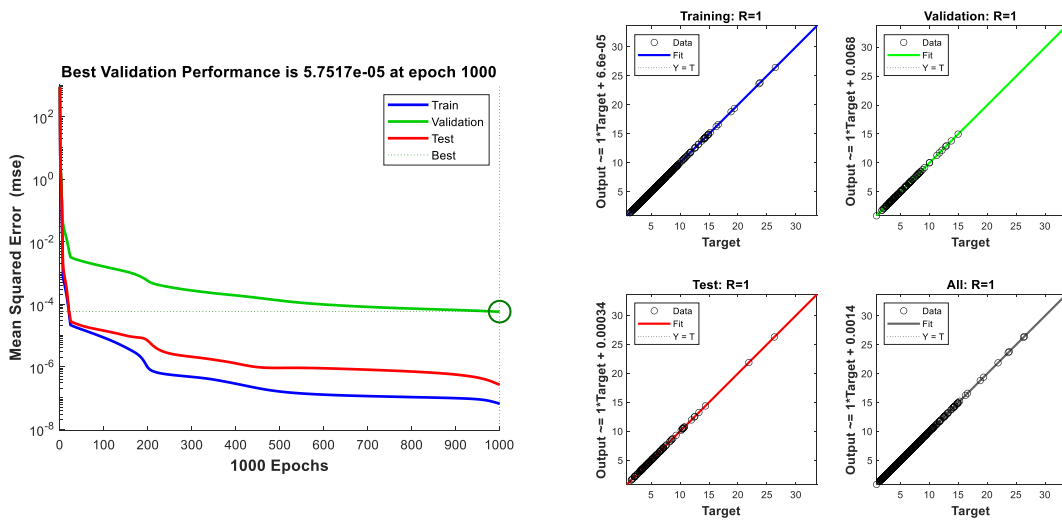


Figure 5.10. Performance metrics of the network that is trained by using 750 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

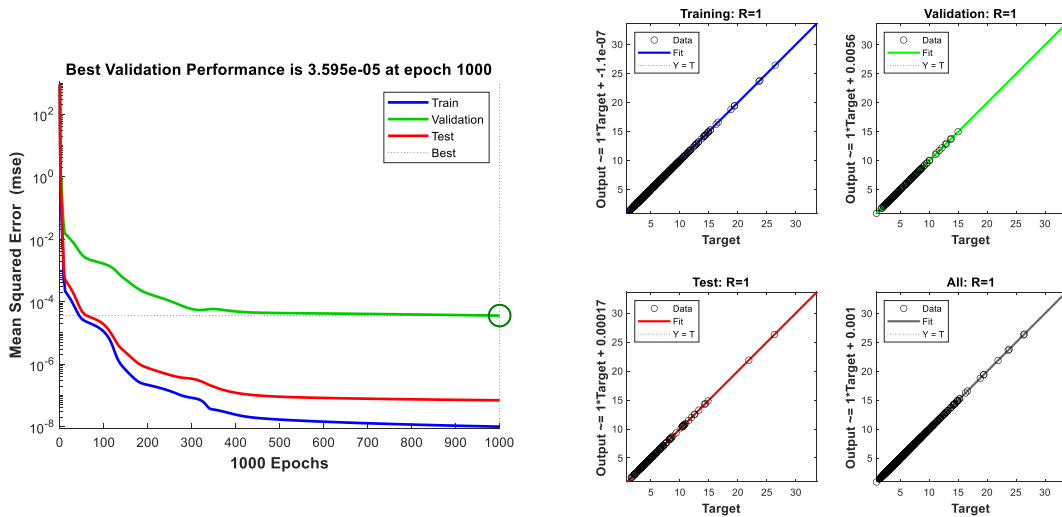


Figure 5.11. Performance metrics of the network that is trained by using 1000 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

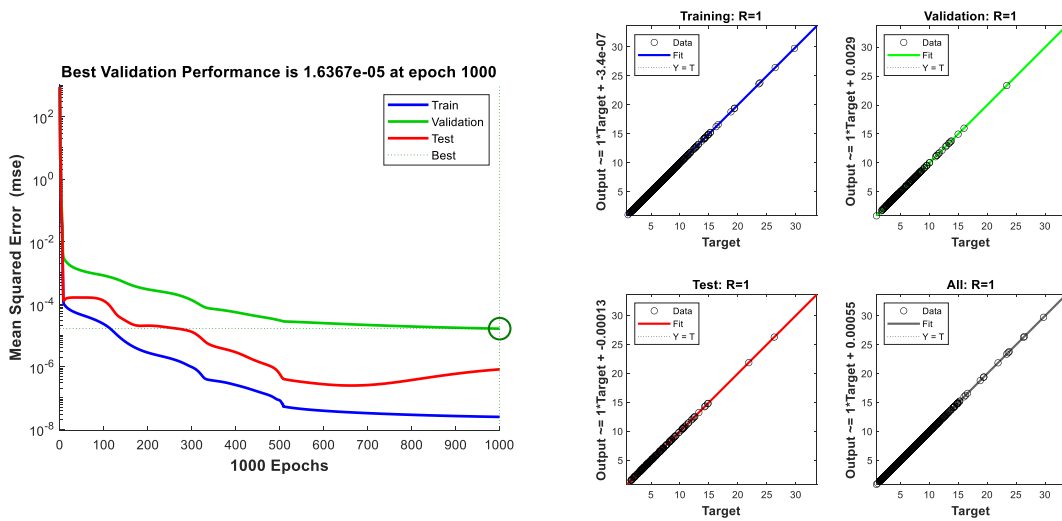


Figure 5.12. Performance metrics of the network that is trained by using 1250 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

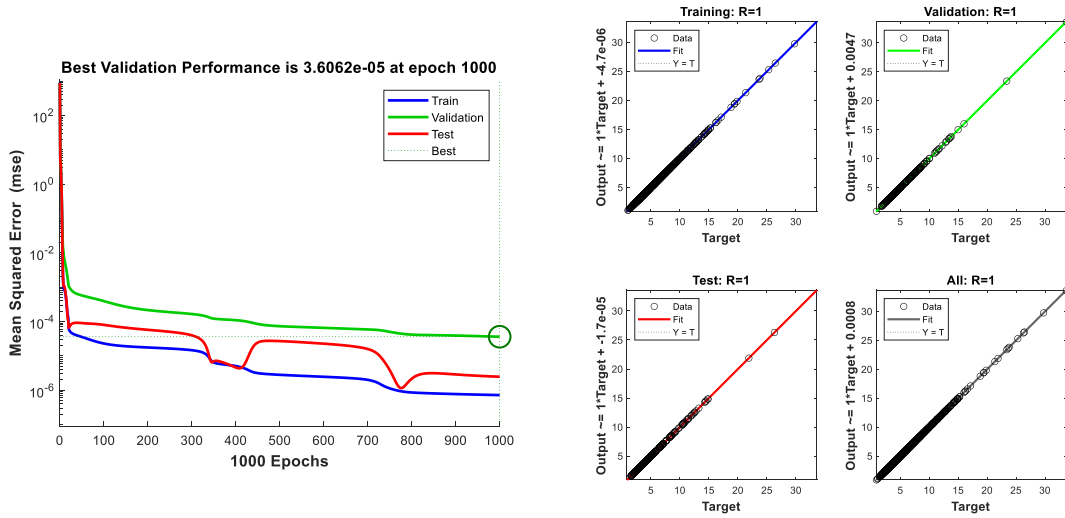


Figure 5.13. Performance metrics of the network that is trained by using 1500 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

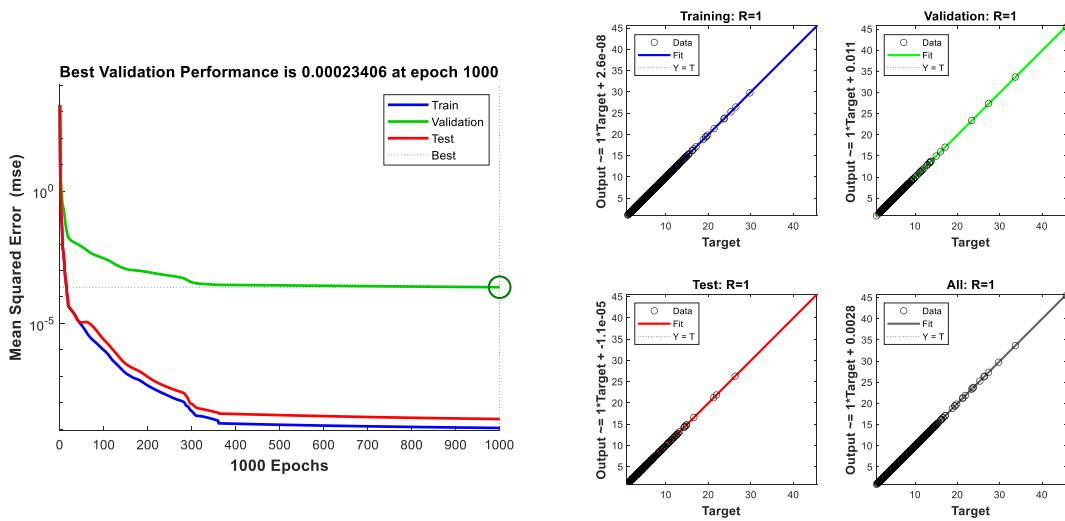


Figure 5.14. Performance metrics of the network that is trained by using 1750 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).



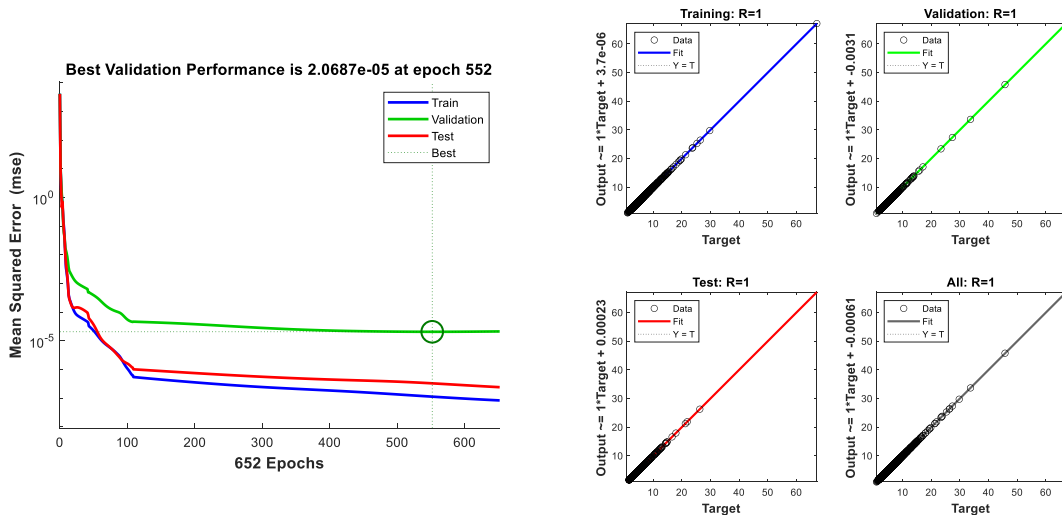


Figure 5.15. Performance metrics of the network that is trained by using 2000 samples for the cantilever beam example. MSE performance (on the left side), regression plots (on the right side).

The probability estimates for the example with respect to the increasing number of training dataset size are given in Figure 5.16. It can be observed that the estimated probability of failure converges to the result obtained by using CMCS when the number of samples in the dataset increases.

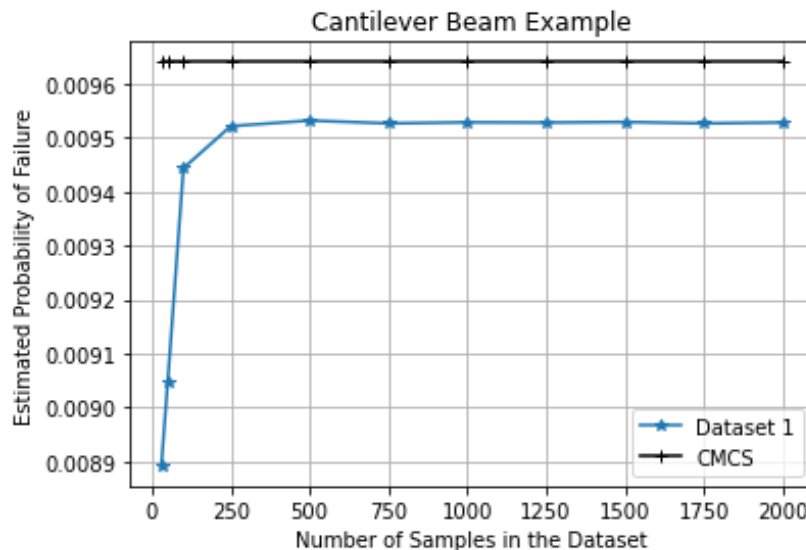


Figure 5.16. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the cantilever beam example.

### 5.3. Simple Portal Frame Example

The simple portal frame structure given in the study of Deng et al. (2005) is revisited in this example. The structural configuration of the frame is given in Figure 5.17 and the limit state function is defined as the exceedance of 0.01 meters of the horizontal displacement in the third node.

In the example, the cross-section area of columns ( $A_1$ ) and beams ( $A_2$ ), and the horizontal wind load ( $P$ ) are considered random variables without correlation whereas the modulus of elasticity ( $E$ ) is a deterministic variable equal to  $2 \times 10^6$  kN/m<sup>2</sup>. Additionally, the relationship between the moment of inertial and the cross-sectional area of the members is defined as follows:

$$I_i = \alpha_i A_i^2 \quad (5.3)$$

The statistical properties of the random variables and coefficients of  $\alpha_i$  are listed in Table 5.3.

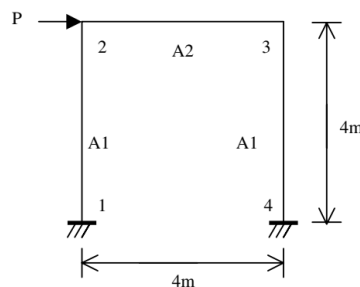


Figure 5.17. Configuration of the portal frame (Source: Deng et al., 2005).

Table 5.3. Statistical properties of the portal frame

Random Variable	Mean Value	Coefficient of Variation	Distribution Type	Coefficient $\alpha_i$
$A_1$	0.36 m <sup>2</sup>	0.10	Lognormal	0.08333
$A_2$	0.18 m <sup>2</sup>	0.10	Lognormal	0.16670
$P$	20.0 kN	0.25	Gumbel	-

### 5.3.1. Results

FORM, SORM, CMCS, and IS methods have been applied to the example given above before the application of ANN-CMCS coupling. The results obtained from FORM, SORM, CMCS, and IS are given in the table below:

Table 5.4. Results of FORM, SORM, IS, and CMCS analyses for the simple portal frame example.

Method	Estimated Probability of Failure
FORM	0.002239
SORM (Breitung)	0.002305
SORM (Breitung HR)	0.002311
IS (8000 Samples with 0.02 CoV)	0.002350
CMCS (1E6 Samples with 0.02 CoV)	0.002343

The change in coefficient of variation (CoV) and estimated probability of failure with respect to the number of simulations for CMCS are given in Figure 5.18 and Figure 5.19.

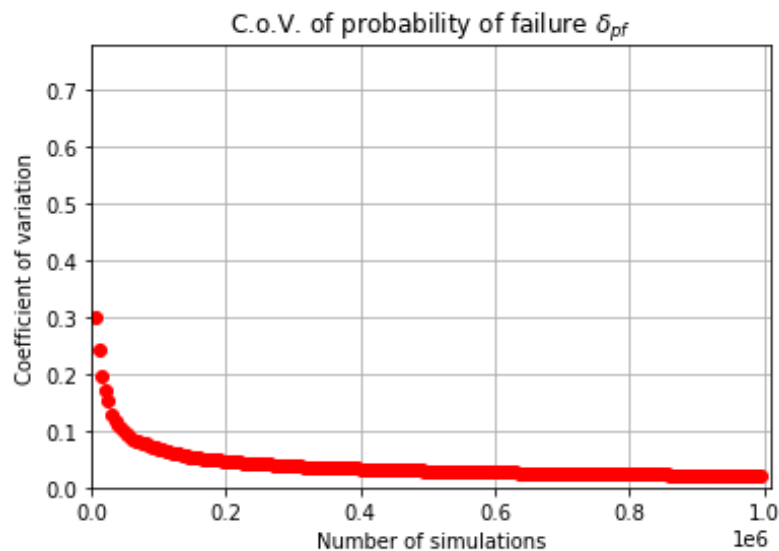


Figure 5.18. Change in the coefficient of variation with respect to the increasing number of simulations for the simple portal frame example.

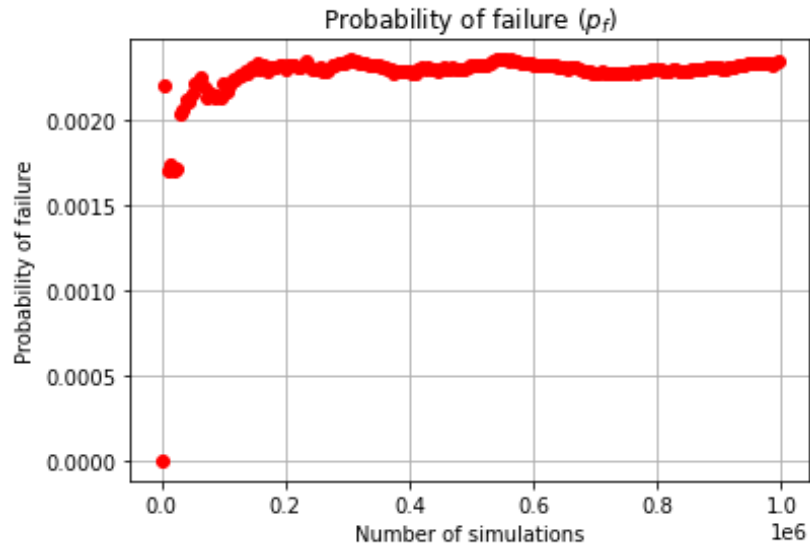


Figure 5.19. Change in the probability estimate with respect to the increasing number of simulations for the simple portal frame example.

The performance metrics for each trained network are given in the figures below indicating that the trained networks for each dataset trained well.

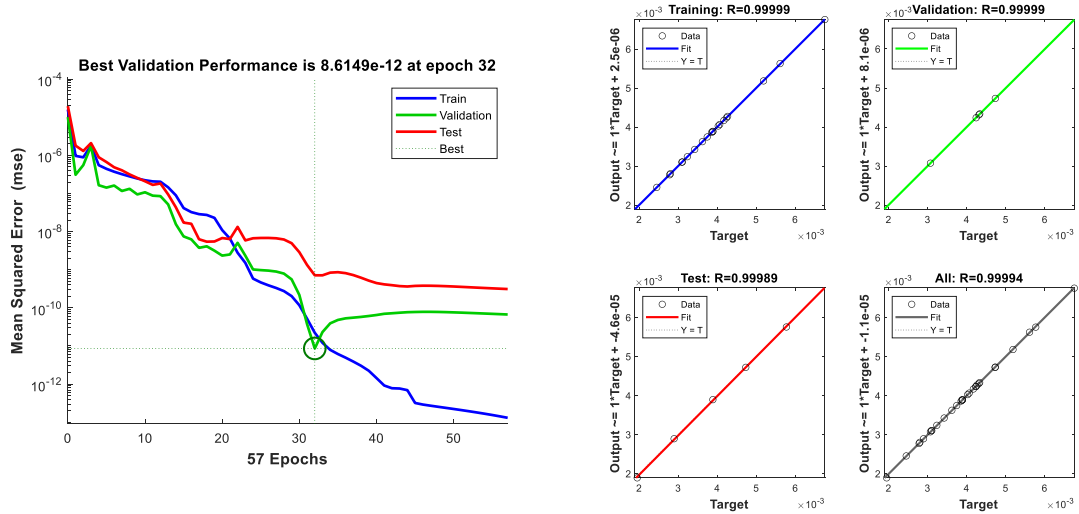


Figure 5.20. Performance metrics of the network that is trained by using 30 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

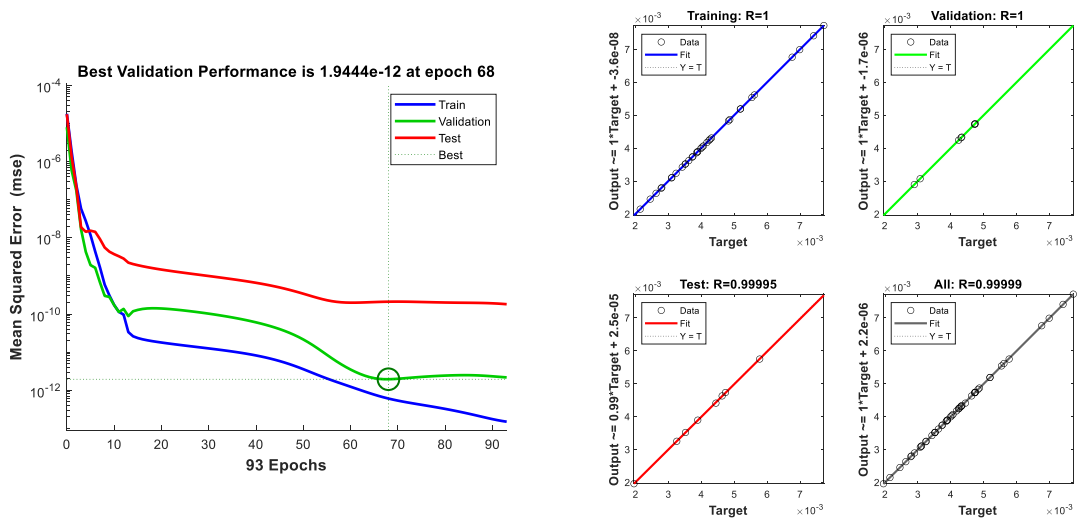


Figure 5.21. Performance metrics of the network that is trained by using 50 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

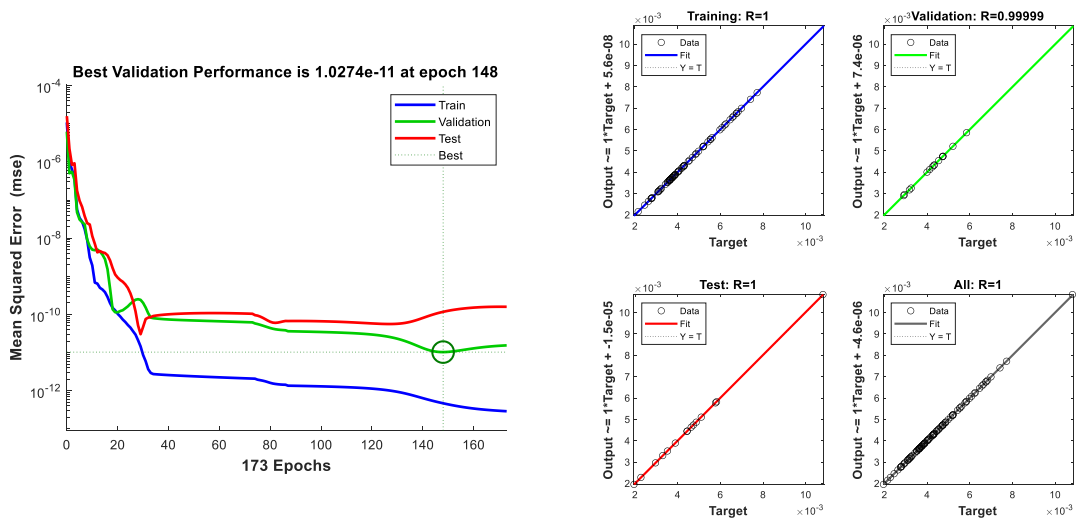


Figure 5.22. Performance metrics of the network that is trained by using 100 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

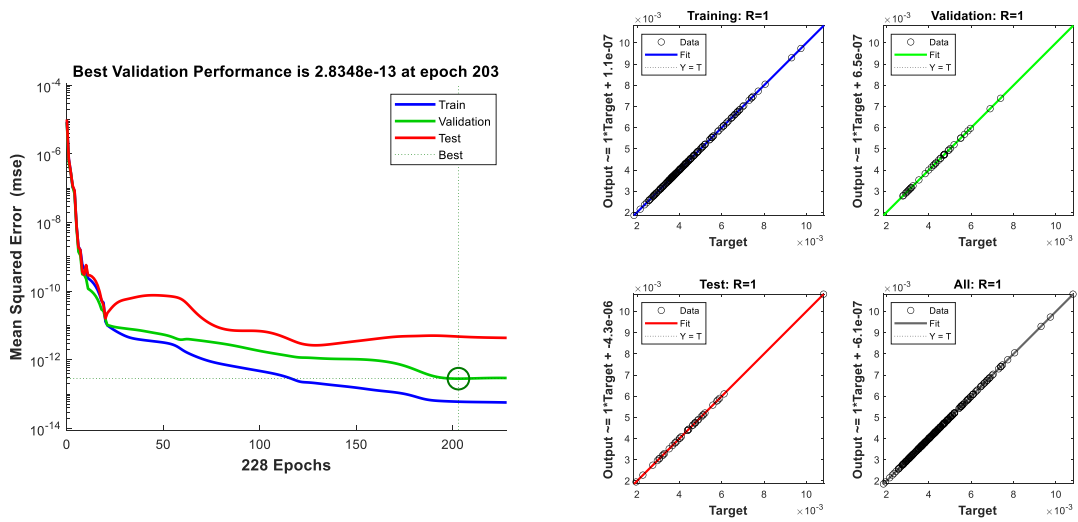


Figure 5.23. Performance metrics of the network that is trained by using 250 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

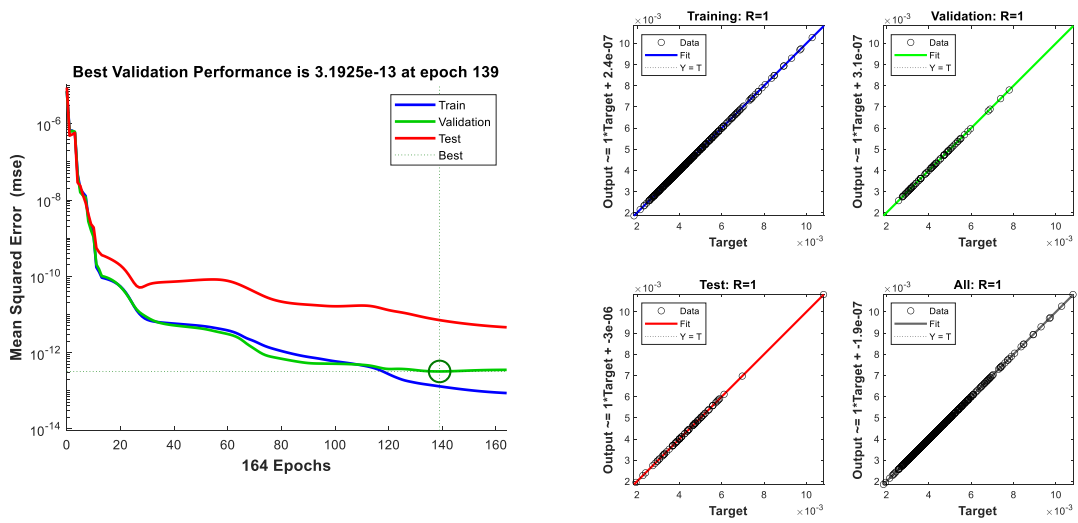


Figure 5.24. Performance metrics of the network that is trained by using 500 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

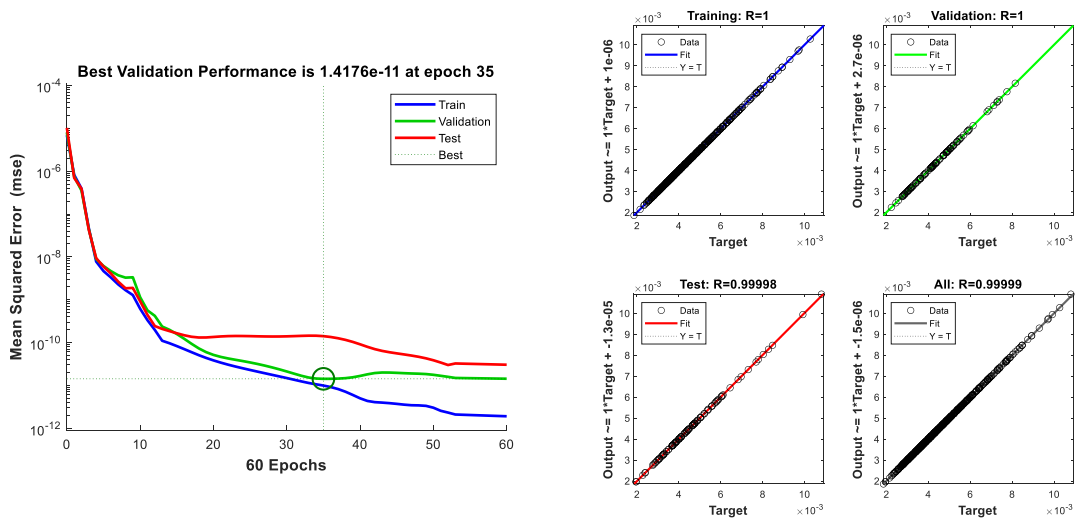


Figure 5.25. Performance metrics of the network that is trained by using 750 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

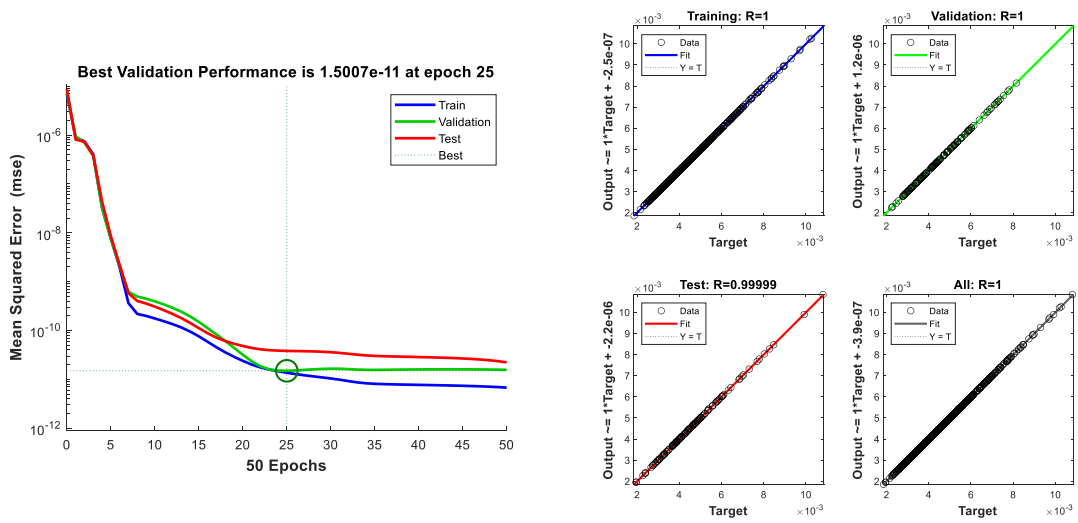


Figure 5.26. Performance metrics of the network that is trained by using 1000 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

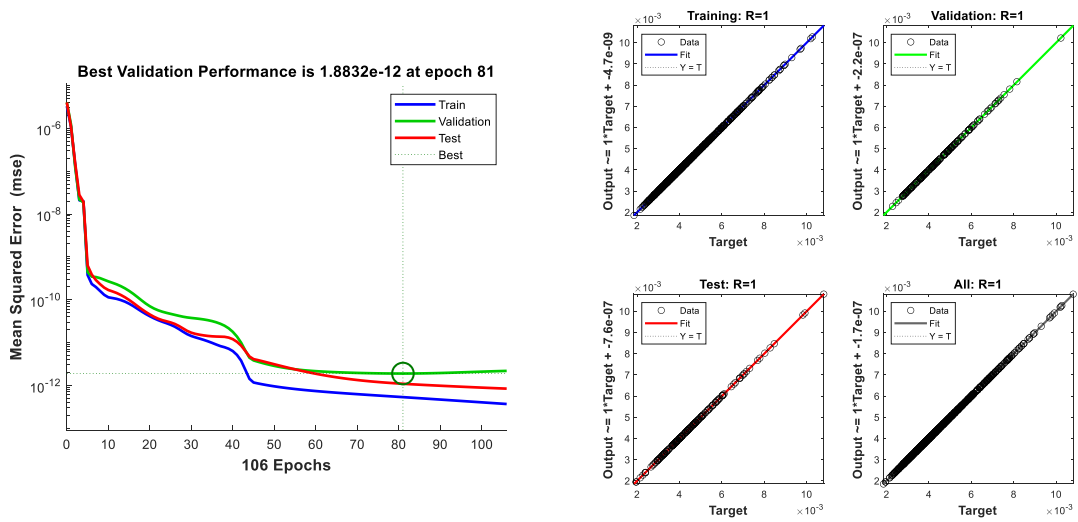


Figure 5.27. Performance metrics of the network that is trained by using 1250 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

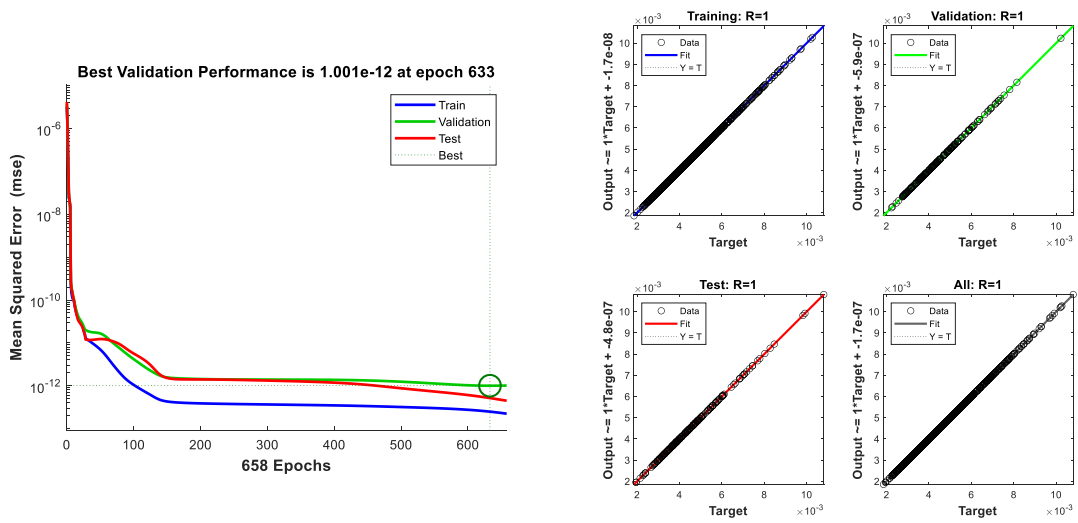


Figure 5.28. Performance metrics of the network that is trained by using 1500 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).



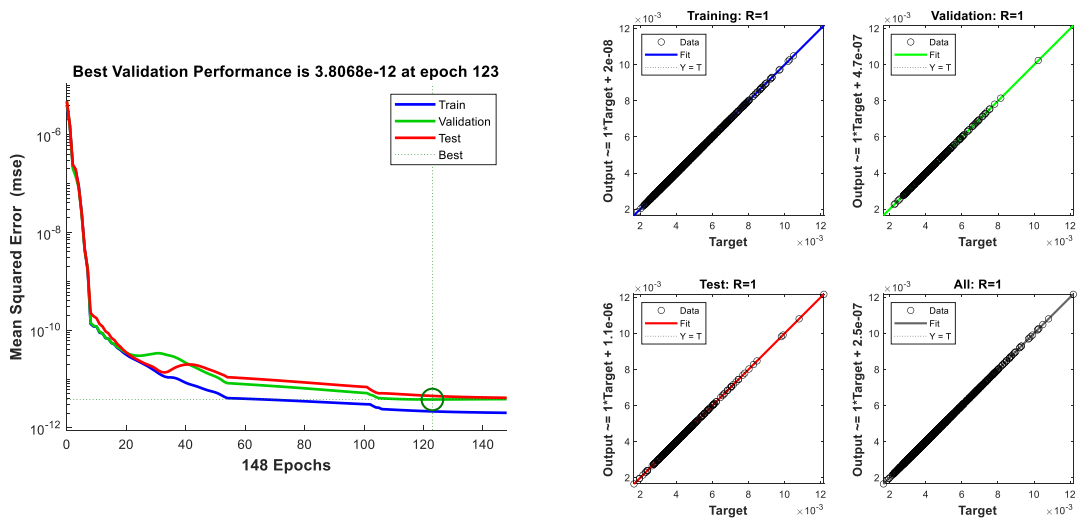


Figure 5.29. Performance metrics of the network that is trained by using 1750 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

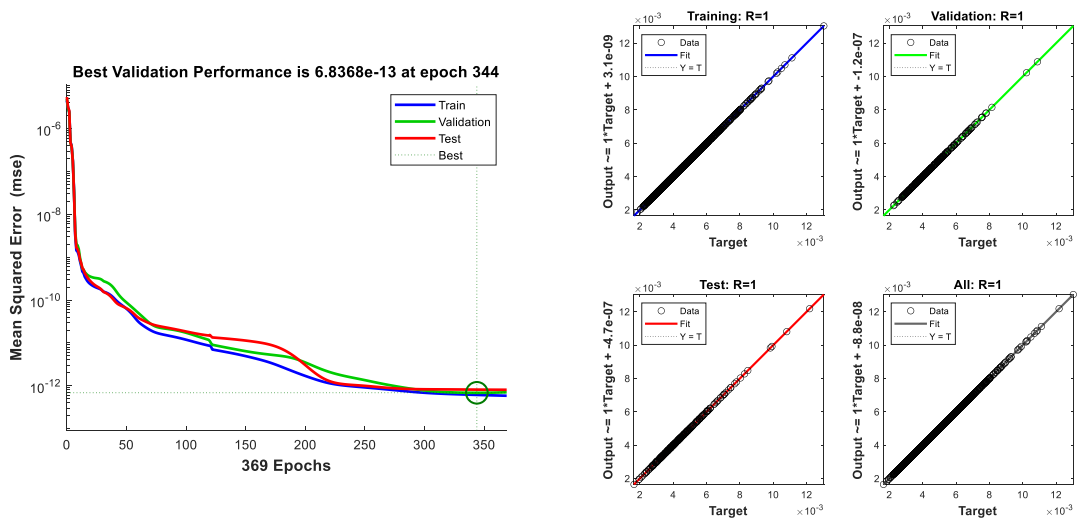


Figure 5.30. Performance metrics of the network that is trained by using 2000 samples for the simple portal frame example. MSE performance (on the left side), regression plots (on the right side).

The probability estimates for the example with respect to the increasing number of training dataset size are given in Figure 5.31. It can be observed that the estimated probability of failure converges to the result obtained by using CMCS with some amount of fluctuations when the number of samples in the dataset increases.

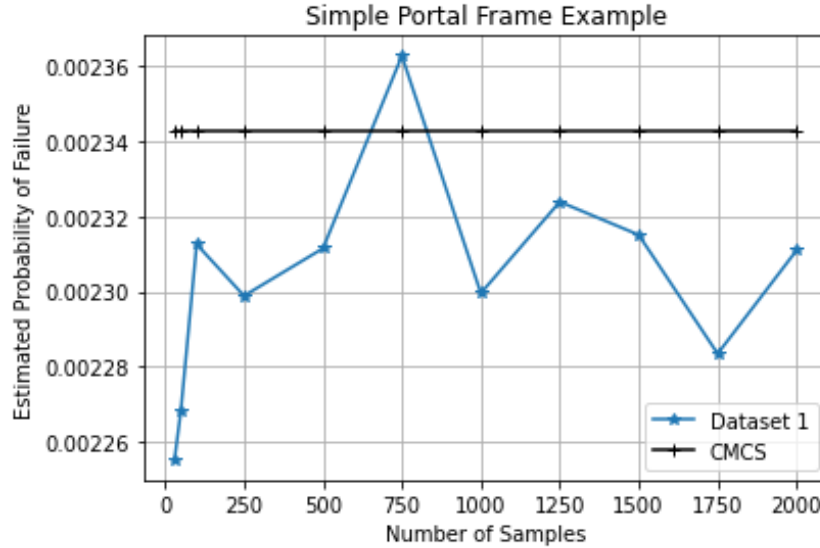


Figure 5.31. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the simple portal frame example.

#### 5.4. 12-Story 3-Bay Frame Structure Example

The 12-story frame structure example given in Figure 5.32 has been covered in previous studies (Cheng and Xiao 2005; Cheng 2007; Beheshti Nezhad, Miri, and Ghasemi 2019a) as a benchmark problem to prove the efficiency of the developed methods. In this study, the frame structure will be analyzed to further investigate the capability of artificial neural networks in the structural reliability analysis field.

The problem consists of six random variables that are the cross-sectional area for five different members ( $A_I$ ) and horizontal acting load ( $P$ ). There is also a relationship exists between the cross-sectional area and the moment of inertia of each member as in the previous example and given in the equation below. The modulus of elasticity is considered a deterministic variable equal to  $2 \times 10^7$  kN/m<sup>2</sup> and there is no correlation defined for the random variables. The limit state function is described as the exceedance a horizontal displacement limit of the top right node,  $u_A$ , as given in the equation below.

$$I_i = \alpha_i A_i^2 \quad (5.4)$$

$$g(A_1, A_2, A_3, A_4, A_5, P) = 0.096 - u_A(A_1, A_2, A_3, A_4, A_5, P) \quad (5.5)$$

The statistical properties of the random variables and coefficients of  $\alpha_i$  are listed in Table 5.5.

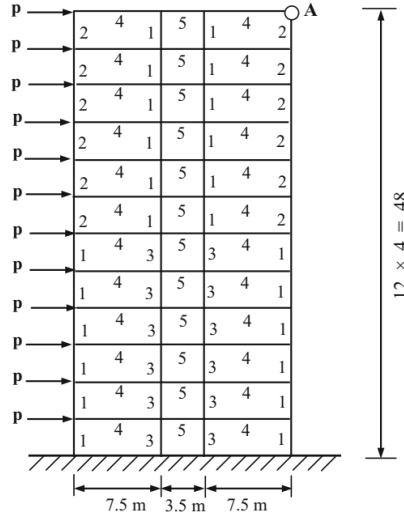


Figure 5.32. Configuration of the 12-story frame (Source: Beheshti Nezhad et al., 2019).

Table 5.5. Statistical properties of the 12-story frame structure

Random Variable	Mean Value	Coefficient of Variation	Distribution Type	Coefficient $\alpha_i$
$A_1$	0.25 m <sup>2</sup>	0.10	Lognormal	0.08333
$A_2$	0.16 m <sup>2</sup>	0.10	Lognormal	0.08333
$A_3$	0.13 m <sup>2</sup>	0.10	Lognormal	0.08333
$A_4$	0.20 m <sup>2</sup>	0.10	Lognormal	0.26670
$A_5$	0.15 m <sup>2</sup>	0.10	Lognormal	0.20000
P	30.0 kN	0.25	Type I Largest	-

### 5.4.1. Results

FORM, SORM, CMCS, and IS methods have been applied to the example given above before the application of ANN-CMCS coupling. The results obtained from FORM, SORM, CMCS, and IS are given in the table below:

Table 5.6. Results of FORM, SORM, IS, and CMCS analyses for the 12-story 3-bay frame structure example.

Method	Estimated Probability of Failure
FORM	0.07292
SORM (Breitung)	0.07583
SORM (Breitung HR)	0.07676
IS (18000 Samples with 0.01 CoV)	0.07631
CMCS (1E5 Samples with 0.01 CoV)	0.07648

The change in coefficient of variation (CoV) and estimated probability of failure with respect to the number of simulations for CMCS are given in Figure 5.33 and Figure 5.34.

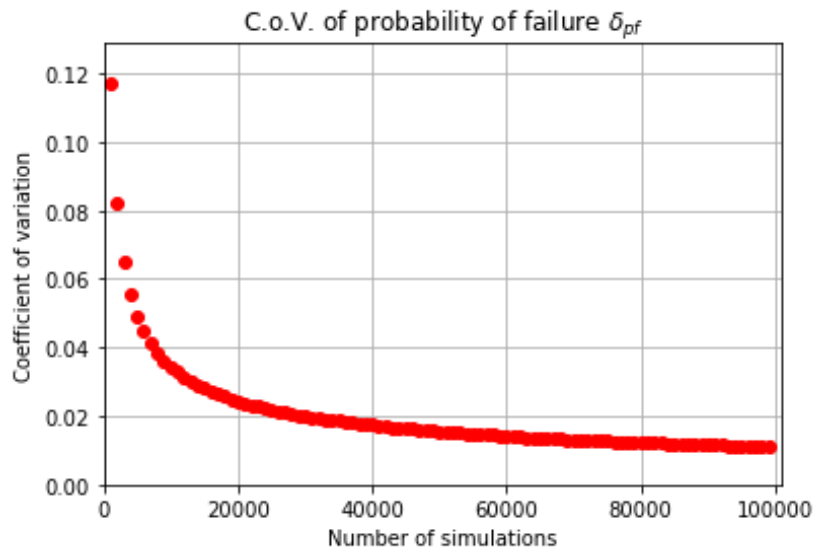


Figure 5.33. Change in the coefficient of variation with respect to the increasing number of simulations for the 12-story 3-bay frame structure example.

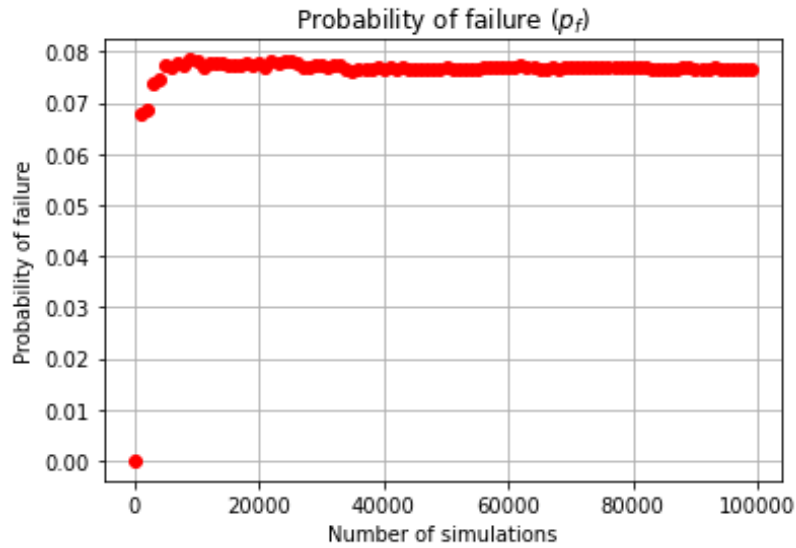


Figure 5.34. Change in the probability estimate with respect to the increasing number of simulations for the 12-story 3-bay frame structure example.

The performance metrics for each trained network are given in the figures below indicating that the trained networks for each dataset trained well.

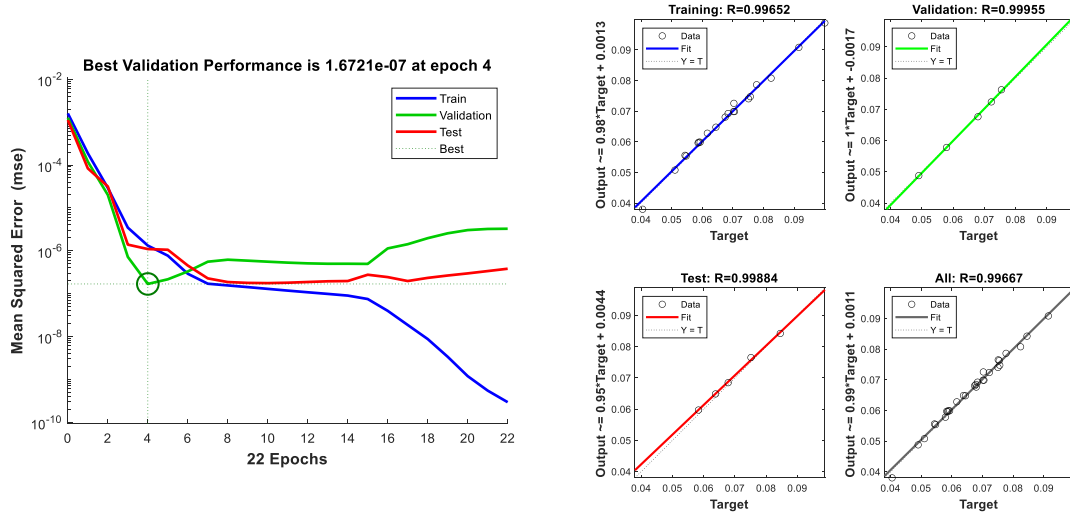


Figure 5.35. Performance metrics of the network that is trained by using 30 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

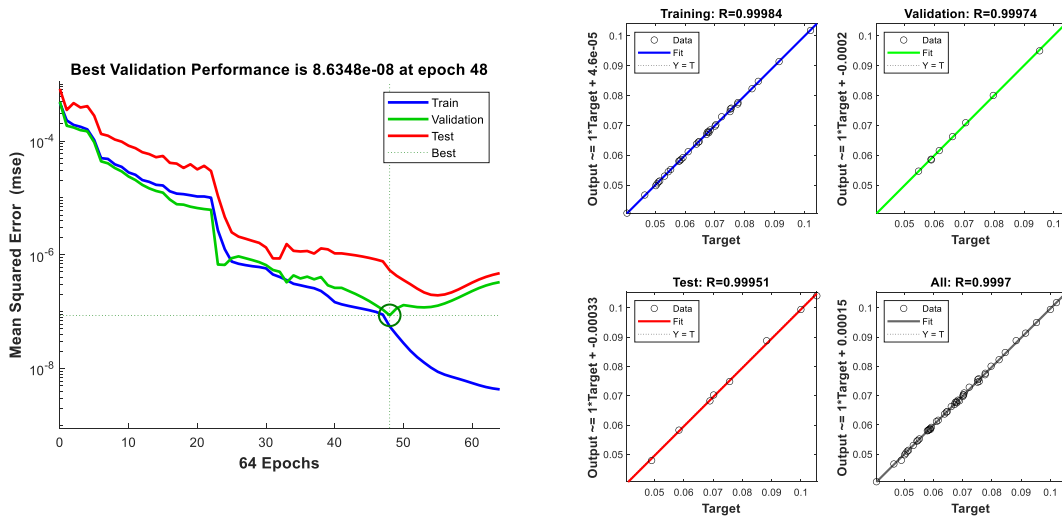


Figure 5.36. Performance metrics of the network that is trained by using 50 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

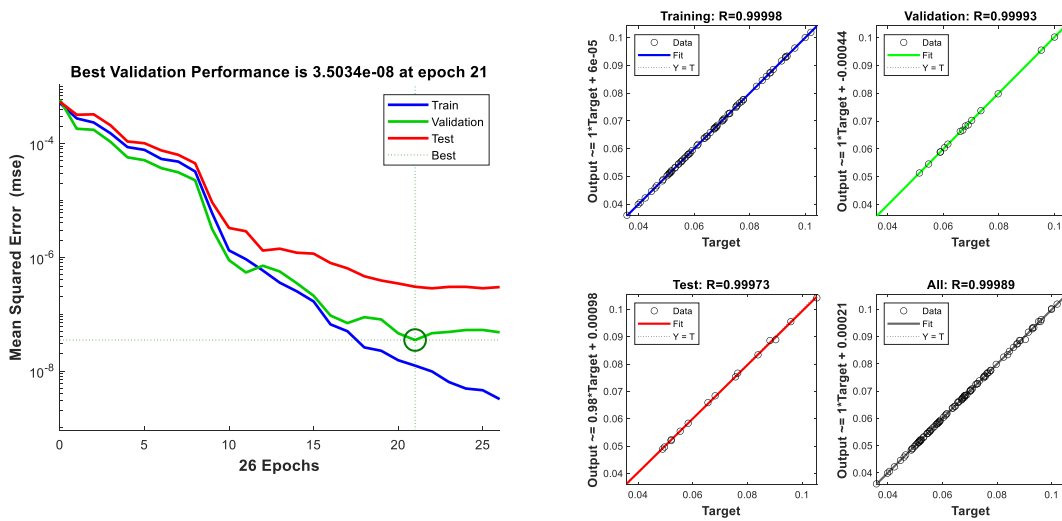


Figure 5.37. Performance metrics of the network that is trained by using 100 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

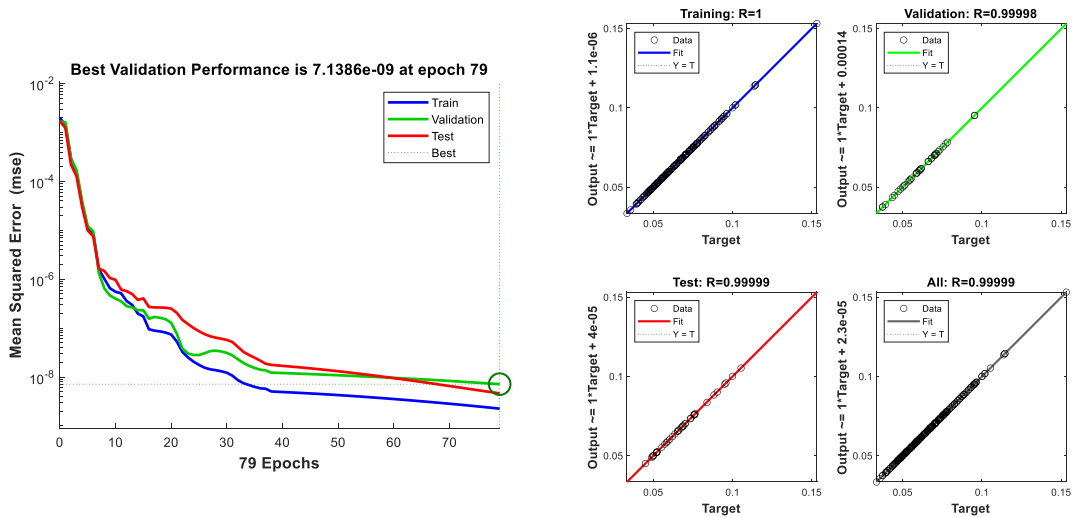


Figure 5.38. Performance metrics of the network that is trained by using 250 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

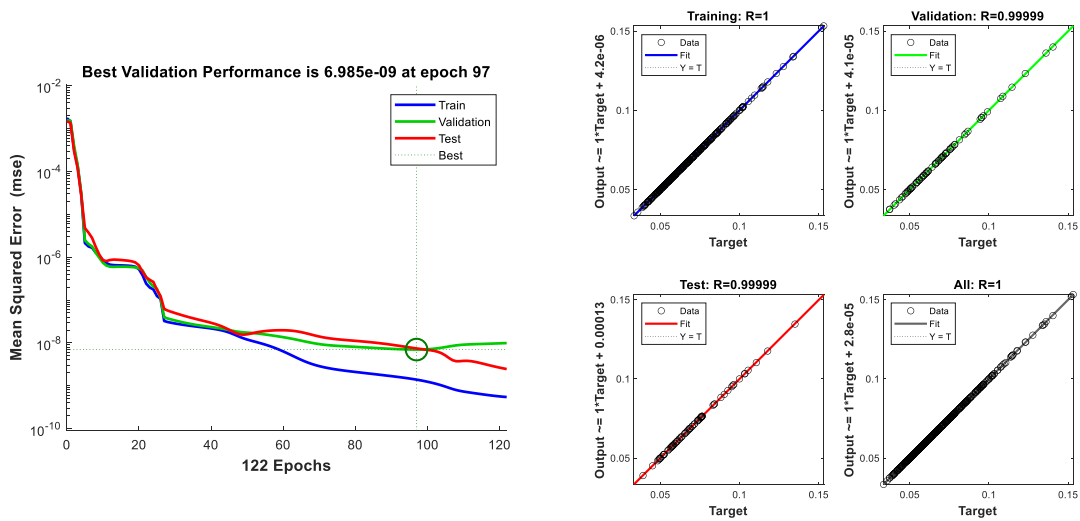


Figure 5.39. Performance metrics of the network that is trained by using 500 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

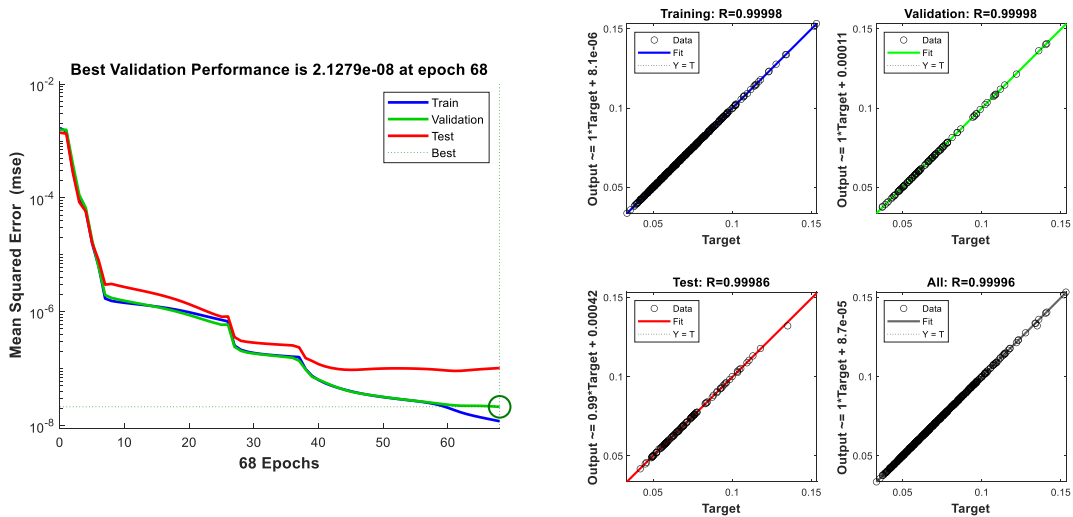


Figure 5.40. Performance metrics of the network that is trained by using 750 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

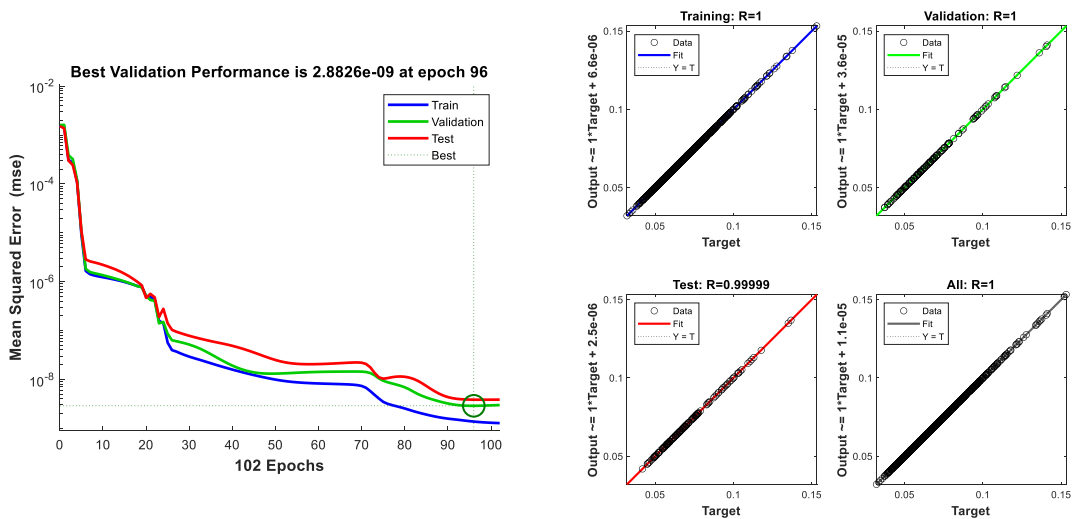


Figure 5.41. Performance metrics of the network that is trained by using 1000 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).



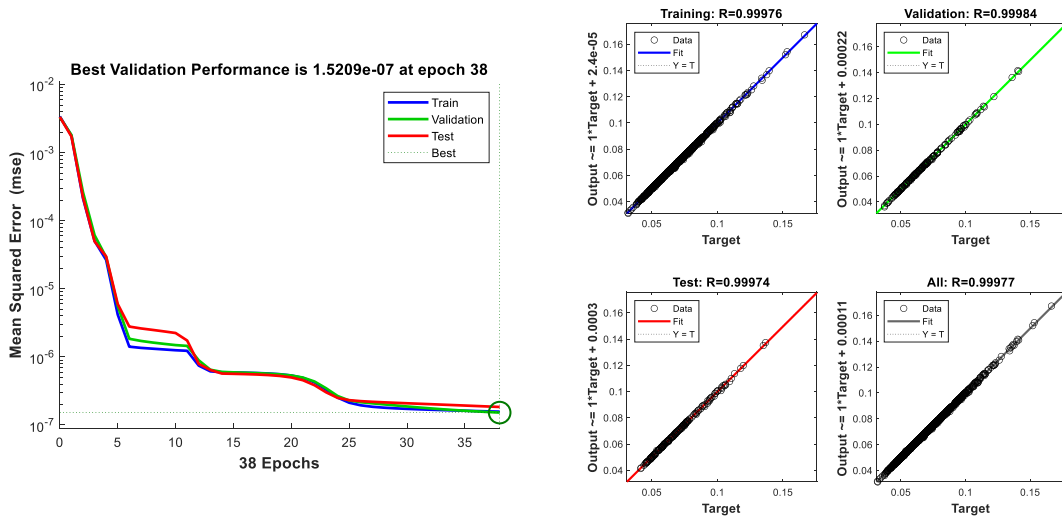


Figure 5.42. Performance metrics of the network that is trained by using 1250 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

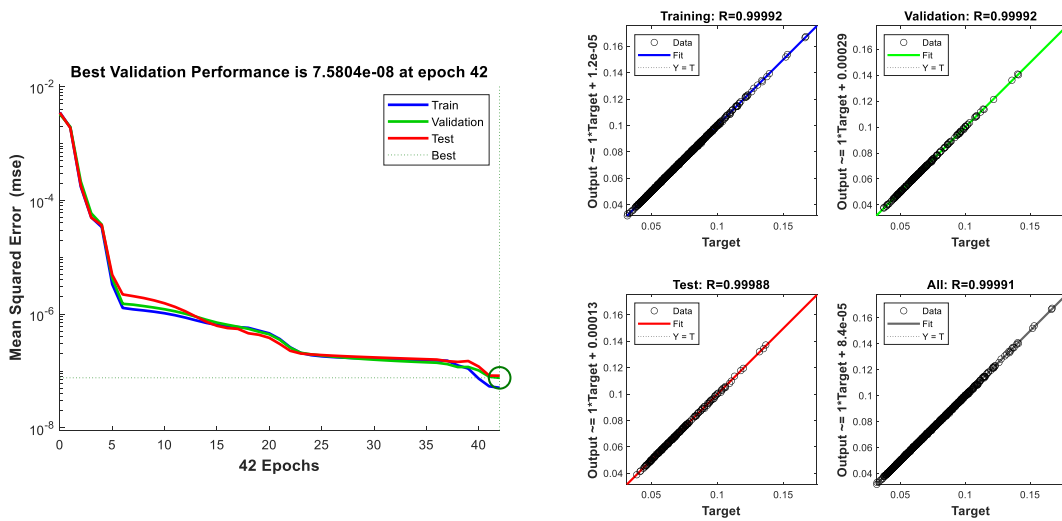


Figure 5.43. Performance metrics of the network that is trained by using 1500 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

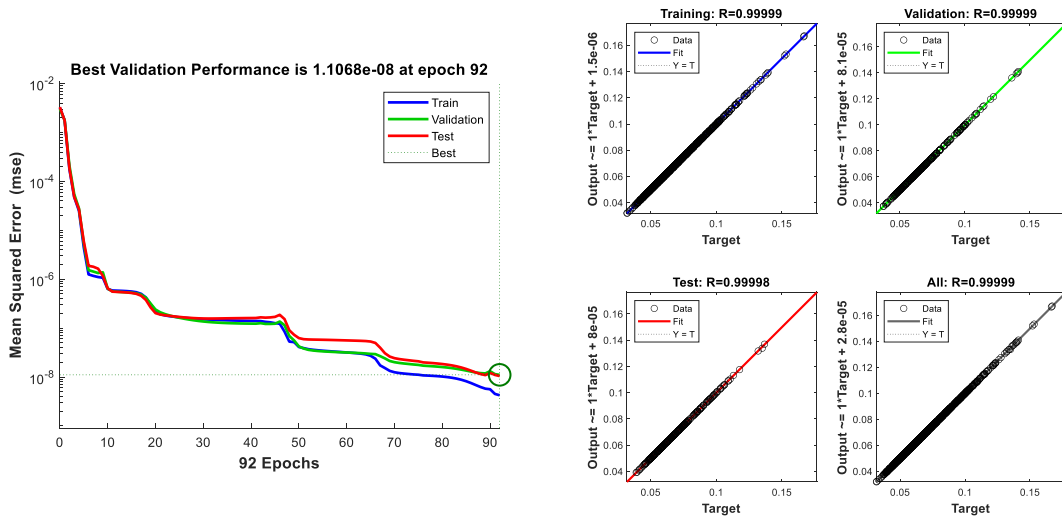


Figure 5.44. Performance metrics of the network that is trained by using 1750 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

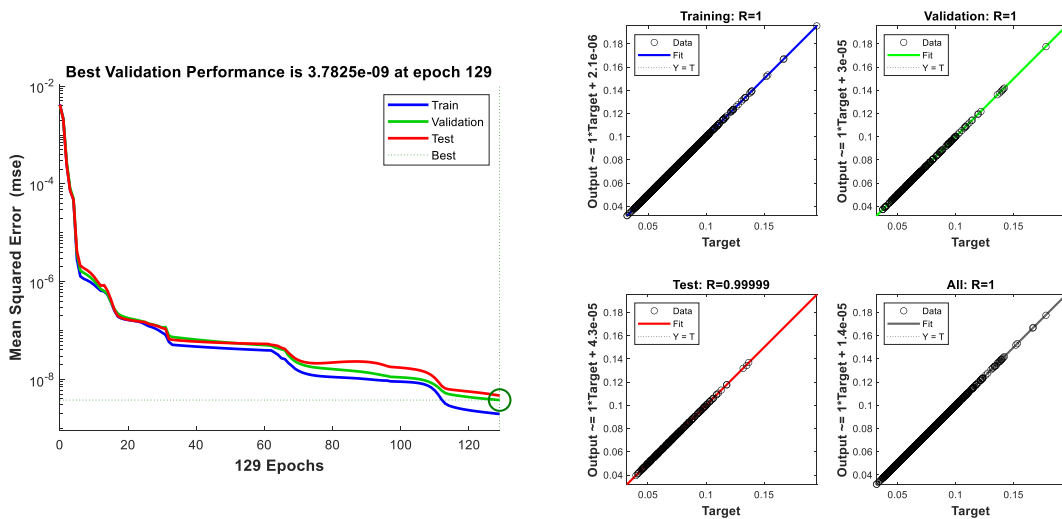


Figure 5.45. Performance metrics of the network that is trained by using 2000 samples for the 12-story 3-bay frame structure example. MSE performance (on the left side), regression plots (on the right side).

The probability estimates for the example with respect to the increasing number of training dataset size are given in Figure 5.46. It can be observed that the estimated probability of failure converges to the result obtained by using CMCS with some amount of fluctuations when the number of samples in the dataset increases.

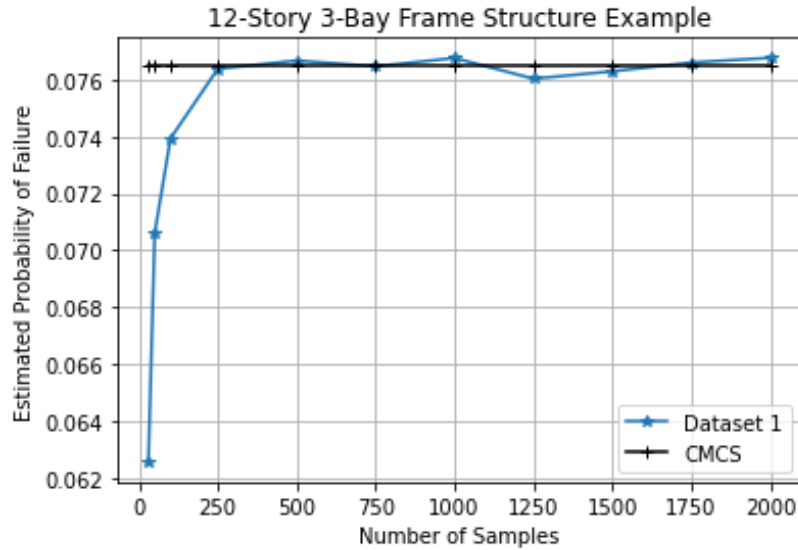


Figure 5.46. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the 12-story 3-bay frame structure example.

### 5.5. 5-Story 3-Bay Correlated Frame Structure Example

In this example, a 5-story 3-bay structural frame system given in Figure 5.47 was reanalyzed. This example has been investigated by several researchers previously (Beheshti Nezhad, Miri, and Ghasemi 2019b; Blatman and Sudret 2010; Bucher and Bourgund 1990a; Liu and Kiureghian 1986; X. S. Nguyen et al. 2009; Richard, Cremona, and Adelaide 2012; Roussouly, Petitjean, and Salaun 2013; Wei and Rahman 2007)

The limit state function is defined as the exceedance of 60 mm horizontal displacement at the top right node. The structural system consists of 8 different members, 2 different elasticity modules, and 3 horizontal-acting loads. The moment of inertia and area of each member is considered random variable along with loads and elasticity modules. Therefore, a total of 21 correlated random variables exist in this problem. The distribution of random variables related to the geometry of members and material properties is deliberately selected as truncated normal distribution in order to prevent the generation of non-physical quantities in the simulation. The statistical properties of the random variables are given in Table 5.7. The coefficients of correlation between the random variables used in this study are given below:

- $\rho_{A_i A_j} = \rho_{I_i I_j} = \rho_{A_i I_j} = 0.13$
- $\rho_{F_i j} = 0.95$
- $\rho_{E_i j} = 0.90$
- $\rho_{A_i I_i} = 0.95$

Table 5.7. Statistical properties of the correlated frame structure

Random Variable	Mean Value	Standard Deviation	Unit	Distribution Type
F <sub>1</sub>	133.454	40.04	kN	Lognormal
F <sub>2</sub>	88.970	35.59	kN	Lognormal
F <sub>3</sub>	71.175	28.47	kN	Lognormal
E <sub>1</sub>	2.173752x10 <sup>7</sup>	1.915200x10 <sup>6</sup>	kN/m <sup>2</sup>	Truncated Normal [0,+∞)
E <sub>2</sub>	2.379736x10 <sup>7</sup>	1.915200x10 <sup>6</sup>	kN/m <sup>2</sup>	Truncated Normal [0,+∞)
I <sub>1</sub>	0.813443x10 <sup>-2</sup>	1.083440x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>2</sub>	1.150936x10 <sup>-2</sup>	1.298048x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>3</sub>	2.137452x10 <sup>-2</sup>	2.506090x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>4</sub>	2.596095x10 <sup>-2</sup>	3.028778x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>5</sub>	1.081076x10 <sup>-2</sup>	2.596095x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>6</sub>	1.415540x10 <sup>-2</sup>	3.461460x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>7</sub>	2.327853x10 <sup>-2</sup>	5.624873x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
I <sub>8</sub>	2.596065x10 <sup>-2</sup>	6.490238x10 <sup>-3</sup>	m <sup>4</sup>	Truncated Normal [0,+∞)
A <sub>1</sub>	3.125640x10 <sup>-1</sup>	5.581500x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>2</sub>	3.721000x10 <sup>-1</sup>	7.442000x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>3</sub>	5.060600x10 <sup>-1</sup>	9.302500x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>4</sub>	5.581500x10 <sup>-1</sup>	11.163000x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>5</sub>	2.530280x10 <sup>-1</sup>	9.302500x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>6</sub>	2.911683x10 <sup>-1</sup>	10.232275x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>7</sub>	3.730300x10 <sup>-1</sup>	12.093250x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)
A <sub>8</sub>	4.186000x10 <sup>-1</sup>	19.537500x10 <sup>-2</sup>	m <sup>2</sup>	Truncated Normal [0,+∞)

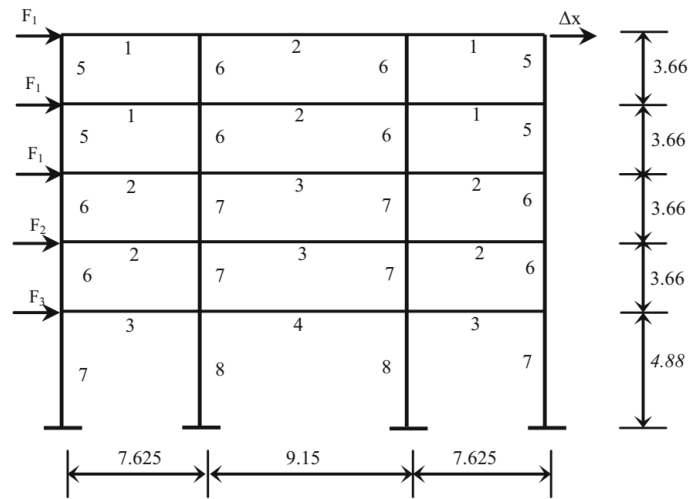


Figure 5.47. The configuration of correlated frame structure (Source: Beheshti Nezhad, Miri, and Ghasemi 2019a).

### 5.5.1. Results

FORM, SORM, CMCS, and IS methods have been applied to the example given above before the application of ANN-CMCS coupling. The results obtained from FORM, SORM, CMCS, and IS are given in the table below:

Table 5.8. Results of FORM, SORM, IS, and CMCS analyses for the 5-story 3-bay correlated frame structure example.

Method	Estimated Probability of Failure
FORM	0.000170
SORM (Breitung)	0.000245
SORM (Breitung HR)	0.000252
IS (46000 Samples with 0.01 CoV)	0.000245
CMCS (2E6 Samples with 0.05 CoV)	0.0002525

The change in coefficient of variation (CoV) and estimated probability of failure with respect to the number of simulations for CMCS are given in Figure 5.48 and Figure 5.49.

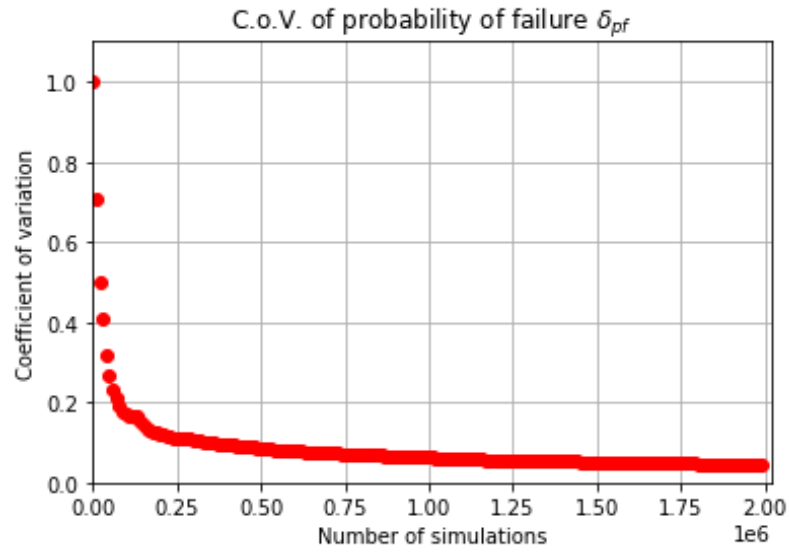


Figure 5.48. Change in the coefficient of variation with respect to the increasing number of simulations for the 5-story 3-bay correlated frame structure example.

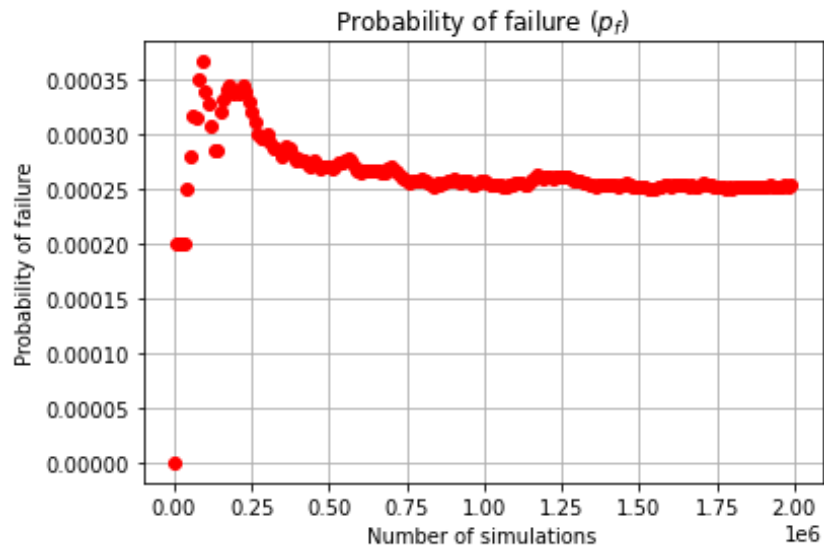


Figure 5.49. Change in the probability estimate with respect to the increasing number of simulations for the 5-story 3-bay correlated frame structure example.

The performance metrics for each trained network are given in the figures below indicating that the trained networks for each dataset trained well.

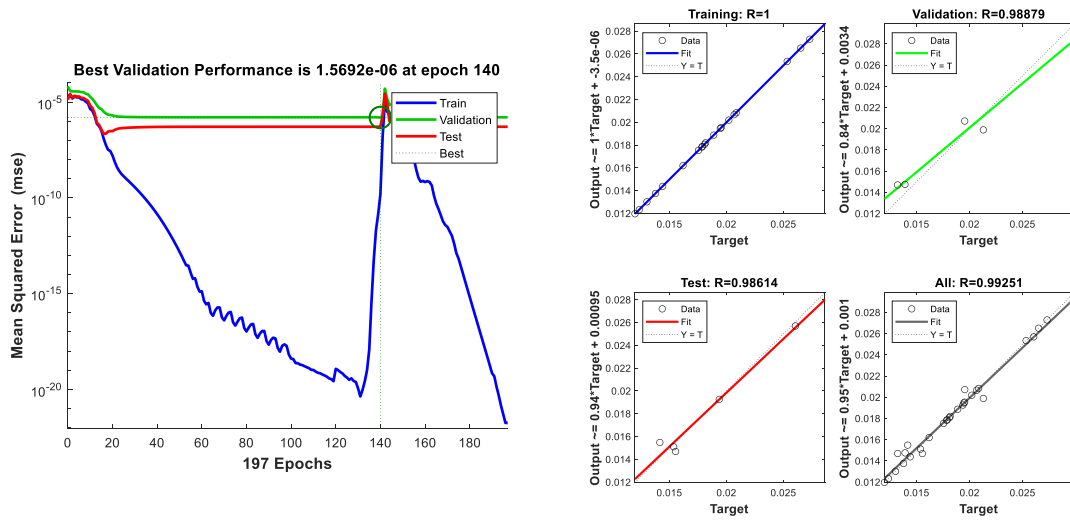


Figure 5.50. Performance metrics of the network that is trained by using 30 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

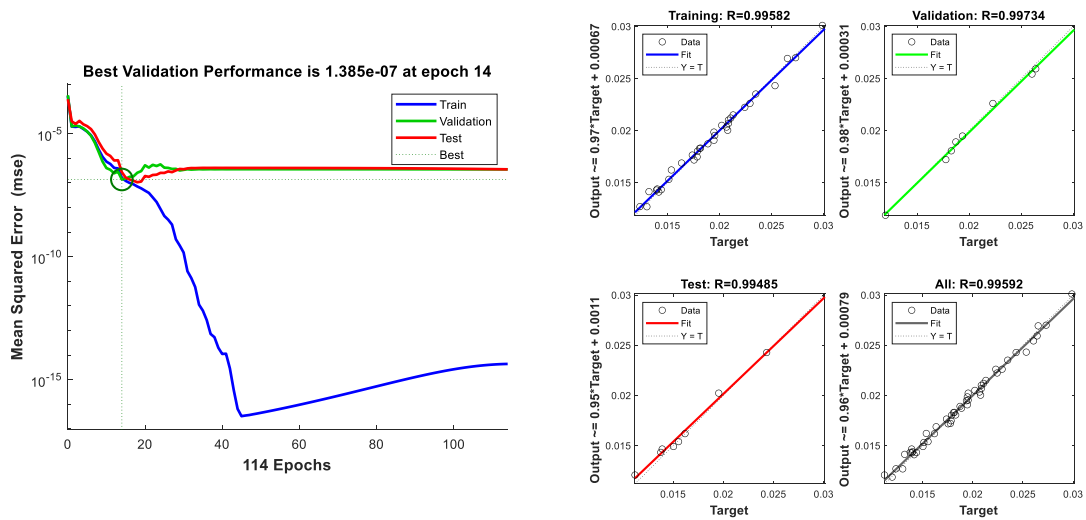


Figure 5.51. Performance metrics of the network that is trained by using 50 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

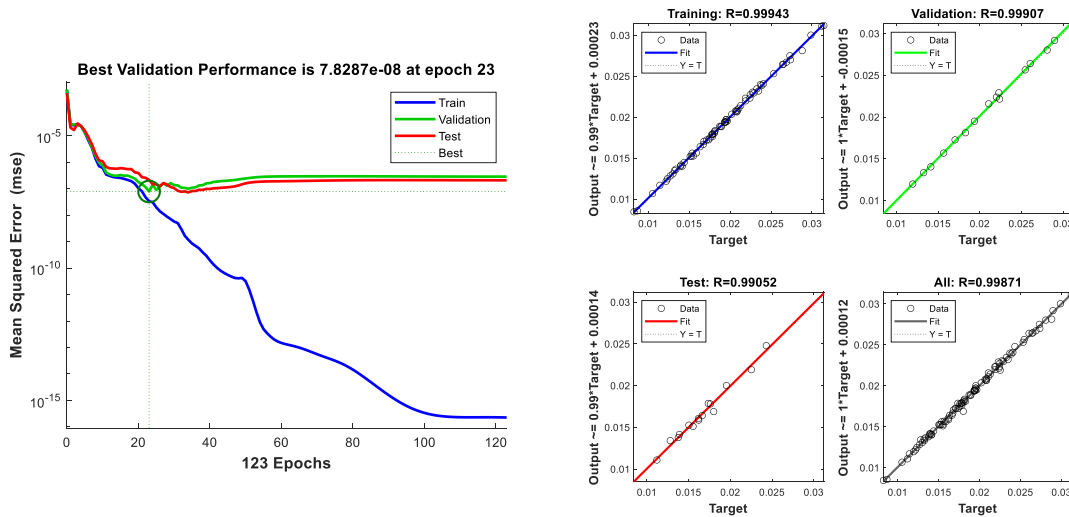


Figure 5.52. Performance metrics of the network that is trained by using 100 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

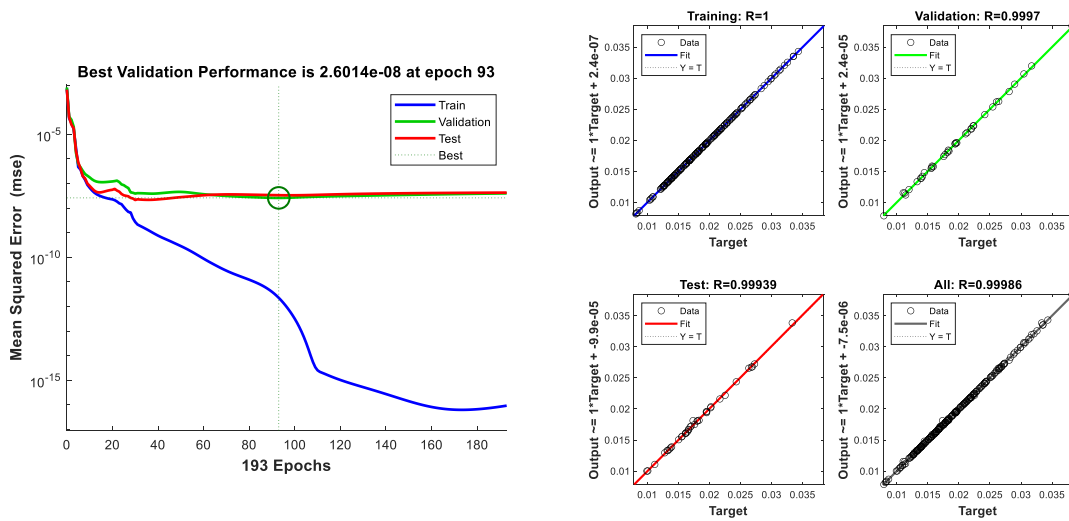


Figure 5.53. Performance metrics of the network that is trained by using 250 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).



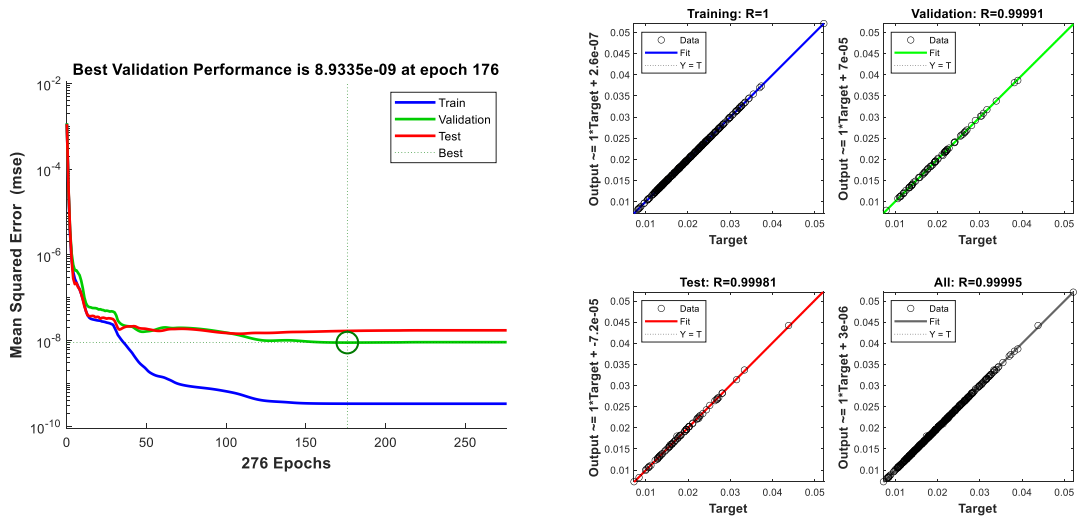


Figure 5.54. Performance metrics of the network that is trained by using 500 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

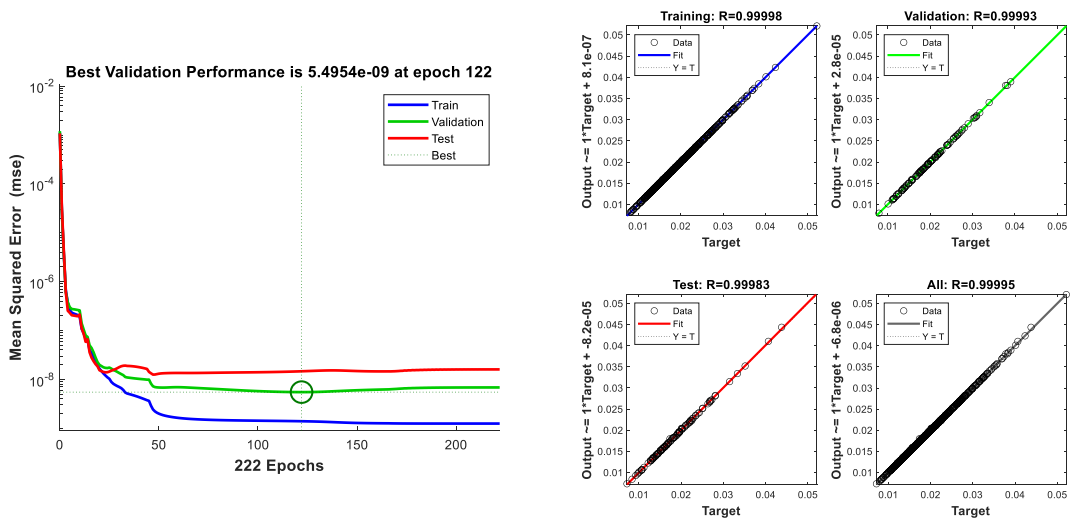


Figure 5.55. Performance metrics of the network that is trained by using 750 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

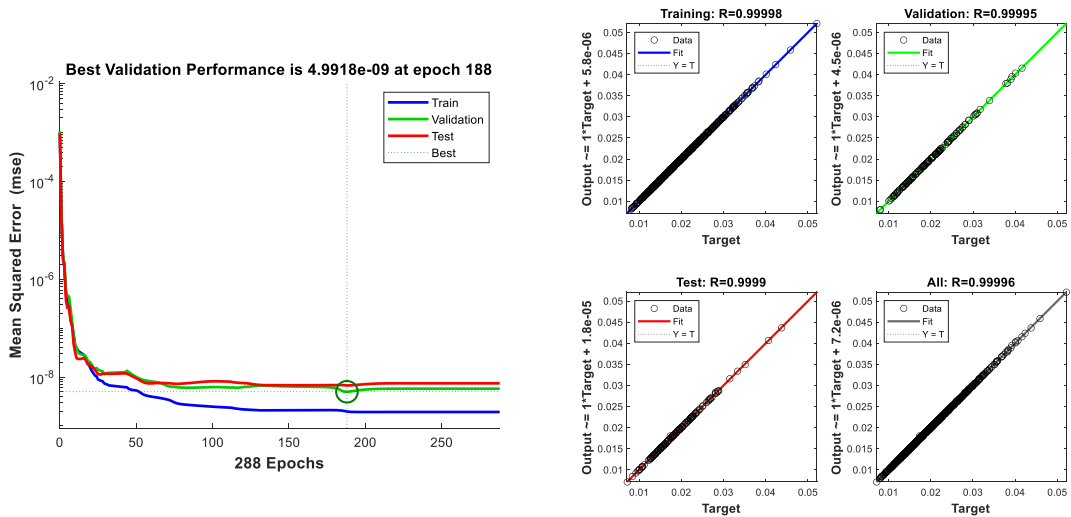


Figure 5.56. Performance metrics of the network that is trained by using 1000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

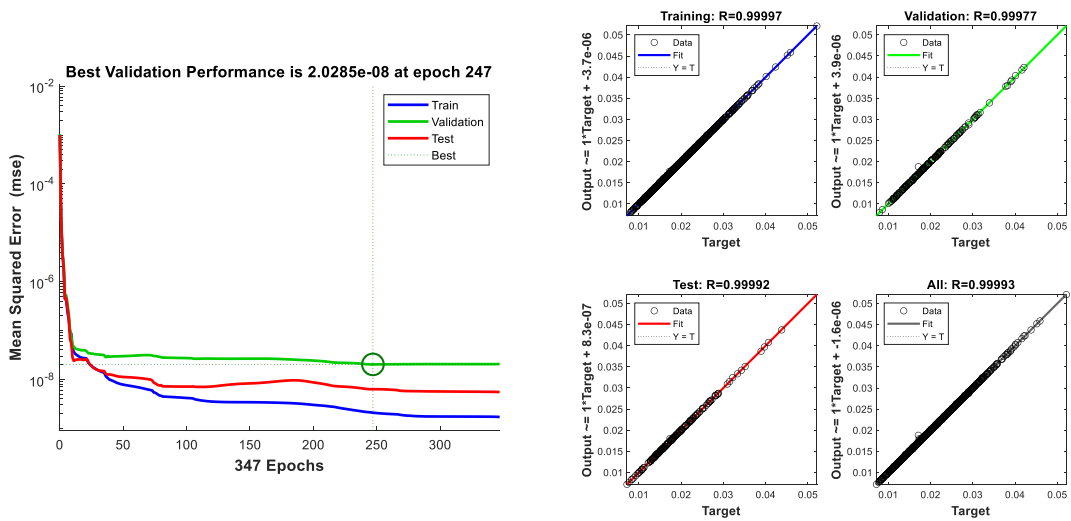


Figure 5.57. Performance metrics of the network that is trained by using 1250 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

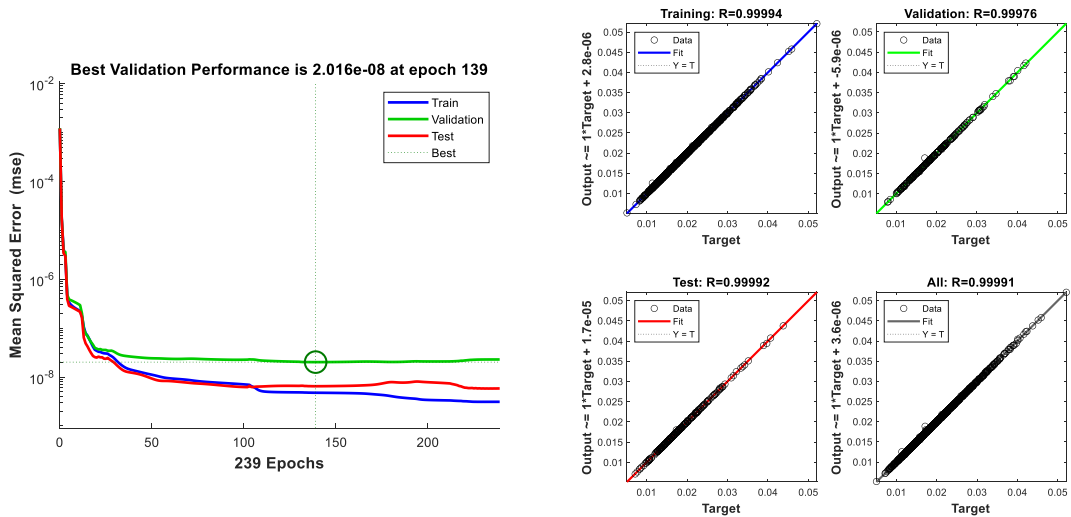


Figure 5.58. Performance metrics of the network that is trained by using 1500 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

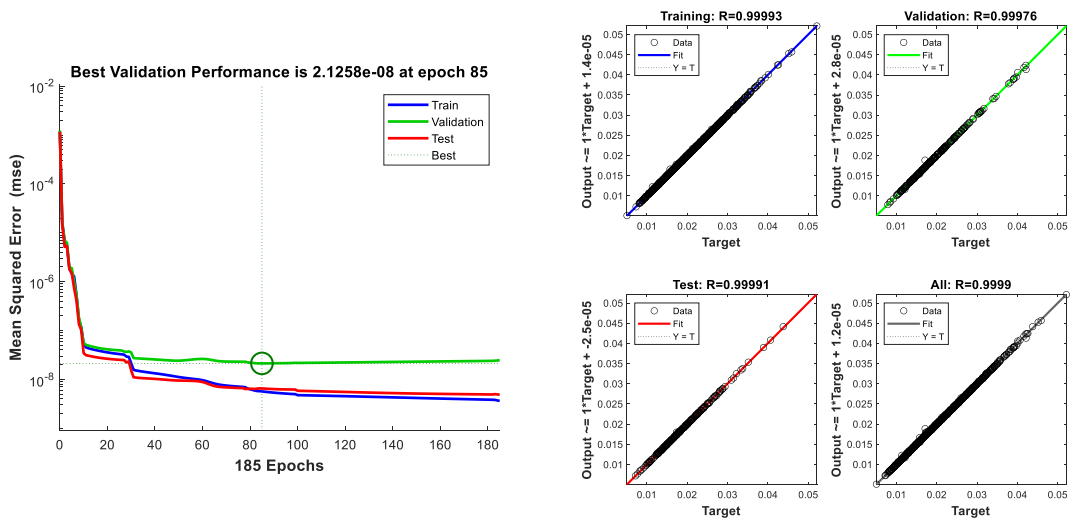


Figure 5.59. Performance metrics of the network that is trained by using 1750 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

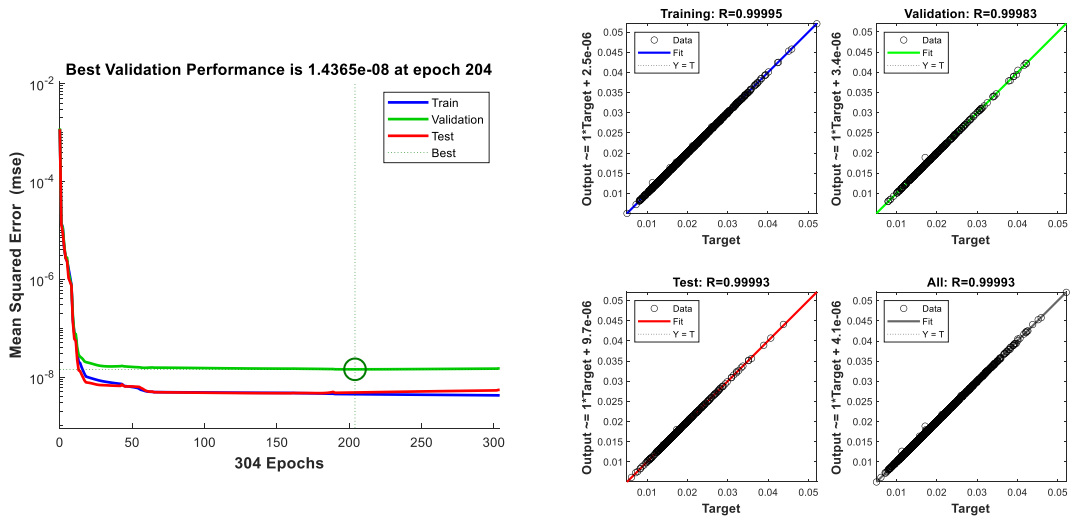


Figure 5.60. Performance metrics of the network that is trained by using 2000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

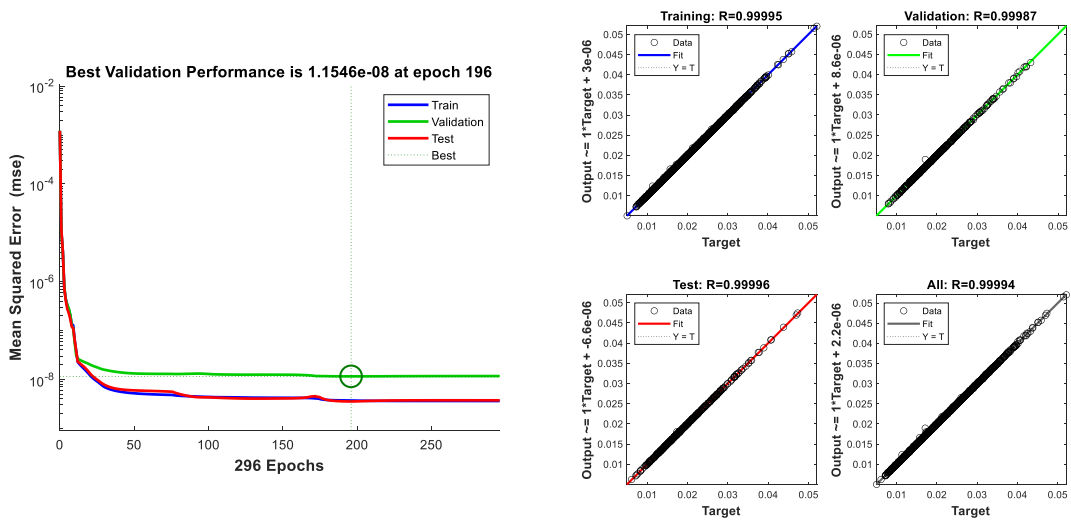


Figure 5.61. Performance metrics of the network that is trained by using 3000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

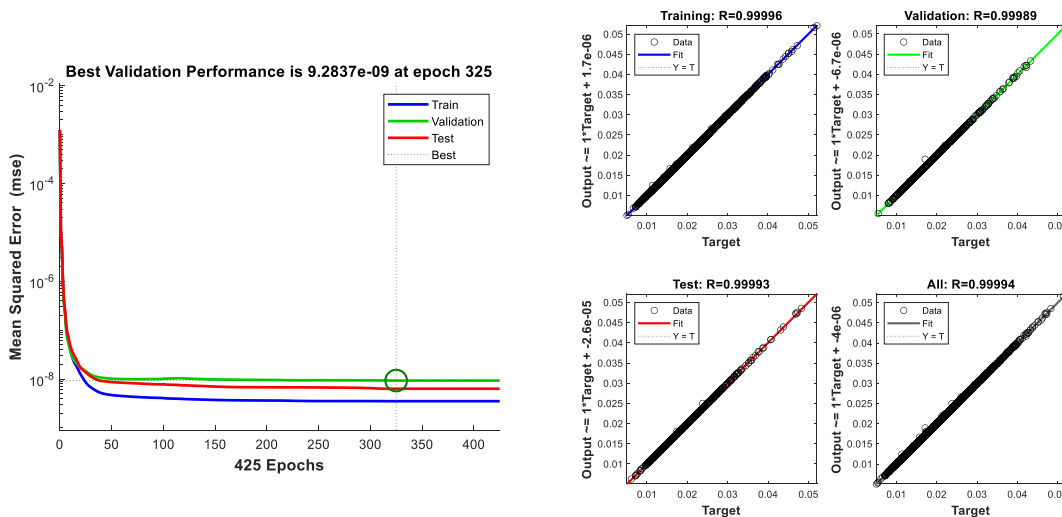


Figure 5.62. Performance metrics of the network that is trained by using 4000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

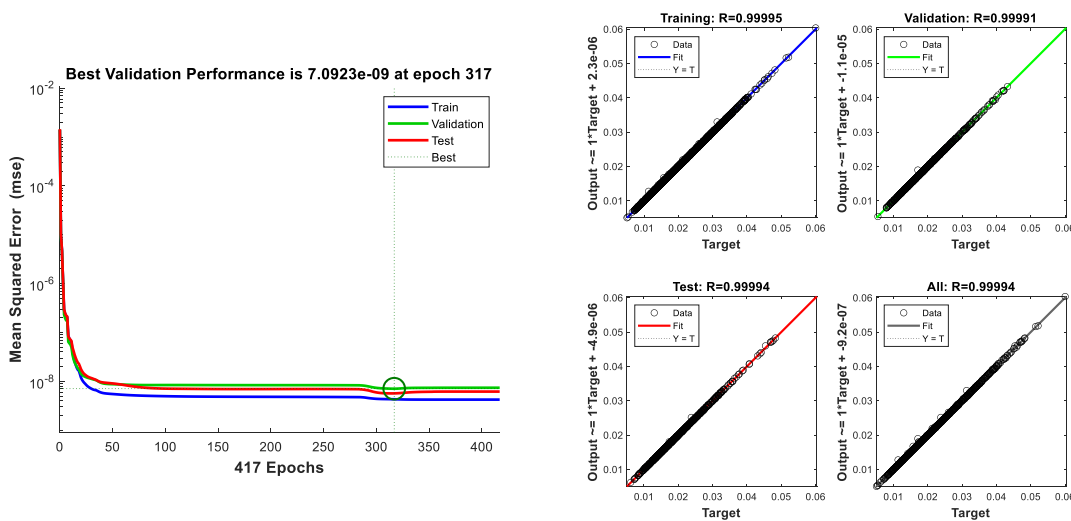


Figure 5.63. Performance metrics of the network that is trained by using 5000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

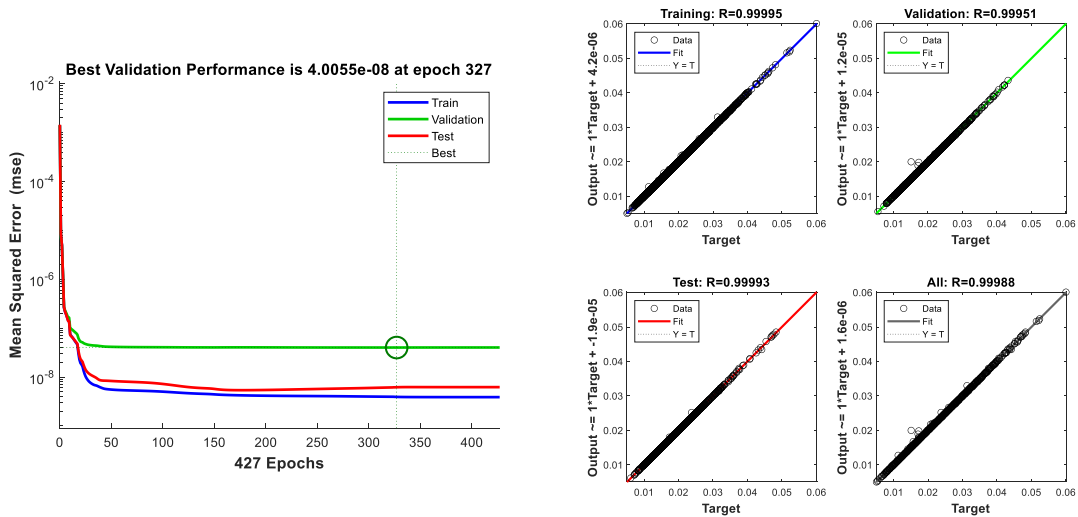


Figure 5.64. Performance metrics of the network that is trained by using 6000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

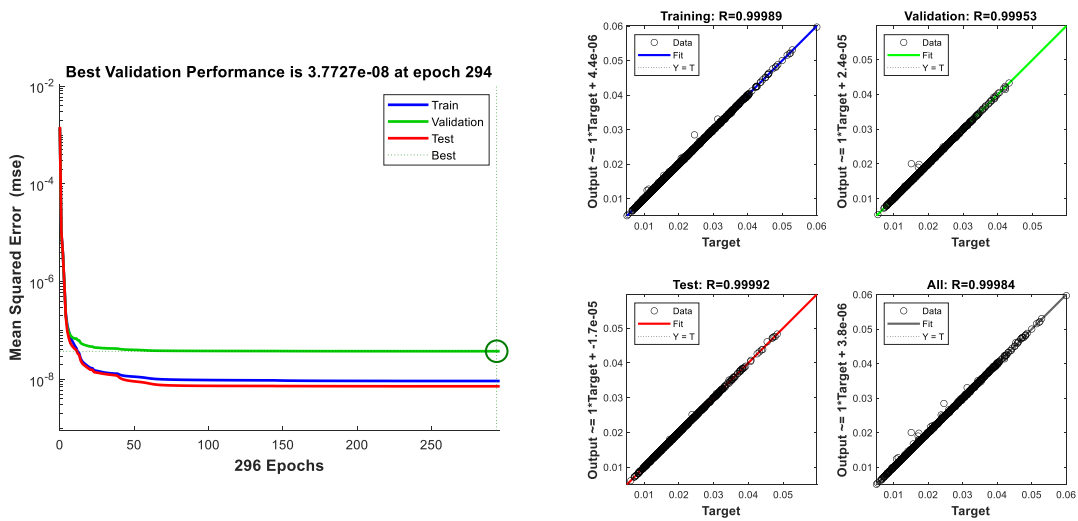


Figure 5.65. Performance metrics of the network that is trained by using 7000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

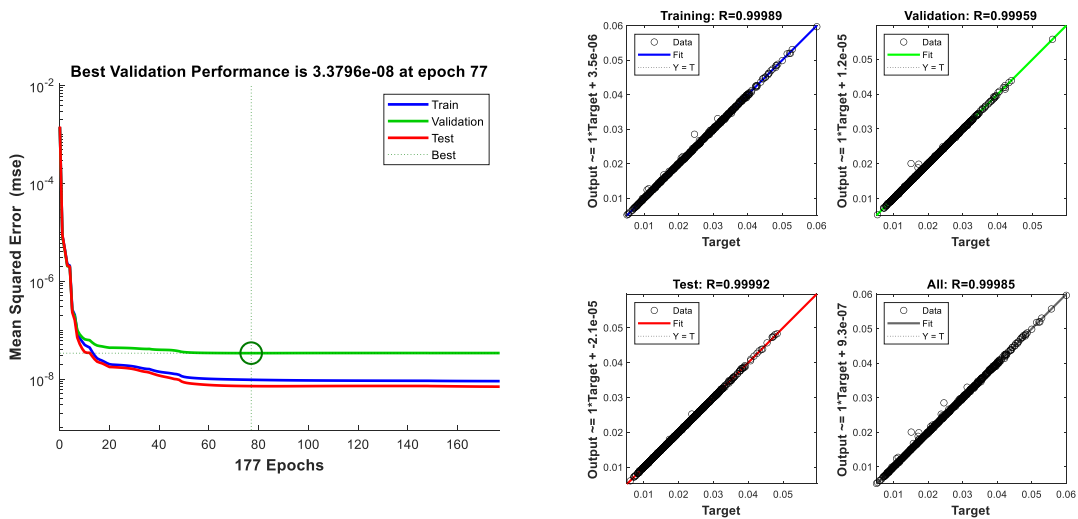


Figure 5.66. Performance metrics of the network that is trained by using 8000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

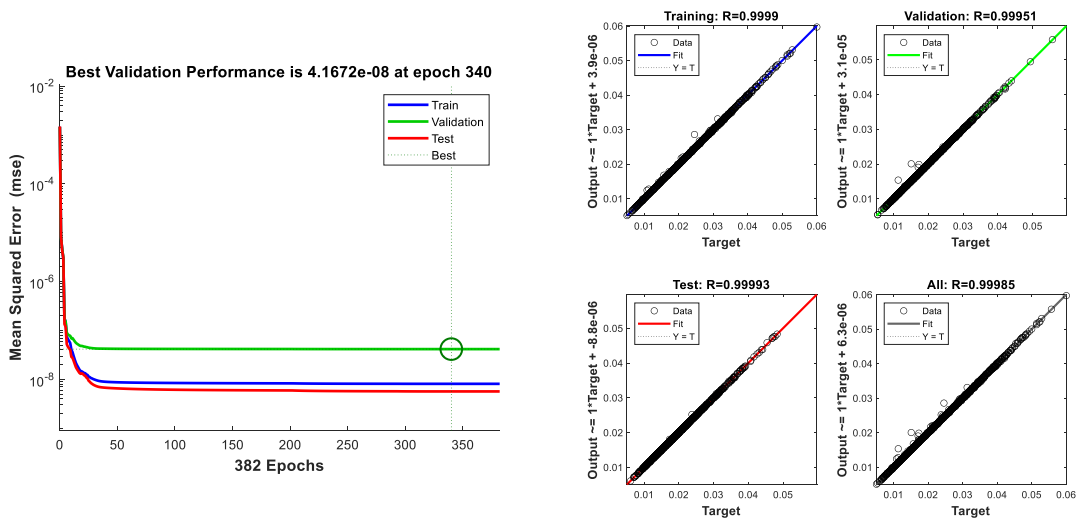


Figure 5.67. Performance metrics of the network that is trained by using 9000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

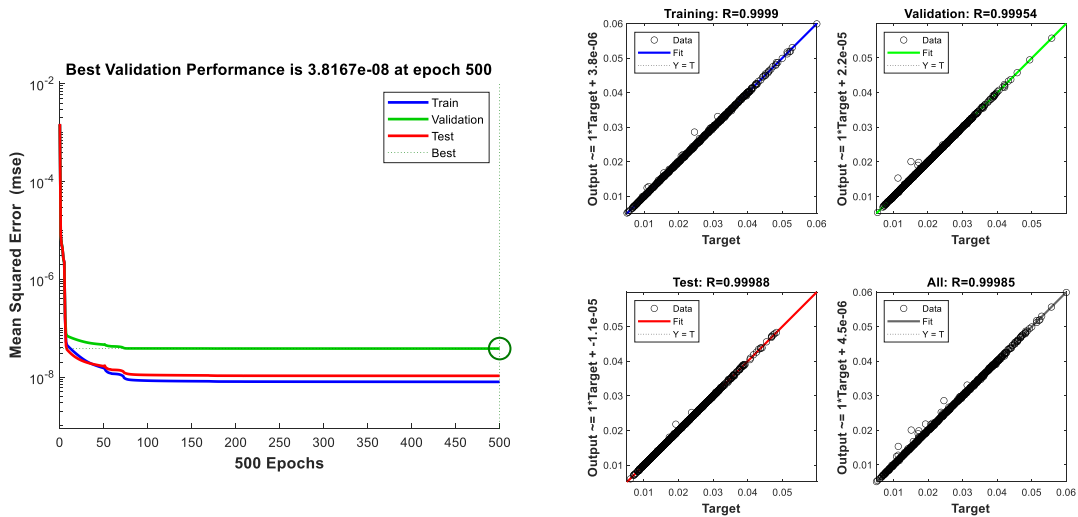


Figure 5.68. Performance metrics of the network that is trained by using 10000 samples for the 5-story 3-bay correlated frame structure example. MSE performance (on the left side), regression plots (on the right side).

The probability estimates for the example with respect to the increasing number of training dataset size are given in Figure 5.69. It can be observed that the estimated probability of failure converges to the result obtained by using CMCS with some amount of fluctuations up to 6000 samples in the dataset. Then, a sudden drop in the probability estimate was seen. The reason for the sudden drop is investigated in detail in Section 6.2.

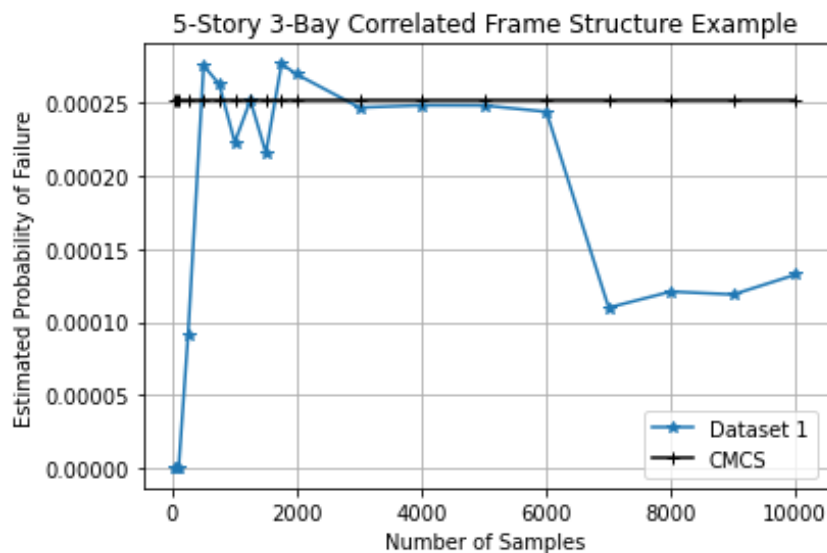


Figure 5.69. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the 5-story 3-bay correlated frame structure example.



## 5.6. A Real-Life 11-Story Reinforced Concrete Structure

A structure located in Kahramanmaraş/Türkiye is selected as a real-life example for this study. The structure has 11 stories and a roof story which is not included in the idealized model. The total height of the structure with the roof story is 37.75 meters whereas without the roof story 34.10 meters. The structure is constructed on sandy-silt soil having soil parameters as given in the table below.

Table 5.9. Soil Properties of the Construction Site

Parameter Name	Value	Unit
Soil Type	ZD (Sandy Soil-Silt Soil)	-
Corner Period for Horizontal Acceleration Spectrum (Elastic Design), $T_A$	0.100	sec.
Corner Period for Horizontal Acceleration Spectrum (Elastic Design), $T_B$	0.502	sec.
Mapped, Short Period, Spectral Response Acceleration Parameter, $S_S$	0.763	-
Mapped, 1.0 Second Period, Spectral Response Acceleration Parameter, $S_1$	0.210	-
Design, Short Period, Spectral Response Acceleration Parameter, $SD_S$	0.912	-
Design, 1.0 Second Period, Spectral Response Acceleration Parameter, $SD_1$	0.458	-
Peak Ground Acceleration, PGA	0.322	g
Peak Ground Velocity, PGV	19.922	cm/sec.

A horizontal acceleration response spectrum can be generated for the location as shown in Figure 5.70 based on the provided soil parameters.

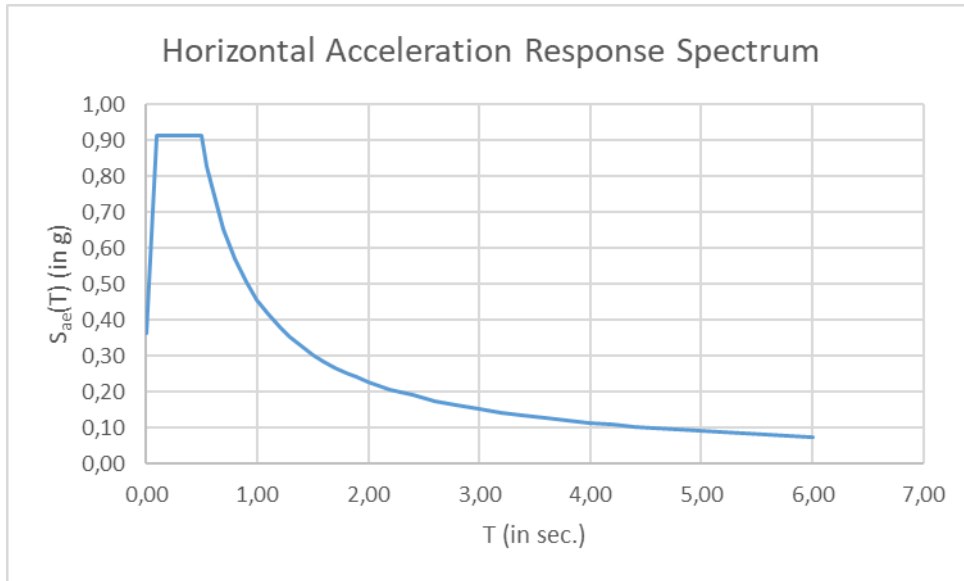


Figure 5.70. The horizontal response spectrum based on the given soil properties.

The material properties for the materials used in the structure are given in Table 5.10.

Table 5.10. Material properties.

Parameter	Value	Unit
Characteristic Compressive Strength, Concrete, $f_{ck}$	30	MPa
Characteristic Tensile Strength, Concrete, $f_{ctk}$	1.917	MPa
Resistance Factor, Concrete, $\gamma_c$	1.5	-
The Factor of Equivalent Rectangular Stress Distribution, $k_1$	0.82	-
Characteristic Yield Strength, Longitudinal Reinforcement, $f_{yk}$	420	MPa
Characteristic Yield Strength, Transverse Reinforcement, $f_{ywk}$	420	MPa
Resistance Factor, Concrete, $\gamma_s$	1.15	-
Modulus of Elasticity, E	31800	MPa
Shear Modulus, G	13250	MPa
Poisson Ratio, $\mu$	0.20	-
Unit Weight, Concrete	2.5	tonf/m <sup>3</sup>
Coefficient of Thermal Expansion	1E-5	-

The section views of the structure are provided in Figure 5.71 and Figure 5.72 to provide information about the architectural configuration of the structure. The floor plans of the ground and mezzanine floors are given in Figure 5.73 and Figure 5.74, respectively. The ground floor has 4 meters in height whereas the mezzanine floor has 3.1 meters. Furthermore, they have additional two frames with respect to the normal floor plan.

The normal floors including the ninth floor have 3 meters in height as depicted in Figure 5.75 and Figure 5.76. The roof floor, on the other hand, has fewer frames relative to the other floors and has the same height as normal floors. The detailed configuration of the roof floor is provided in Figure 5.77.

As can be seen from the floor plans, the structure has shear walls against the lateral loads and they provide large amount stiffness of to the system due to their lengths and directions. However, a coupled shear wall exists in Grid D that attracts most of the lateral forces onto it. Even though this observation yields a guess about the stiffest frame in the structure, it is necessary to prove the guess mathematically. Determination of the stiffest grid in the structure is performed in Section 5.6.2.

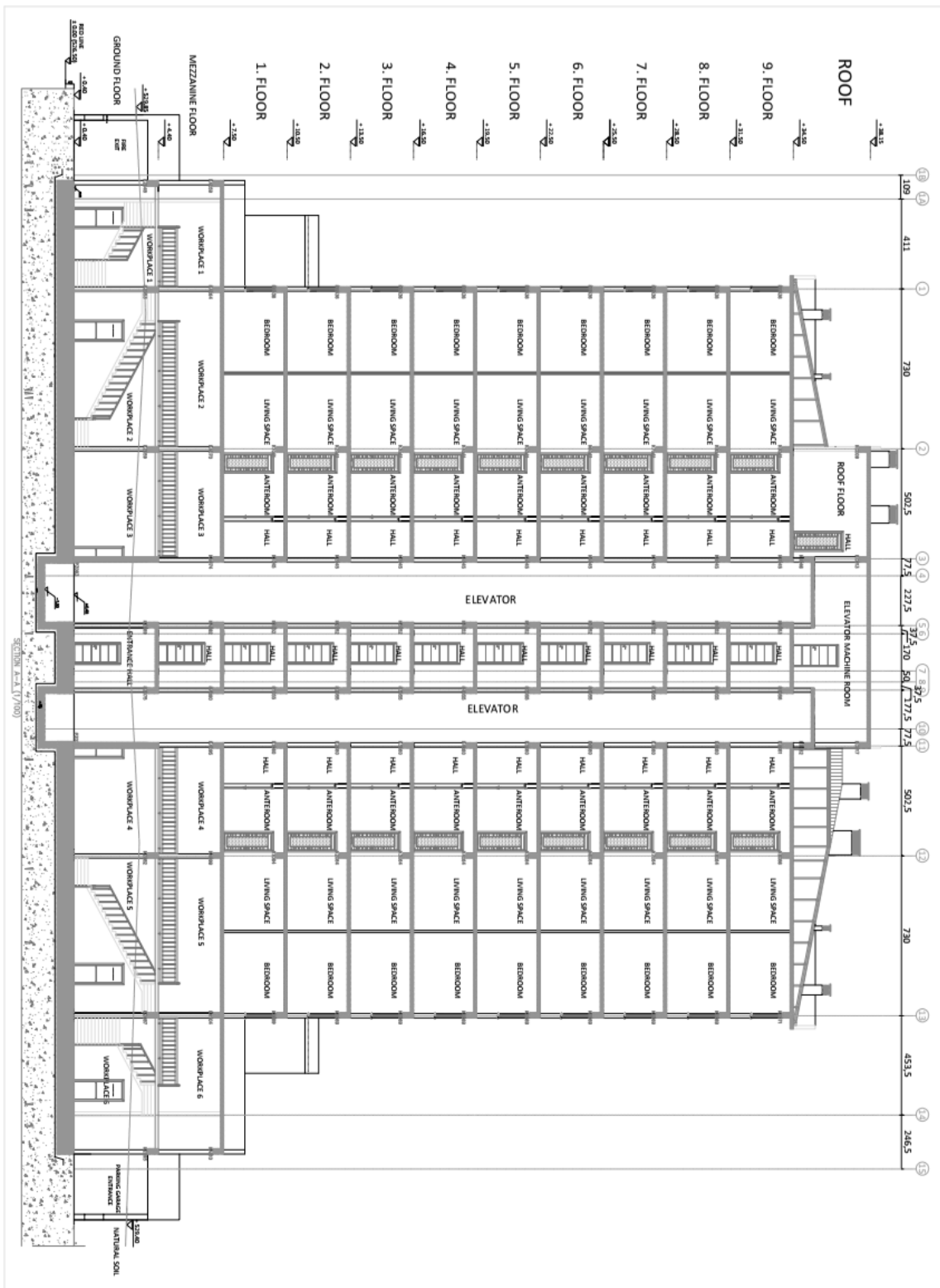


Figure 5.71. Section view of the structure in North-East direction.

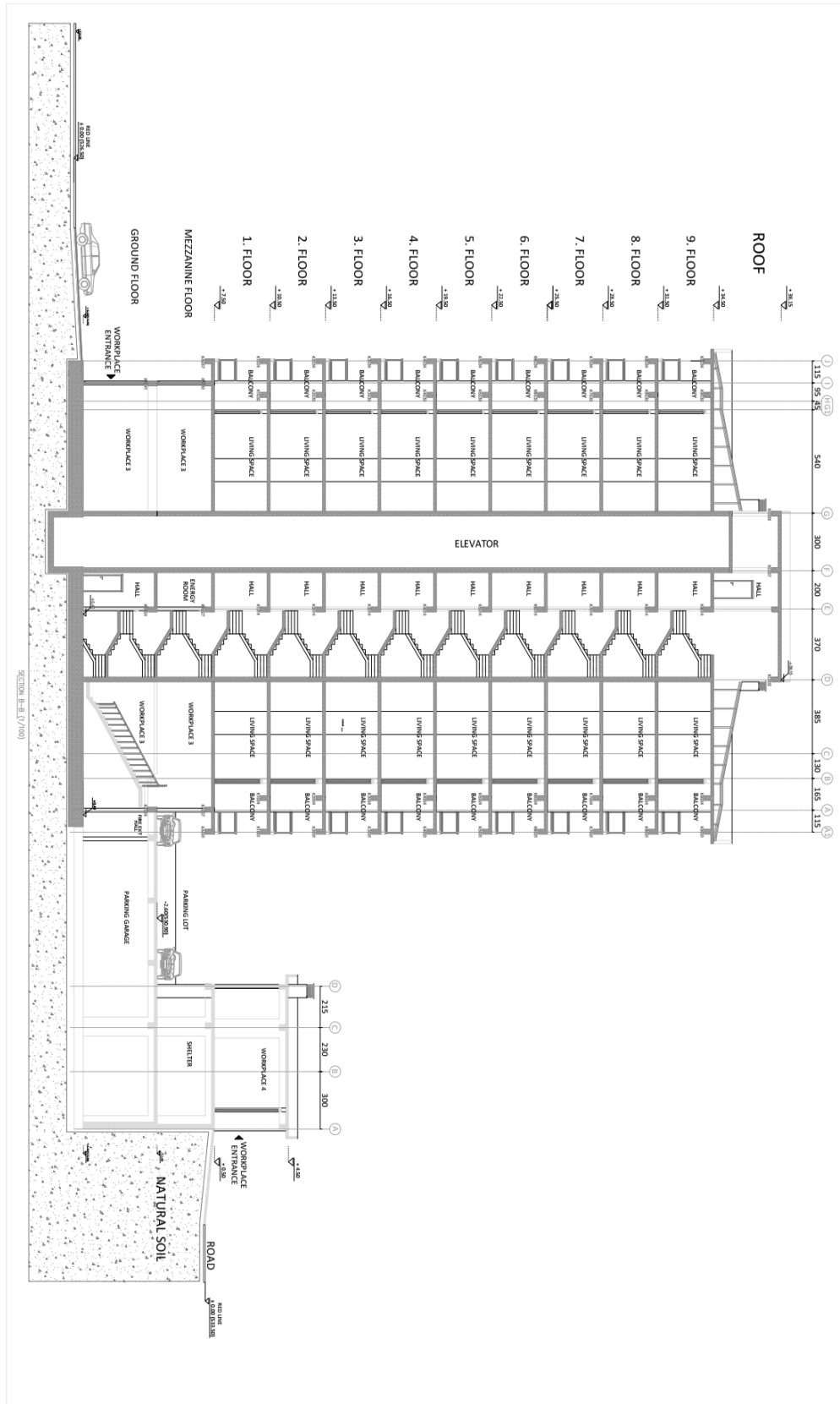


Figure 5.72. Section of the structure in North-West direction.

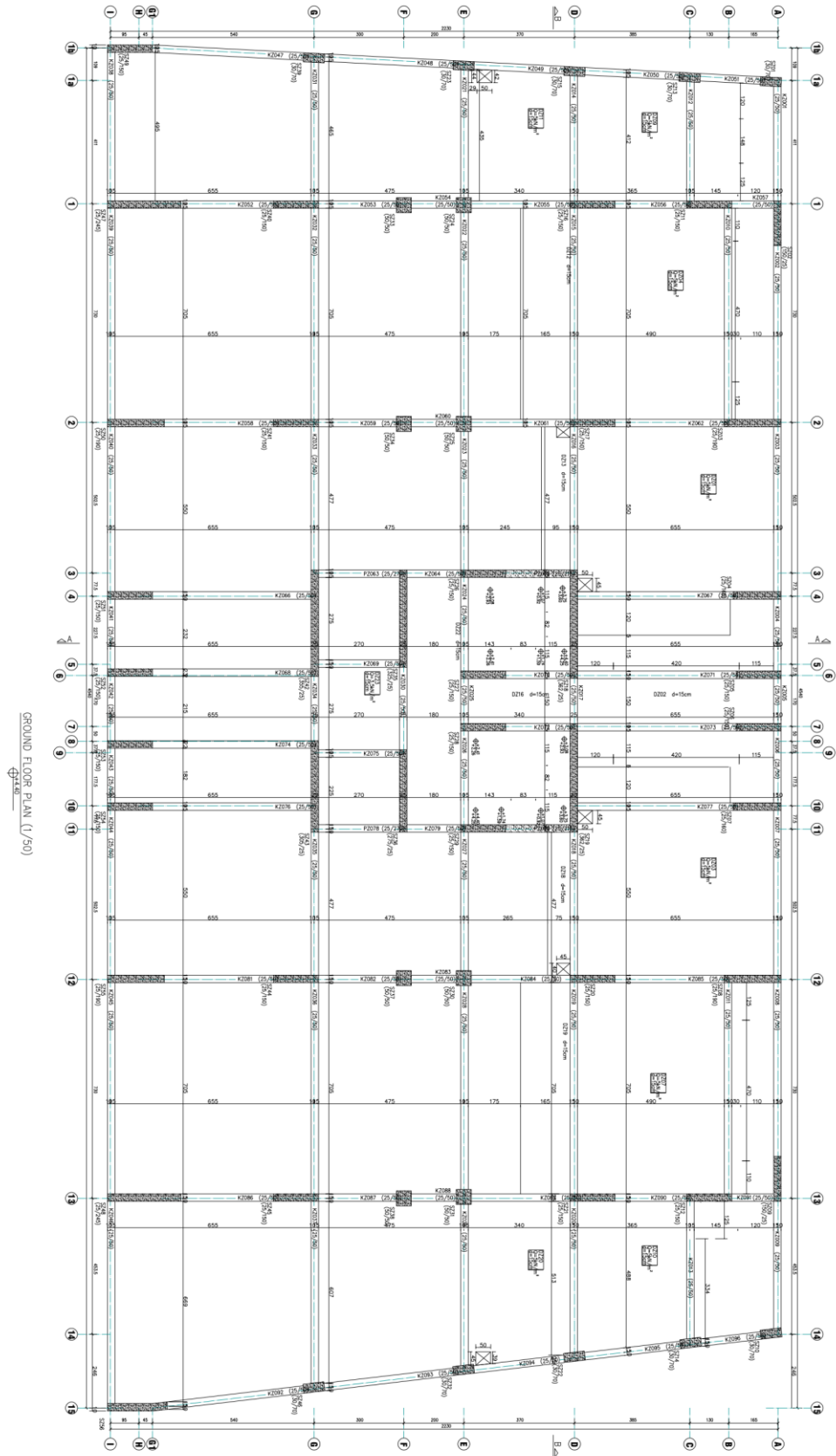


Figure 5.73. Ground floor plan.

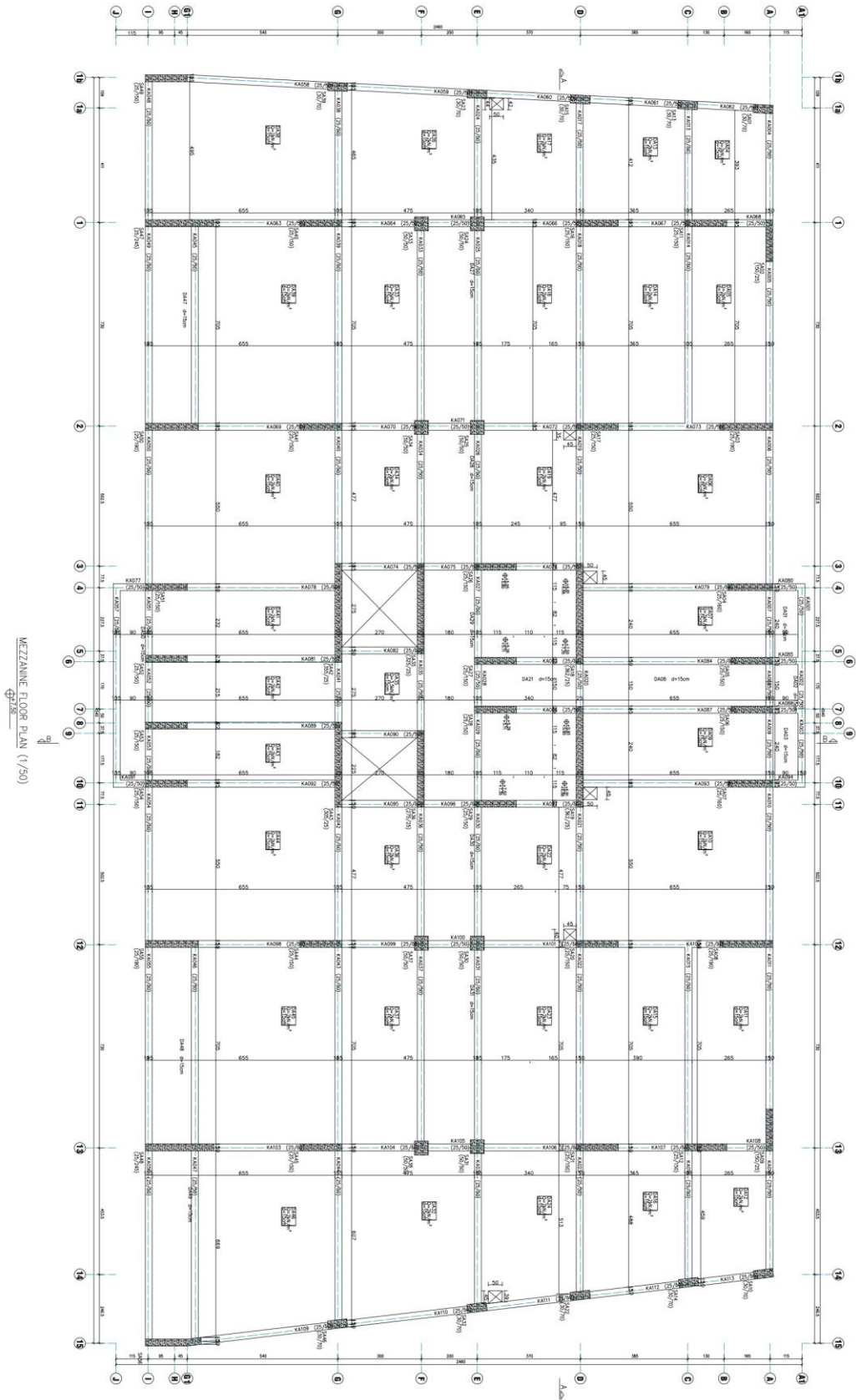


Figure 5.74. Mezzanine floor plan.

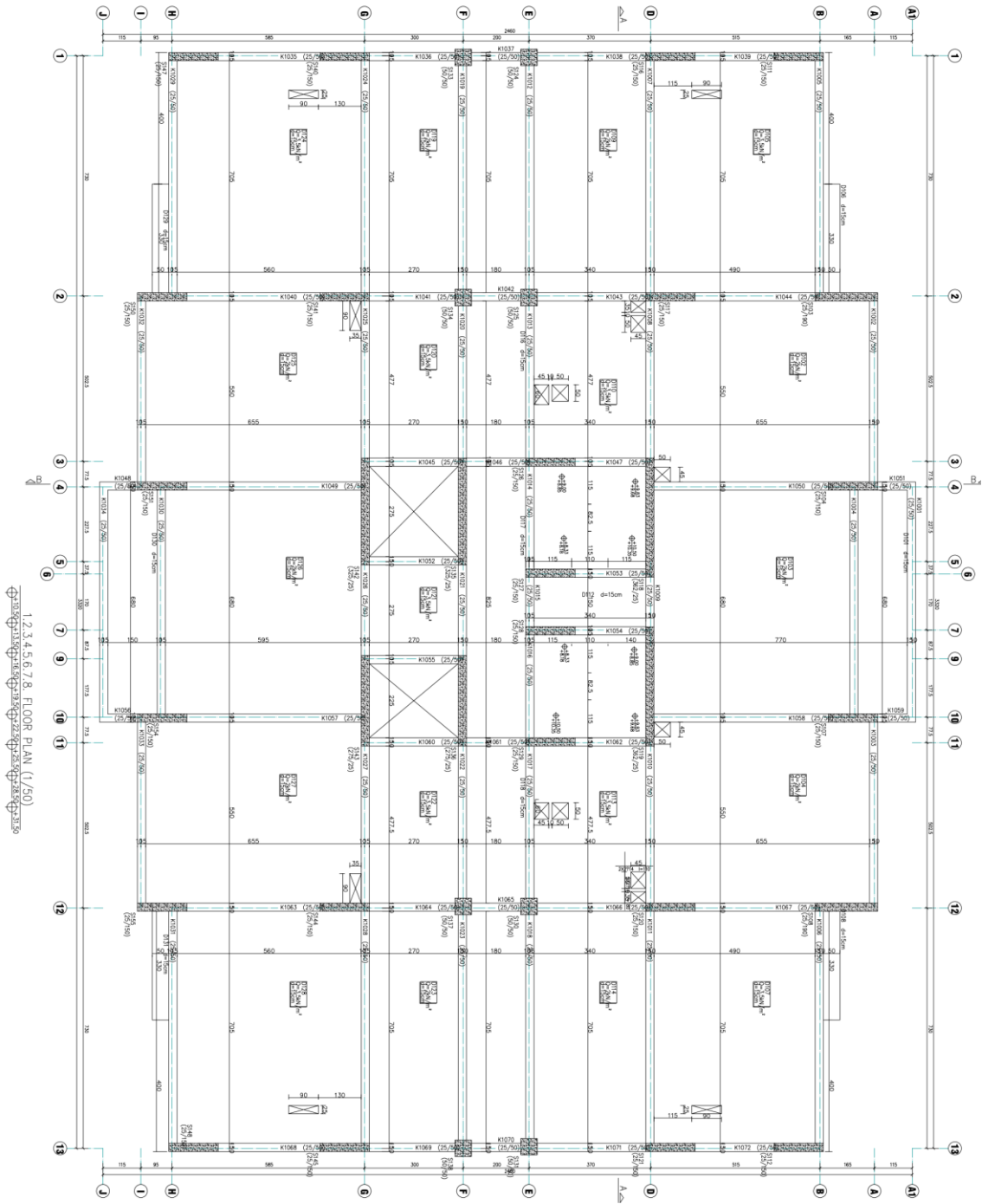


Figure 5.75. Normal floor plan.



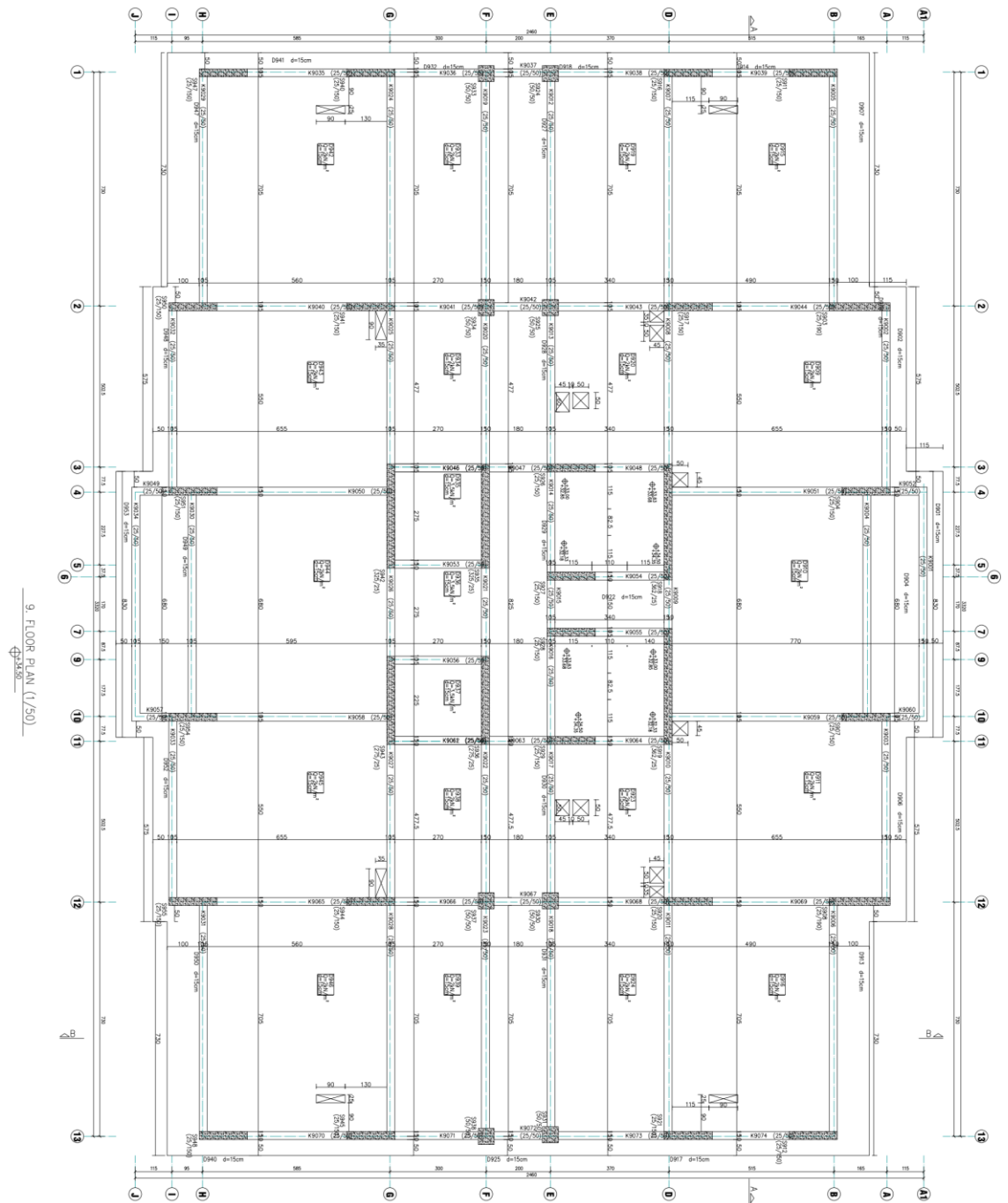


Figure 5.76. Ninth floor plan.

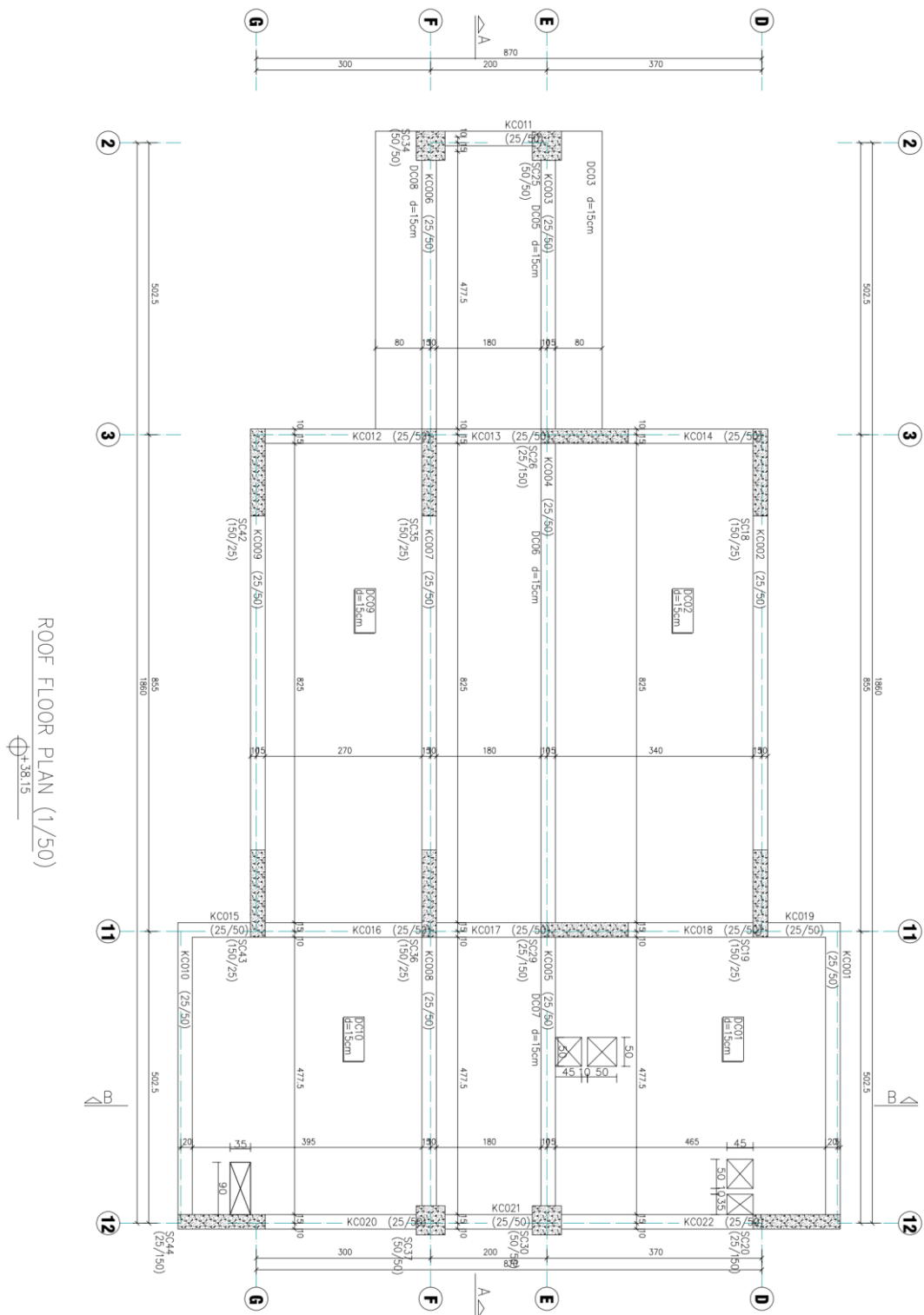


Figure 5.77. Roof floor plan.

### 5.6.1. Lateral Loads and Selection of the Representative Frame

The earthquake load and wind load acting on the structure have been provided as a report companion to the drawings by the engineer. The calculated wind loads for each story are given in Table 5.11 whereas the earthquake loads calculate based on the modal superposition and equivalent earthquake load methods are provided in Table 5.12.

Table 5.11. Wind loads acting on the structure.

Floor	X Direction (kN)	Y Direction (kN)
Roof	53.35	92.13
9	100.19	133.59
8	95.54	128.93
7	95.54	128.93
6	95.54	128.93
5	95.54	128.93
4	69.48	93.77
3	69.48	93.77
2	69.48	93.77
1	69.48	93.77
Mezzanine	44.88	82.81
Ground	52.49	106.85

Table 5.12. Calculated earthquake loads for the structure.

Floor	X Direction			Y Direction		
	Modal Super Position (kN)	Equivalent Earthquake Loading (kN)	Selected Load (kN)	Modal Super Position (kN)	Equivalent Earthquake Loading (kN)	Selected Load (kN)
Roof	284.82	378.92	284.82	345.92	421.78	345.92
9	624.05	407.40	624.05	649.18	453.47	649.18
8	384.32	400.37	384.32	434.37	445.64	434.37
7	220.59	361.75	220.59	276.10	402.66	276.10
6	154.10	323.12	154.10	206.54	359.67	206.54
5	136.11	284.50	136.11	178.88	316.69	178.88
4	133.56	245.88	133.56	162.26	273.69	162.26
3	140.51	207.26	140.51	169.85	230.70	169.85
2	161.46	168.64	161.46	183.31	187.72	183.31
1	178.23	130.03	178.23	194.70	144.73	194.70
Mezzanine	388.41	123.40	388.41	402.91	137.36	402.91
Ground	287.14	48.67	287.14	249.53	54.17	249.53
Total	3097.75	3079.94	3097.75	3453.59	3428.29	3453.59

The stiffest grid on the structure can be obtained by using the dimensions of the columns and shear walls in the structure. The percentage for sharing lateral loads for each grid is provided in Table 5.13.

Table 5.13. Percentage of loads resisted by each grid.

Grid	Ground Floor (%)	Mezzanine Floor (%)	Normal Floors (%)
A	3.19	3.19	0.20
B	0.08	0.08	0.09
C	0.06	0.06	0.00
D	40.49	40.48	45.47
E	0.65	0.65	0.69
F	23.89	23.81	26.79
G	31.16	31.24	26.49
H	0.00	0.00	0.09
I	0.48	0.48	0.18
Total	100.00	100.00	100.00

The stiffest grid in the structure is Grid D as expected previously. The earthquake loads acting on the grid at each floor level then can be calculated by multiplying the loads given in Table 5.12 and the percentages provided in Table 5.13. The calculated loads for the grid are indicated in Table 5.14.

Table 5.14. Earthquake loads acting on Grid D.

Floor	$F_x$ (kN)	$F_y$ (kN)
Roof	129.48	157.26
9	283.70	295.16
8	174.73	197.52
7	102.33	125.51
6	70.05	93.90
5	61.89	81.33
4	60.73	73.79
3	63.89	77.22
2	73.39	83.33
1	81.01	88.50
Mezzanine	157.23	163.10
Ground	116.25	101.01
Total	1374.68	1537.63

It is known that Grid D is the stiffest grid in the structure based on the percentages provided in Table 5.13. Therefore, the frame in Grid D is going to attract most of the lateral forces itself and it will become the most critical frame and it can represent the behavior of the structure. For that reason, it is reasonable to select Grid D for the reliability analysis. The geometrical configuration of Grid D is shown in Figure 5.78, and the configuration will be used as a basis for the development of a real-life 11-story reinforced concrete structure problem. The proposed frame for the configuration illustrated in Figure 5.78 is given in Figure 5.79. In Figure 5.79, the elements denoted with R.L represent the assigned rigid links between the shear walls and adjacent beam elements.



## 5.6.2. Determination of Random Variables and Limit State Function

The random variables that will be used in the problem can be considered in three categories based on the origin of uncertainty. The uncertainties related to material, geometry, and load will be considered in this problem. Therefore, the modulus of elasticity of the concrete and its Poisson ratio, the cross-sectional area of the frame sections, and lateral loads can be taken into account as random variables. The following table contains the random variables that will be used in the problem.

Table 5.15. The list of random variables for real-life RC frame structure.

Random Variable	Units	Mean Value	CoV	Distribution Type
E	kN/m <sup>2</sup>	3.18E7	0.10	Normal
$\mu$	-	0.20	0.10	Normal
b <sub>1</sub>	m	0.70	0.05	Normal
b <sub>2</sub>	m	1.50	0.05	Normal
b <sub>3</sub>	m	0.25	0.05	Normal
h <sub>1</sub>	m	0.30	0.05	Normal
h <sub>2</sub>	m	0.25	0.05	Normal
h <sub>3</sub>	m	0.50	0.05	Normal
P <sub>11</sub>	kN	283.70	0.20	Lognormal
P <sub>10</sub>	kN	174.73	0.20	Lognormal
P <sub>9</sub>	kN	102.33	0.20	Lognormal
P <sub>8</sub>	kN	70.05	0.20	Lognormal
P <sub>7</sub>	kN	61.89	0.20	Lognormal
P <sub>6</sub>	kN	60.73	0.20	Lognormal
P <sub>5</sub>	kN	63.89	0.20	Lognormal
P <sub>4</sub>	kN	73.39	0.20	Lognormal
P <sub>3</sub>	kN	81.01	0.20	Lognormal
P <sub>2</sub>	kN	157.23	0.20	Lognormal
P <sub>1</sub>	kN	116.25	0.20	Lognormal



The limit state function can be constructed by considering a serviceability limit state function for infill walls. The residents experience high panic when cracks on infill walls are developed or they fear entering their building if heavy damage/failure in the walls occurs. This fear causes post-earthquake problems such as accommodation problems and renovation problems for a region affected by an earthquake and has a population of up to millions. Those problems related to fear and panic are observed by the author in the recent Türkiye-Syria earthquake.

In Chapter 4.9 of 2018 Turkish Seismic Code (TSC2018) maximum limits for inter-story drift ratio are defined based on the connection of infill walls to structural elements. If there is no connection between the infill walls and structural elements exist then, the maximum inter-story drift ratio for a structure is defined as:

$$\lambda \frac{\delta_{i,max}^{(X)}}{h_i} \leq 0.008\mathcal{K} \quad (5.6)$$

where  $h_i$  represents the story height,  $\lambda$  is the ratio of spectral accelerations at the period of the structure for DD-3 and DD-2 earthquake levels defined in the code and it is approximately equal to 0.404. The constraint,  $\mathcal{K}$ , is taken as 1.0 for reinforced concrete structures, and  $\delta_{i,max}^{(X)}$  is defined as the largest effective inter-story drift.

The inter-story drift is calculated in TSC2018 as follows:

$$\delta_{i,max}^{(X)} = \frac{R}{I} \Delta_i^{(X)} \quad (5.7)$$

where  $R$  is a coefficient that describes the structural system behavior based on its expected ductility.  $R$  is taken as 8 if the structure is to be designed as ductile coupled walls and ductile column elements. The building importance factor,  $I$ , is taken as 1.0 for residential-type structures. The displacement between successive two floors is denoted by  $\Delta_i^{(X)}$  and calculated as:

$$\Delta_i^{(X)} = u_i^{(X)} - u_{i-1}^{(X)} \quad (5.8)$$

where  $u_i^{(X)}$  represents the displacement at the upper node of a column or a shear wall element.

The same equation can also be used as follows if a connection between the infill walls and structural elements is provided.

$$\lambda \frac{\delta_{i,max}^{(X)}}{h_i} \leq 0.016\mathcal{K} \quad (5.9)$$

In this real-life frame problem, it is decided that the limit state function can be constructed by using equation (5.9). The limit state function for the problem can be written as follows if the equation is arranged as below:

$$g(X) = 0.008 - \max_{\substack{i=1 \text{ to total} \\ \text{number of stories}}} \frac{(0.404)\delta_{i,max}^{(X)}}{h_i} \quad (5.10)$$

### 5.6.3. Modeling of the Frame in OpenseesPy

The frame structure was modeled in 2D in OpenseesPy by using “*forceBeamColumn*” elements with elastic sections. It is assumed that the frame elements stay within their linear elastic limits under the defined loads. Furthermore, the deflections that occurred due to shear forces were also considered by imposing shear modulus and shear area to the elastic sections. The shear modulus was calculated by using the modulus of elasticity and Poisson’s ratio.

Shear walls in the frame were modeled by using the wide column approach and they linked to adjacent beams via rigid links with beam type defined in OpenseesPy. The wide column approach was selected due to observed mesh sensitivity problems related to the drilling degree of freedom when shell elements are used. A shear wall in the frame was isolated from other elements to compare the performances of shell modeling (by using ShellMITC4 shell elements available in OpenseesPy) and the wide column approach by imposing the mean value of the loads acting at story levels. The exaggerated deflected shapes of the two approaches are shown in Figure 5.80.

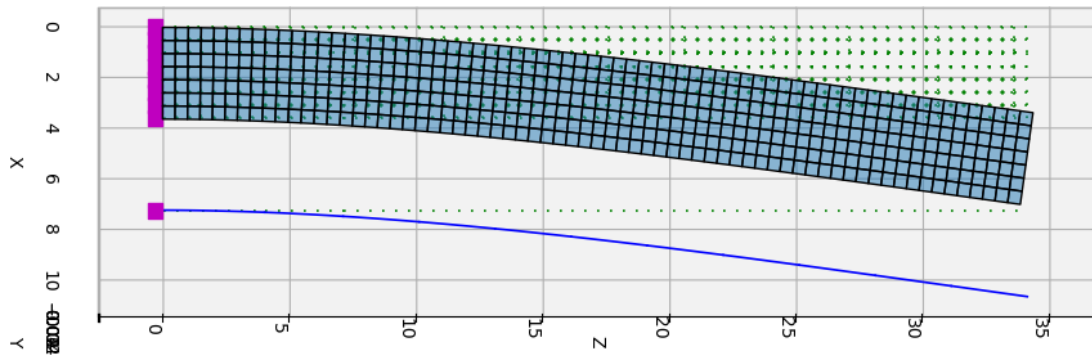


Figure 5.80. Deflected shape of shell elements and equivalent frame element.

The tip displacements of the two approaches are 27.74 cm and 28.11 cm for the equivalent wide column and shell elements respectively. The difference between the displacement of the two models is equal to 1.31 percent. Based on the results, it can be said that the equivalent frame elements can be used in the structure to model the shear walls for the purpose of this study.

In the model, the effective stiffness multipliers stated by TSC2018 were used. For columns, the moment of inertia was multiplied by 0.7. For regular beams, the moment of inertia multiplied by 0.35 whereas the moment of inertia for coupling beams was multiplied by 0.15. Multipliers 0.5 and 0.5 were used to obtain effective stiffness in bending and shear for the equivalent frame element that is used for modeling shear walls.

The model of the frame based on the wide column approach is given in Figure 5.81 whereas the deformed shape of the model when the mean value of loads is applied is shown in Figure 5.82.

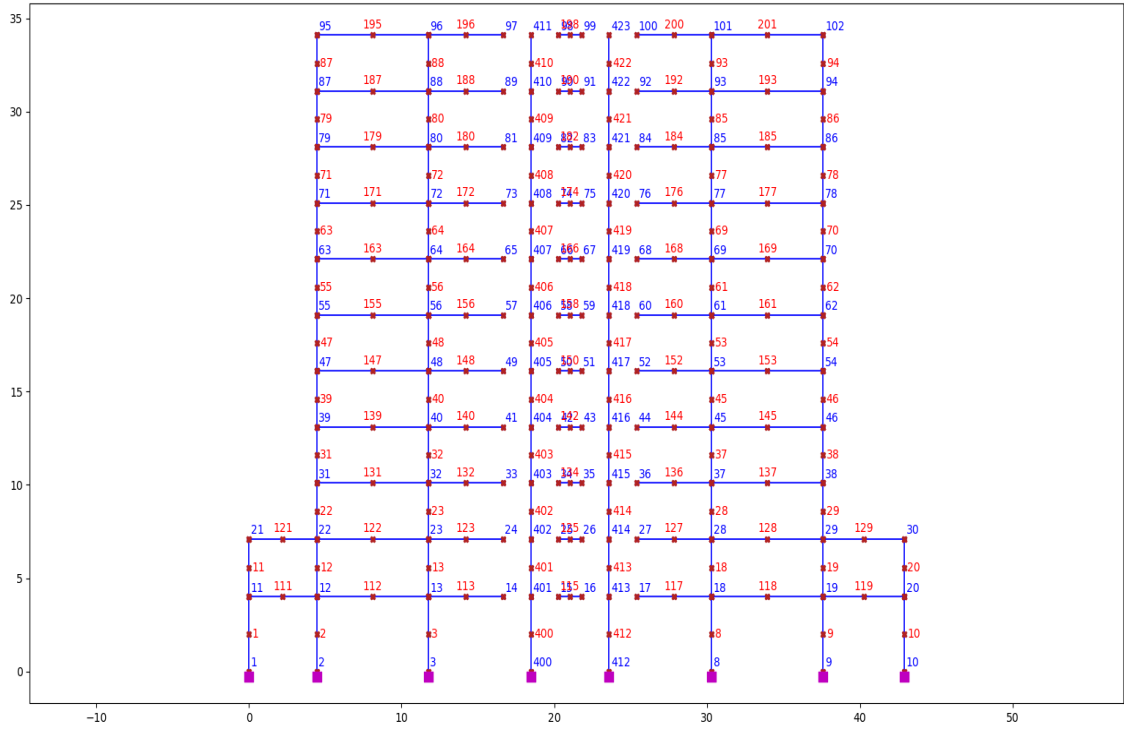


Figure 5.81. Frame model built in OpenseesPy.

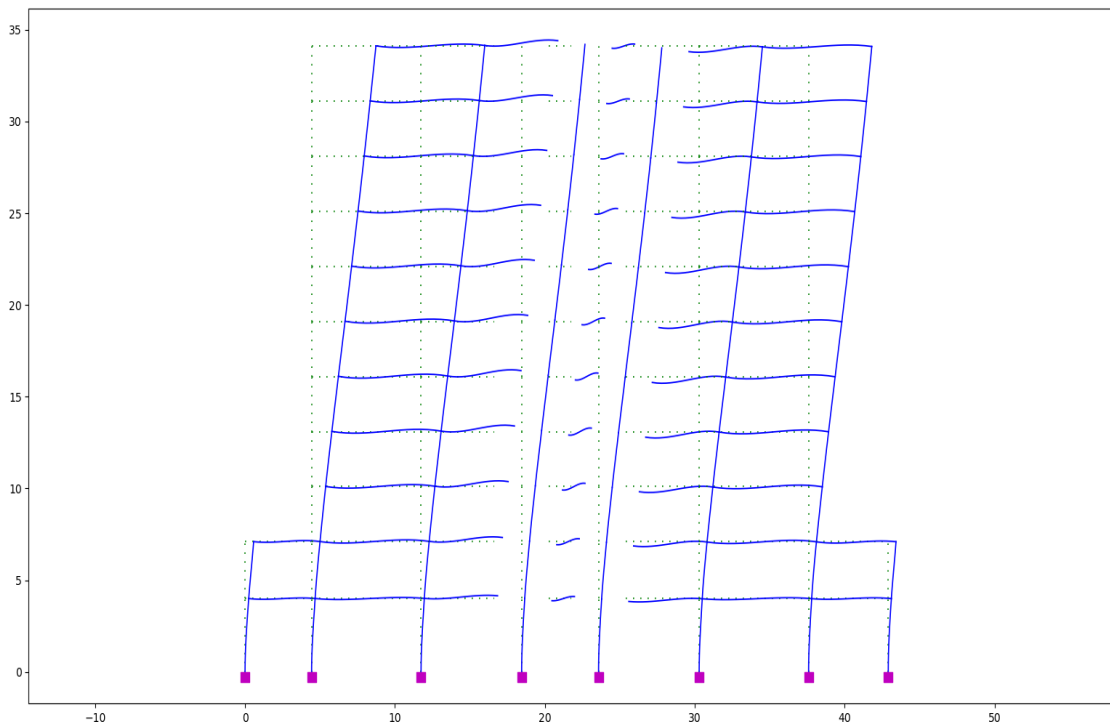


Figure 5.82. Deformed shape of the frame.

Another important issue to consider in the modeling of shear walls as equivalent wide columns is checking the amount of rotations at the ends of rigid links to validate the small rotation assumption for rigid links in OpenseesPy. Based on the analysis results of the model shown in Figure 5.81, rotation at both ends of the rigid links are provided in Table 5.16.

Table 5.16. Rotation at both ends of rigid links.

Floor	Rotation at the Left End of Rigid Link on the Left	Rotation at the Right End of Rigid Link on the Left	Rotation at the Left End of Rigid Link on the Right	Rotation at the Left End of Rigid Link on the Right
Ground	-0.000907	-0.000907	-0.000904	-0.000904
Mezzanine	-0.001311	-0.001311	-0.001311	-0.001311
1	-0.001535	-0.001535	-0.001537	-0.001537
2	-0.001654	-0.001654	-0.001655	-0.001655
3	-0.001703	-0.001703	-0.001704	-0.001704
4	-0.001705	-0.001705	-0.001705	-0.001705
5	-0.001670	-0.001670	-0.001669	-0.001669
6	-0.001609	-0.001609	-0.001607	-0.001607
7	-0.001529	-0.001529	-0.001526	-0.001526
8	-0.001444	-0.001444	-0.001440	-0.001440
9	-0.001379	-0.001379	-0.001374	-0.001374

From the table, it can be observed that the rotations are within acceptable limits to not violate the small rotation assumption. Therefore, there is no problem with using rigid links in the model.

### 5.6.4. Reliability Analysis by Using Pystra and ANN-CMCS

FORM, SORM, CMCS, and IS methods have been applied to the example given above before the application of ANN-CMCS coupling. The results obtained from FORM, SORM, CMCS, and IS are given in the table below:

Table 5.17. Results of FORM, SORM, IS, and CMCS analyses for the real-life 11-story RC frame structure example.

Method	Estimated Probability of Failure
FORM	0.002453
SORM (Breitung)	0.003082
SORM (Breitung HR)	0.003228
IS (23000 Samples with 0.01 CoV)	0.032423
CMCS (2.93E5 Samples with 0.01 CoV)	0.033048

The change in coefficient of variation (CoV) and estimated probability of failure with respect to the number of simulations for CMCS are given in Figure 5.83 and Figure 5.84.

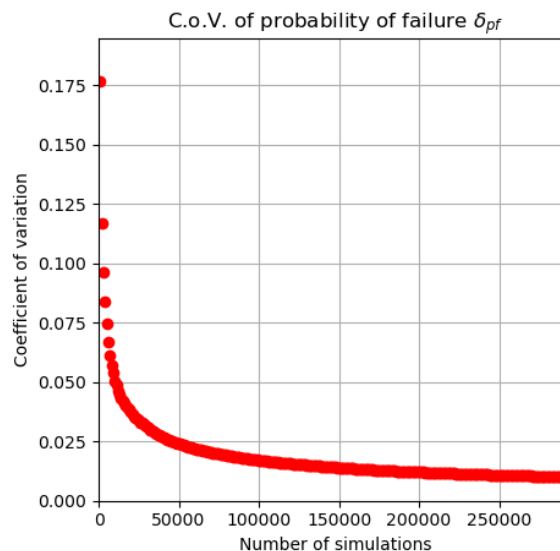


Figure 5.83. Change in the coefficient of variation with respect to the increasing number of simulations for the real-life 11-story RC frame structure example.

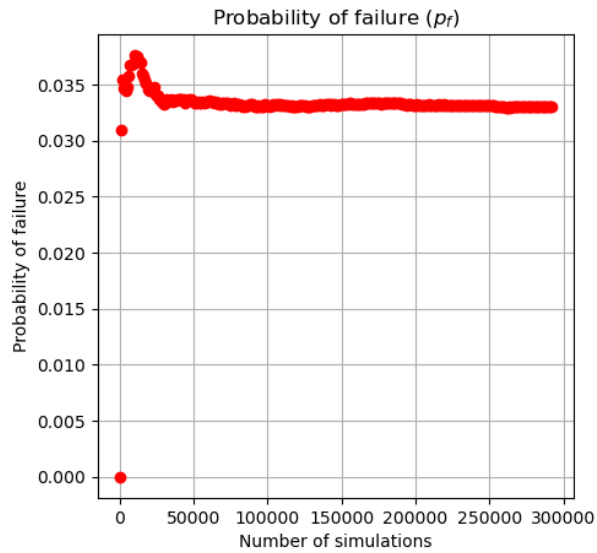


Figure 5.84. Change in the probability estimate with respect to the increasing number of simulations for the real-life 11-story RC frame structure example.

The performance metrics for each trained network are given in the figures below indicating that the trained networks for each dataset trained well.

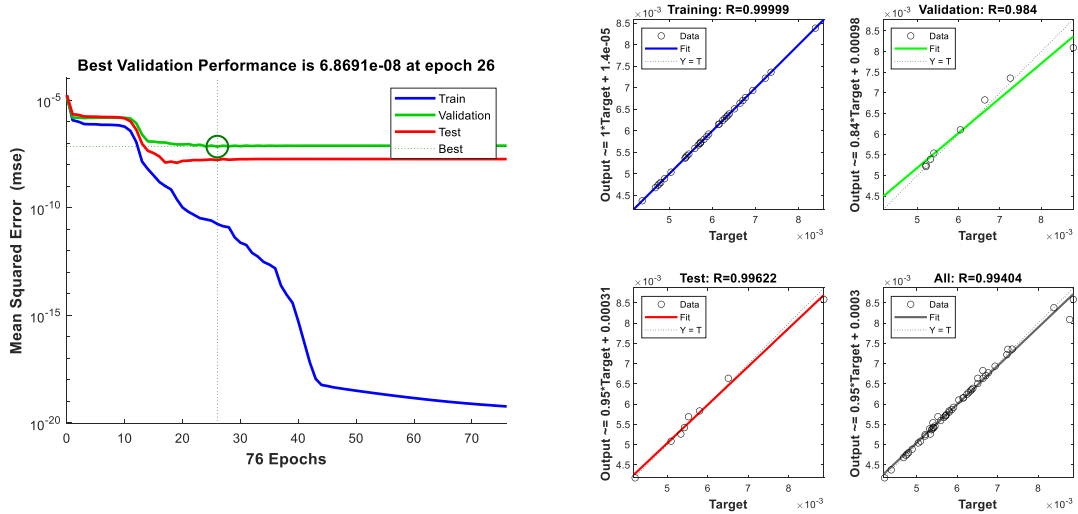


Figure 5.85. Performance metrics of the network that is trained by using 50 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

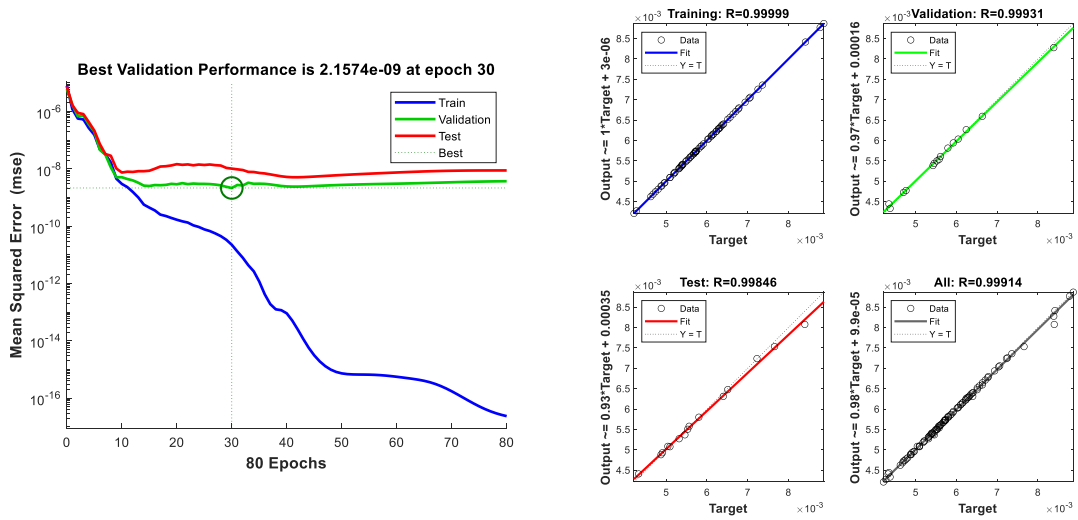


Figure 5.86. Performance metrics of the network that is trained by using 100 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

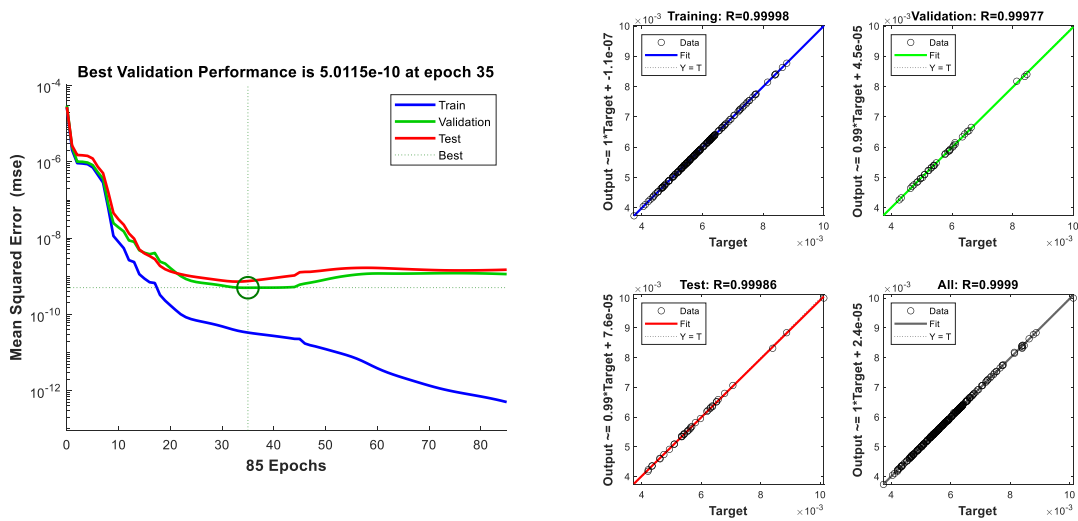


Figure 5.87. Performance metrics of the network that is trained by using 250 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).



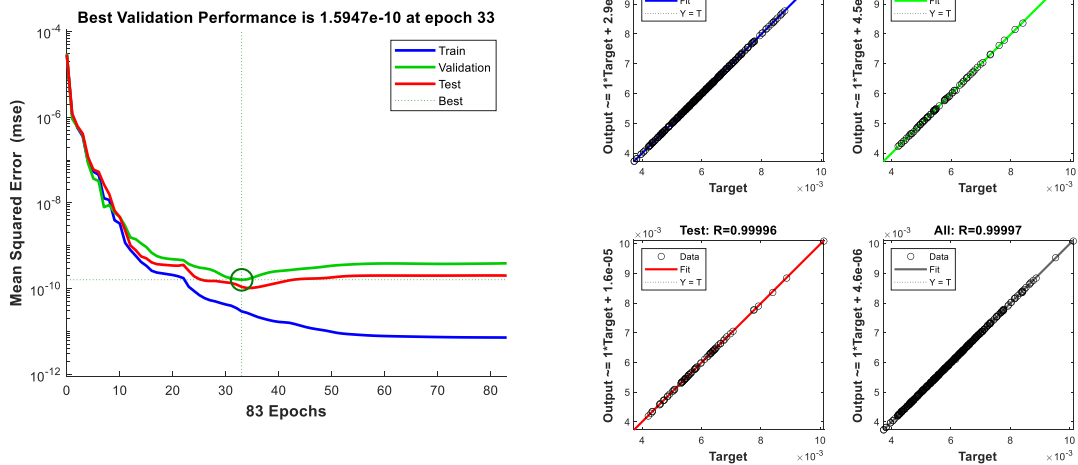


Figure 5.88. Performance metrics of the network that is trained by using 500 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

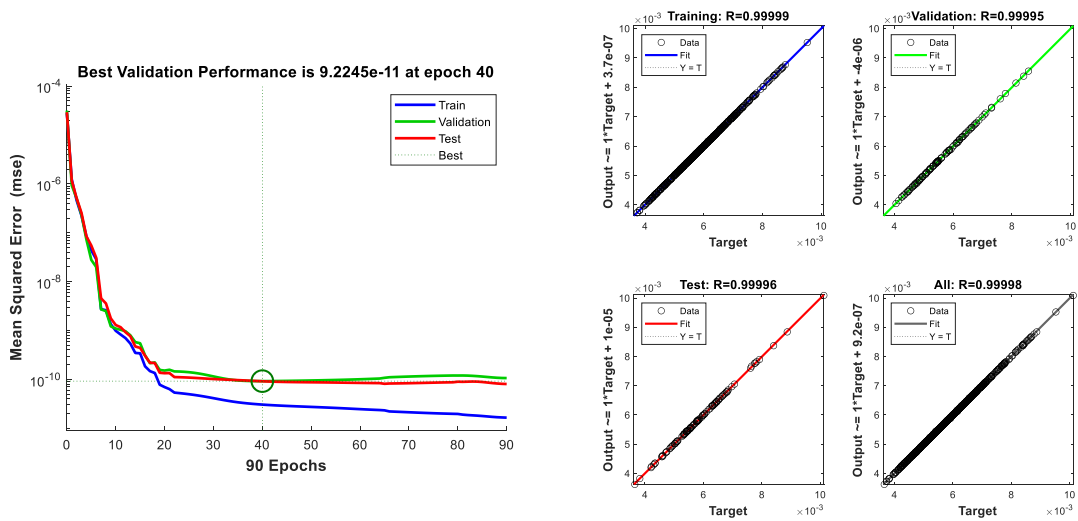


Figure 5.89. Performance metrics of the network that is trained by using 750 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

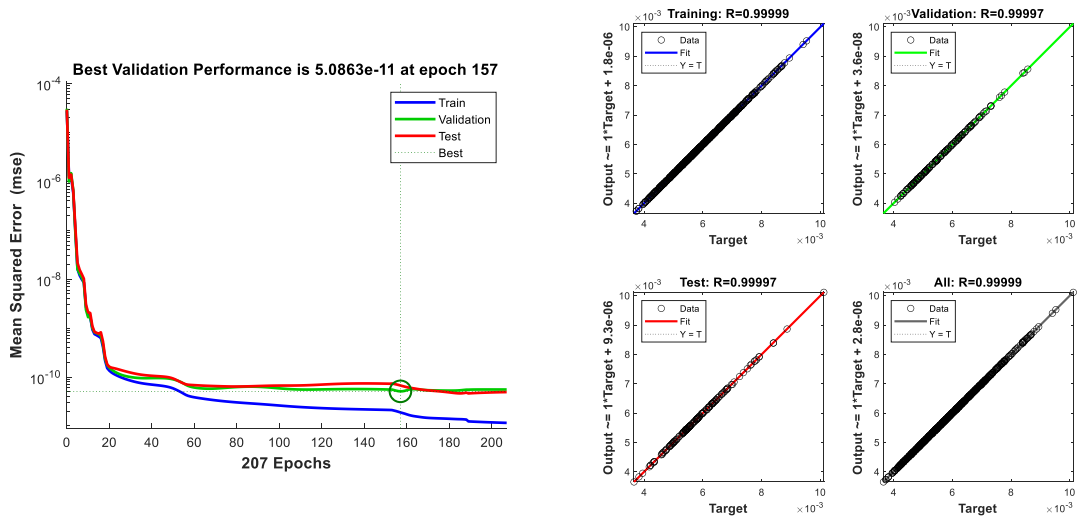


Figure 5.90. Performance metrics of the network that is trained by using 1000 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

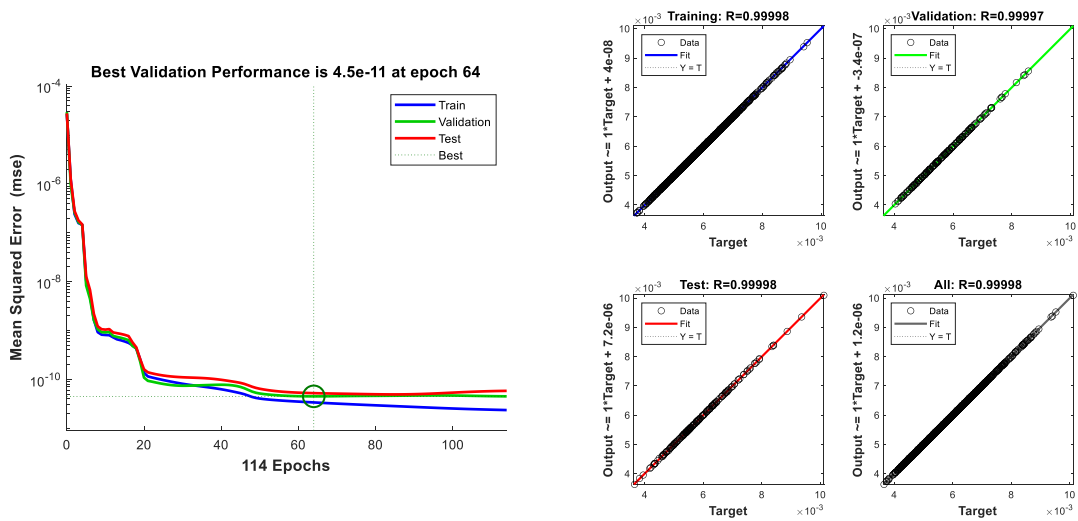


Figure 5.91. Performance metrics of the network that is trained by using 1250 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

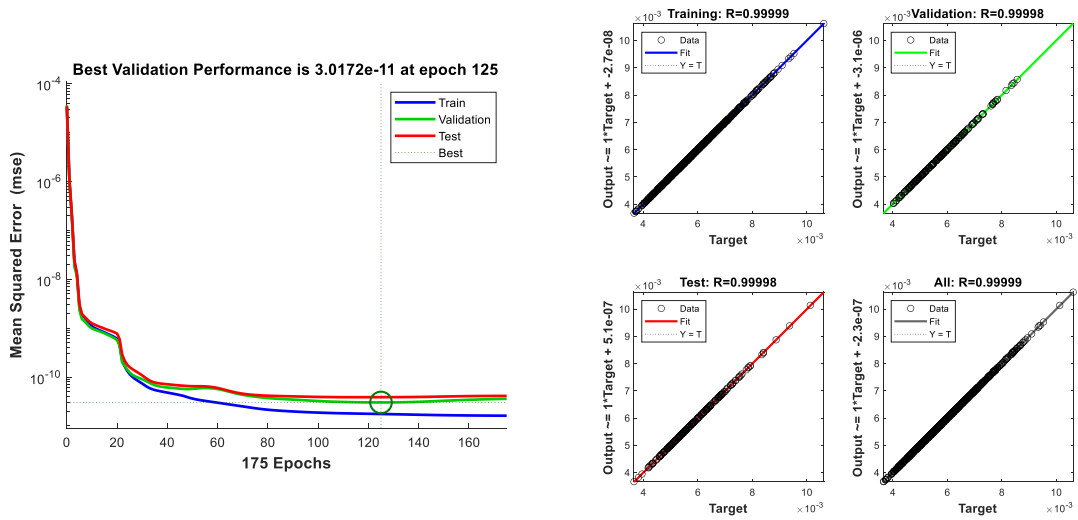


Figure 5.92. Performance metrics of the network that is trained by using 1500 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

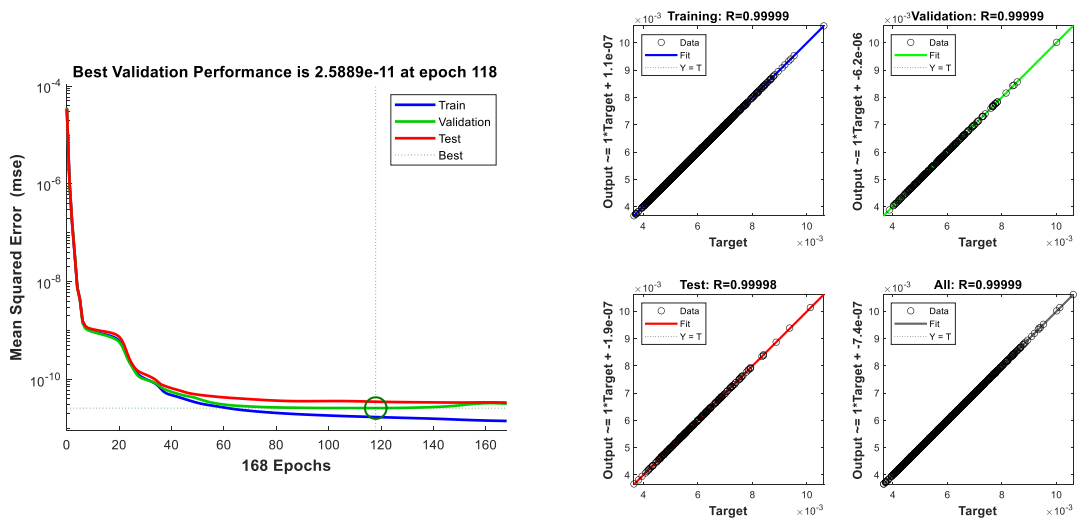


Figure 5.93. Performance metrics of the network that is trained by using 1750 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

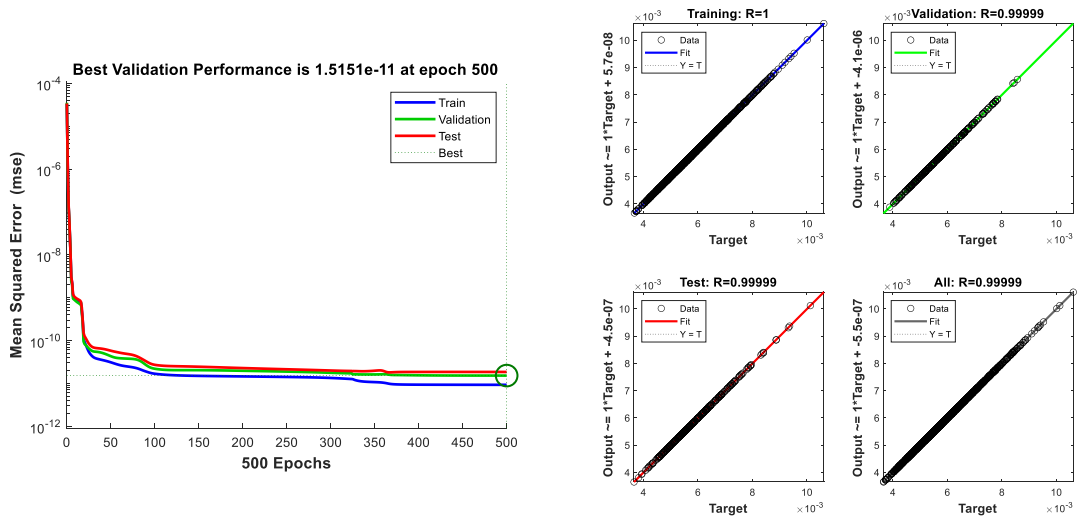


Figure 5.94. Performance metrics of the network that is trained by using 2000 samples for the real-life 11-story RC frame structure example. MSE performance (on the left side), regression plots (on the right side).

The probability estimates for the example with respect to the increasing number of training dataset size are given in Figure 5.95. It can be observed that the estimated probability of failure converges to the result obtained by using CMCS when the number of samples in the dataset increases.

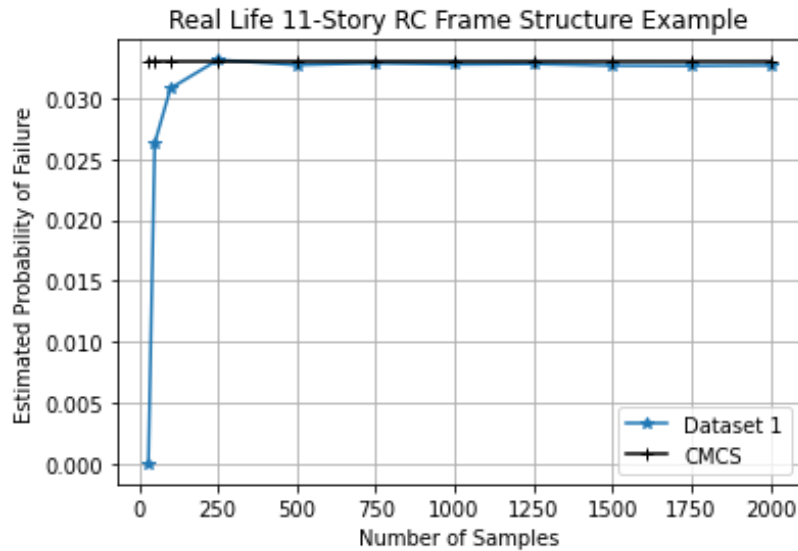


Figure 5.95. Change in probability estimate with respect to the increasing number of samples in the datasets used in the neural networks for the real-life 11-story RC frame structure example.

## CHAPTER 6

### DISCUSSION

This chapter aims to further investigate the results obtained for the numerical examples and introduce fruitful discussions based on the observations made from the examples.

The results of the numerical examples have shown that the estimated probability of failure by using ANN-CMCS coupling fluctuates if the number of samples in a dataset used in the training of the neural networks is increased. Depending on the number of samples in the dataset, the amount of fluctuation can be high or low. Therefore, the different estimates of the probability of failure in each number of samples cause the question of how many samples are needed in a dataset to obtain a probability of failure estimate with low fluctuation. It is basically a problem of convergence to a stable probability of failure estimation under the increasing number of samples in the training dataset. In this chapter, a novel convergence criterion will also be introduced to obtain a final estimate of the probability of failure among a set of probability of failure estimates.

An interesting observation can be made from the 5-story 3-bay correlated frame structure example. In that example, the probability of failure estimation starts to diverge from the stable estimate when the number of samples increases in the dataset. It is a noteworthy question whether this behavior still be observed if a different dataset is going to be used in the training of a neural network. In addition to this problem, it should also be investigated whether different datasets have any effect on the final probability of failure estimate. For these reasons, 4 additional different datasets - independent from each other and the one used in Chapter 5 - have been generated for each example and analyses for the probability of failure estimation are repeated.

Another important issue is to be pointed out that the performance of neural networks is satisfactory for each number of samples. However, even though the performances are well, the corresponding probability of failure estimate does not have to be satisfactory.

## 6.1. 3-Step Convergence Criterion

A general observation can be made from the results of the numerical examples that the existence of fluctuations in the probability of failure estimates for the different number of samples in the training dataset. If those fluctuations are investigated deeply, one can realize that the final probability of failure estimate based on the amount of fluctuation between the successive two steps can be misleading because a decision based on the successive two steps can lead to an overestimation or underestimation.

Alternatively, successive 3 steps can be used to determine the convergence of the estimation made for the probability of failure. If the probability of failure estimation for a specific number of samples is named as the current step,  $p_{f_i}$ , then, the one-step lower number of samples can be named the backward step and can be denoted by  $p_{f_{i-1}}$ . The same logic can be applied to the forward step. For the ideal case, all those 3 steps should be equal for a perfect convergence which means that the average of the probability of failures obtained from backward and forward steps is equal to the probability of failure corresponding to the current step. However, in practice, there is always a difference i.e. fluctuation between them. If the amount of fluctuation is represented by  $\chi$  then, a formulation can be derived for the convergence criterion as given below:

$$\frac{|p_{f_{i+1}} - 2p_{f_i} + p_{f_{i-1}}|}{2} = \chi \quad (6.1)$$

If the amount of fluctuation is defined by a percentage of the probability of failure estimate at the current step, then the equation given above can be rewritten as follows for convergence check:

$$\frac{|p_{f_{i+1}} - 2p_{f_i} + p_{f_{i-1}}|}{2} < \psi p_{f_i} \quad (6.2)$$

The term denoted by  $\psi$  in equation (6.2) can be considered as a coefficient for sensitivity. Generally, taking  $\psi$  between 0.005 and 0.02 provides satisfactory results. However, it should also be noted that the higher coefficient of sensitivity comes with a price of higher error in the probability of failure estimate.

Once the convergence criterion is satisfied for a number of samples in the dataset then, the final estimate for the probability of failure can be calculated by using equation (6.3).

$$\bar{p}_f = \frac{p_{f_{i+1}} + p_{f_i} + p_{f_{i-1}}}{3} \quad (6.3)$$

Based on the convergence criterion introduced in equation (6.2) and formulation for the final probability of failure estimate defined in equation (6.3), the final probability estimate for each example is given in the table below:

Table 6.1. Results the application of 3-step convergence criterion to the examples.

Example	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
Cantilever Beam	0.00949857	1.47	0.01	500
Simple Portal Frame	0.00227873	2.74	0.01	100
12-Story 3-Bay Frame Structure	0.0736438	3.71	0.01	250
5-Story 3-Bay Correlated Frame Structure	0.000247967	1.60	0.01	5000
Real-Life 11-Story RC Structure	0.0329086	0.42	0.01	750

From the table above, it can be stated that the proposed convergence criterion provides accurate estimates for the probability of failure with a lower number of samples compared to FORM, SORM, IS, and CMCS for the covered examples.

## 6.2. Effect of Different Datasets on the Probability of Failure Estimate

The behavior observed in the 5-story 3-bay correlated frame example needs further investigation of the effect of the datasets used in the training of neural networks. Additional 4 different datasets independent from each other and the one used in the examples have been generated for each example in this study to observe the effect of datasets on the probability of failure estimate.

The results of the analyses made for each example with 5 datasets are given in the following figures.

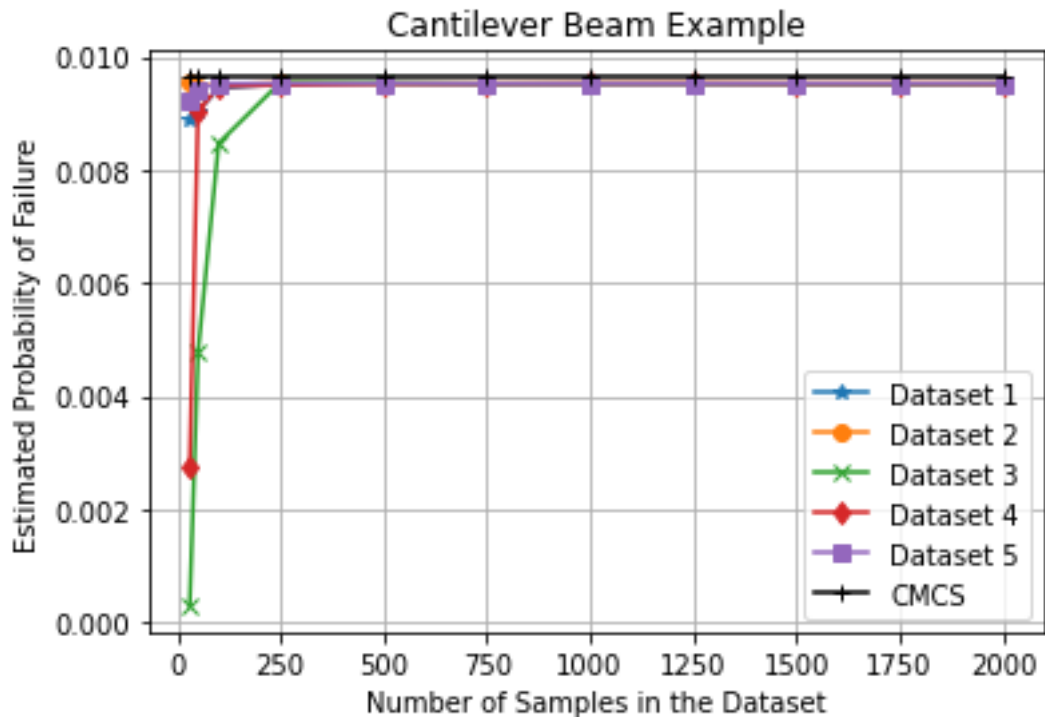


Figure 6.1. Effect of the dataset on the probability of failure estimate for the cantilever beam example.



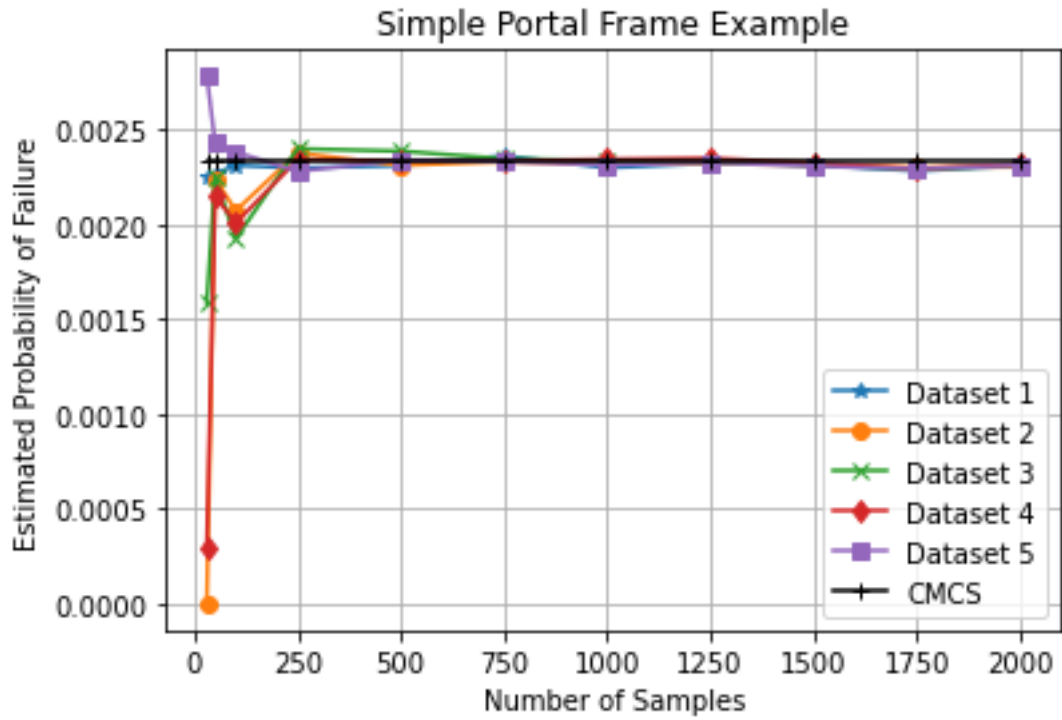


Figure 6.2. Effect of the dataset on the probability of failure estimate for the simple portal frame example.

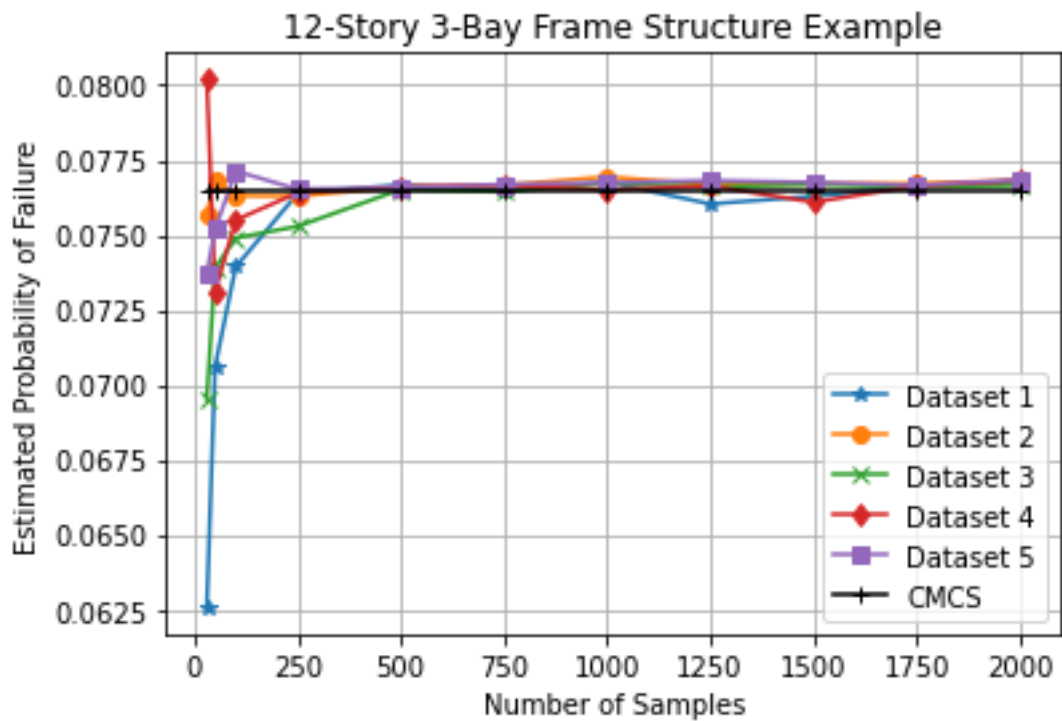


Figure 6.3. Effect of the dataset on the probability of failure estimate for the 12-story 3-bay frame structure example.

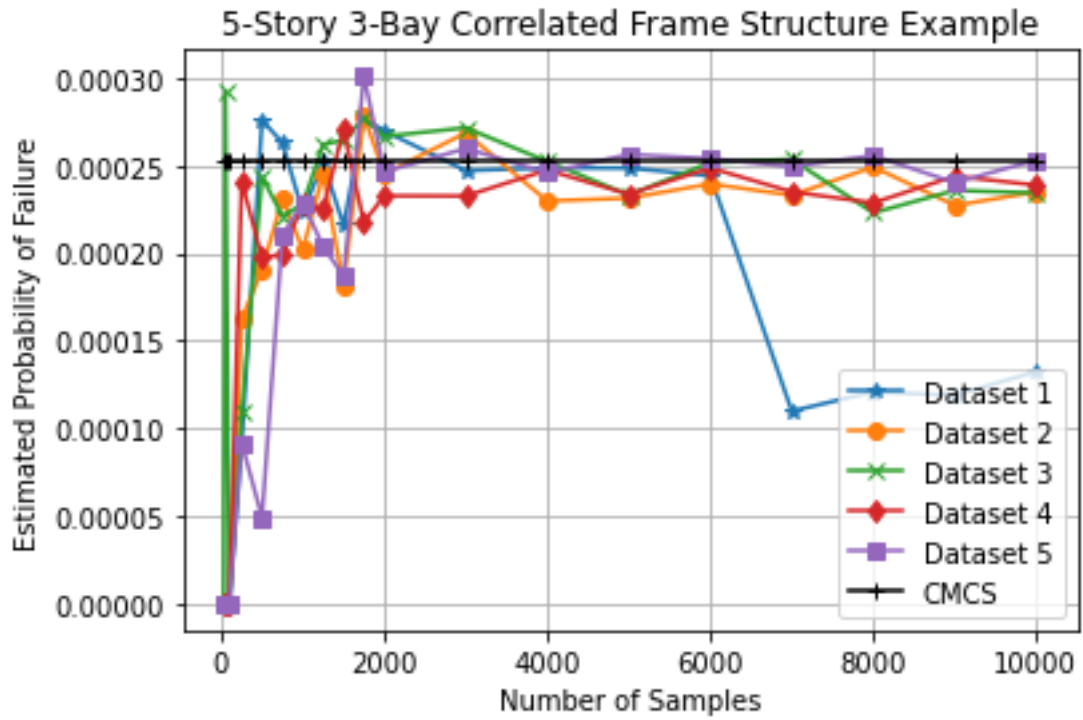


Figure 6.4. Effect of the dataset on the probability of failure estimate for the 5-story 3-bay frame structure example.

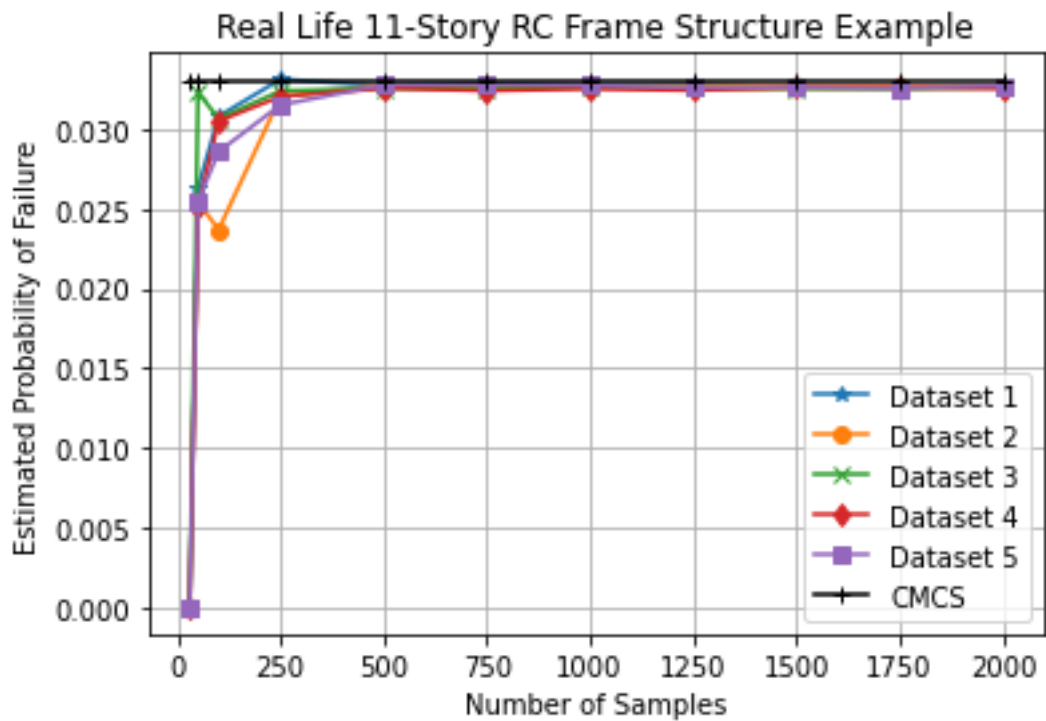


Figure 6.5. Effect of the dataset on the probability of failure estimate for the real life 11-story RC frame structure example.

The figures from Figure 6.1 to Figure 6.5 shows that the observation made for the 5-story 3-bay correlated frame structure is a unique case related to the quality of the dataset. It is possible that the added samples can be sampled from the locations of the region that is not interested in reliability analysis of the limit state function. The shape of the limit state function estimated by the neural network might be affected by those samples. Consequently, those samples might lead to divergence from the expected probability of failure.

Detailed information on the required number of samples for the convergence, probability of failure estimates, the coefficient of sensitivity, and the corresponding errors for each example and the dataset is provided in the tables below:

Table 6.2. Effect of different datasets on the cantilever beam example.

Dataset No.	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
1	0.00949857	1.47	0.01	500
2	0.00952280	1.22	0.01	100
3	0.00953810	1.06	0.01	750
4	0.00950003	1.45	0.01	500
5	0.00939387	2.55	0.01	100

Table 6.3. Effect of different datasets on the simple portal frame structure example.

Dataset No.	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
1	0.00227873	2.74	0.01	100
2	0.00232063	0.95	0.01	1000
3	0.00237833	-1.51	0.01	750
4	0.00233900	0.17	0.01	750
5	0.00232893	0.60	0.01	1000

Table 6.4. Effect of different datasets on the 12-story 3-bay frame structure example.

Dataset No.	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
1	0.07364380	3.71	0.01	250
2	0.07648150	0.00	0.01	250
3	0.07469640	2.33	0.01	250
4	0.07500890	1.92	0.01	250
5	0.07535070	1.48	0.01	100

Table 6.5. Effect of different datasets on the 5-story 3-bay correlated frame structure example.

Dataset No.	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
1	0.00024797	1.60	0.01	5000
2	0.00023343	7.37	0.02	6000
3	0.00025220	-0.08	0.01	5000
4	0.00023733	5.82	0.02	8000
5	0.00025310	-0.44	0.01	7000

Table 6.6. Effect of different datasets on the real-life 11-story RC frame structure example.

Dataset No.	Final Probability Estimate	Error Compared to CMCS Result (%)	$\psi$	Required Number of Samples for Convergence
1	0.0329086	0.42	0.01	750
2	0.0327076	1.03	0.01	1000
3	0.0324963	1.67	0.01	750
4	0.0324975	1.67	0.005	1000
5	0.0285349	13.66	0.005	250

Throughout the tables from Table 6.2 to Table 6.6, it can be observed that the final estimate for the probability of failure that is made by using the 3-step convergence criterion approach has been not affected much by the different datasets for all examples. However, the dataset used in the training of neural networks certainly has an effect on the required number of samples for convergence. Furthermore, in some cases, it was necessary to change the coefficient of sensitivity to provide convergence or increase the accuracy of the estimation. Even though those changes are dependent on the user's

preference, in general, the coefficient of sensitivity can be taken as 0.01 at the beginning of the process. If the convergence is not achieved, then it can be gradually increased to 0.02. In a similar manner, the coefficient of sensitivity can be decreased to 0.005 in order to improve the accuracy.

Another important point to be noted is that the required number of samples and convergence are also sensitive to the amount of difference between the number of samples for each step. For example, in this work, the number of samples is increased by 1000 samples per step after 2000 samples. However, this non-uniform fashion of increasing the number of samples is not mandatory. One can also increase the number of samples, for example, 100 per step. In such a scenario, if the probability of failure estimate at the current step satisfies the convergence criterion, the required number of samples will be equal to 2100 due to the need for evaluation of the forward step. In the case that is used in this study, the required number of samples will be equal to 3000 samples due to the 1000 sample increase per step. Even though 100 samples per step makes sense in terms of requiring less number of samples, it should also be kept in mind that one has to train much more neural networks because of the increase in the total number of steps until the convergence. Basically, this trade-off can be performed by considering the cost of evaluation of the limit state function at hand. If the evaluation of the limit state function is cheap, then a high amount of increase in the number of samples between steps can be more logical.

### **6.3. Performance of a Neural Network in the Training Phase and Probability of Failure Estimate Capability**

The performance metrics of each neural network for each example have a high value of the coefficient of determination for both train, validation, and test sets because of the stopping criterion defined based on the coefficient of regression in the training phase i.e. the neural networks that have a lower coefficient of regression than the threshold value are rejected in the training phase. The higher coefficient of determination for those sets indicates that the neural network has high prediction potential and is well-trained. On the other hand, however, even if the coefficient of determination for each set is high for a low number of samples, it is not a must that the estimation of the probability

of failure should be satisfactory. Therefore, a sole decision based on the coefficient of determination of the train, validation, and test sets can be deceiving. The main reason for this problem is that the coefficient of determination for those sets is calculated by using the limited data used in the training. It should also be noted that the training data includes the total data used in the train, validation, and test sets, therefore it is not only the number of samples in the train set. For that reason, the coefficient of determination of the test set is only an indicator of the prediction capability of neural networks for limited unseen data. Hence, the network might perform well for the limited unseen data but this does not guarantee that the network is able to predict all points in the limit state function.

#### **6.4. Comparison of ANN-CMCS Coupling with Other Methods**

It can be seen clearly from the results of the numerical examples and the tables provided in Sections 6.1 and 6.2 that ANN-CMCS coupling for structural reliability analysis provides accurate results with limited data. In some cases, the performance of SORM is superior to the other methods in terms of the computational cost and good level of accuracy. However, as stated in Section 3.1.6, it might converge in all cases and the level of accuracy depends on the shape of the limit state function which is not known for the implicit limit state functions priorly. On the other hand, the results of the simulation methods implemented in this study have shown that they require a large number of samples to yield a probability of failure estimate with a low coefficient of variation. At this point, ANN-CMCS coupling seems a promising method to balance convergence guarantee, accuracy, and low computational cost.

It should also be realized that each method can be superior with respect to the others depending on the dimension of the problem, the shape of the limit state function, or the magnitude of the probability of failure to be estimated. For example, FORM provides excellent accuracy and much lower computational cost with respect to the other methods if the limit state function at hand is linear. On the other hand, it starts to lose its accuracy when the limit state function is highly non-linear. For the limit state functions close to the second-order polynomials, SORM becomes superior in terms of accuracy and computational cost balance. However, neither FORM nor SORM does guarantee a

convergency for all types of limit state functions and they start to lose their accuracy if the basic assumptions made in their derivation are starts to not valid.

For non-linear and complex limit state functions the simulation methods can be preferable to reach a reliable probability of failure estimate. But their computational cost is high and IS might not be used if the prior FORM analysis does not converge to a design point. At this point, ANN-CMCS coupling becomes attractive because it is free from any prior analysis and it can perform well with the limited number of samples. Nevertheless, using ANN-CMCS coupling for reliability problems that have simple limit state functions – either linear or explicit – cannot be efficient due to the effort performed for the training of neural networks, and they also have a certain level of error inherently because of creating a surrogate for the limit state function at hand.

Consequently, one should decide which method is more appropriate than the others based on the problem at hand. The following flowchart is developed based on the observations made from the examples introduced in this study to help the one who will perform structural reliability analysis. If the limit state function is not explicit, then the flowchart might be tracked from the decision point of “Is dataset limited?” question.

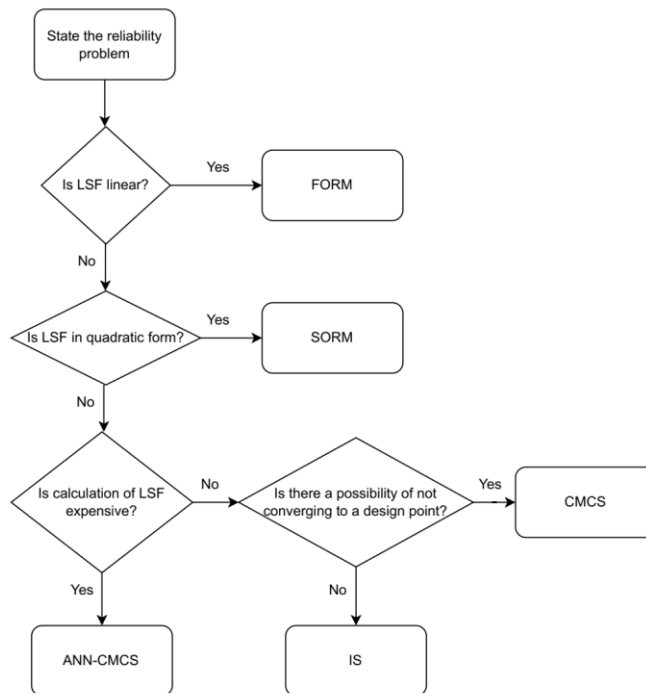


Figure 6.6. Flowchart for selecting appropriate method under problem-specific conditions.



## 6.5. Comparison of the Results with the Results of Previous Studies

In this subsection, a comparison between the obtained results from ANN-CMCS coupling for the benchmark examples and the results obtained from the selected previous studies that include the related problem will be performed. This subsection is essential because of providing a check of the consistency between the results and identifying the inconsistencies if there exist. Even though it is not possible to compare their computational efficiency because the number of samples used to find the result is not available in each study, it is possible to compare them in terms of accuracy.

### 6.5.1. Comparison of the Results for the Cantilever Beam Example

The cantilever beam example implemented in this study was also studied by Beheshti Nezhad, Miri, and Ghasemi (2019b), Rajashekhar and Ellingwood (1993), Cheng and Li (2008), and Ren and Bai (2011). The corresponding results obtained from the studies and the results obtained in this study are given in Figure 6.7. Based on the comparison of the results, it can be observed that ANN-CMCS coupling provides satisfactory results for the problem at hand.

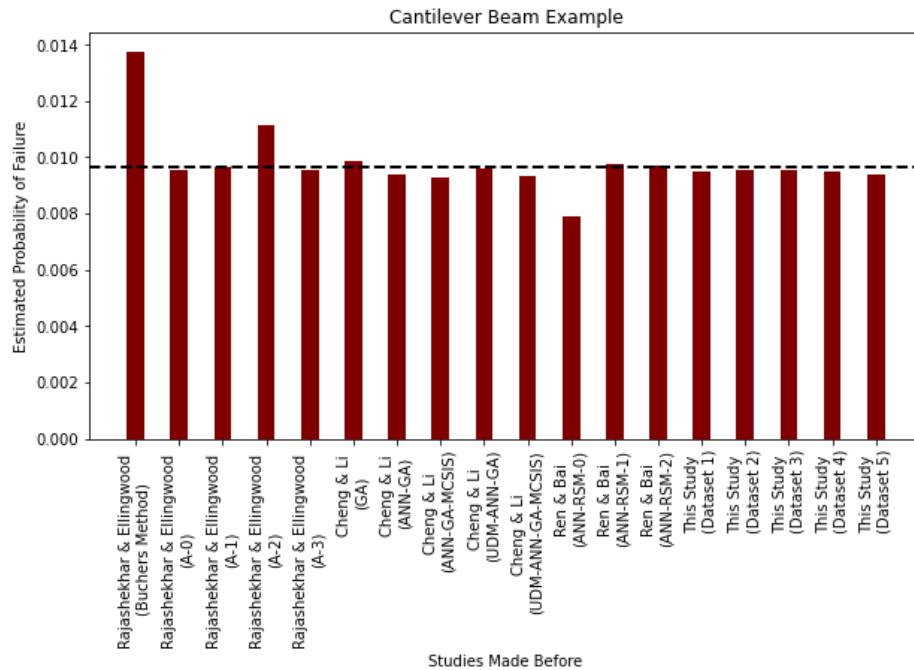


Figure 6.7. Comparison of different results obtained for the cantilever beam example.

### 6.5.2. Comparison of the Results for the Simple Portal Frame Example

The simple portal frame structure example is provided in the study of Deng et al. (2005). In the study, there are two results presented for the problem. The first one is the result of an ANN-CMCS coupling like in this study. The second result is based on the probability of failure estimate obtained by using the modified response surface approach without using cross terms. The results obtained from this study and Deng et al. (2005) are presented in Figure 6.8. From the figure, it can be said that the results agree well.

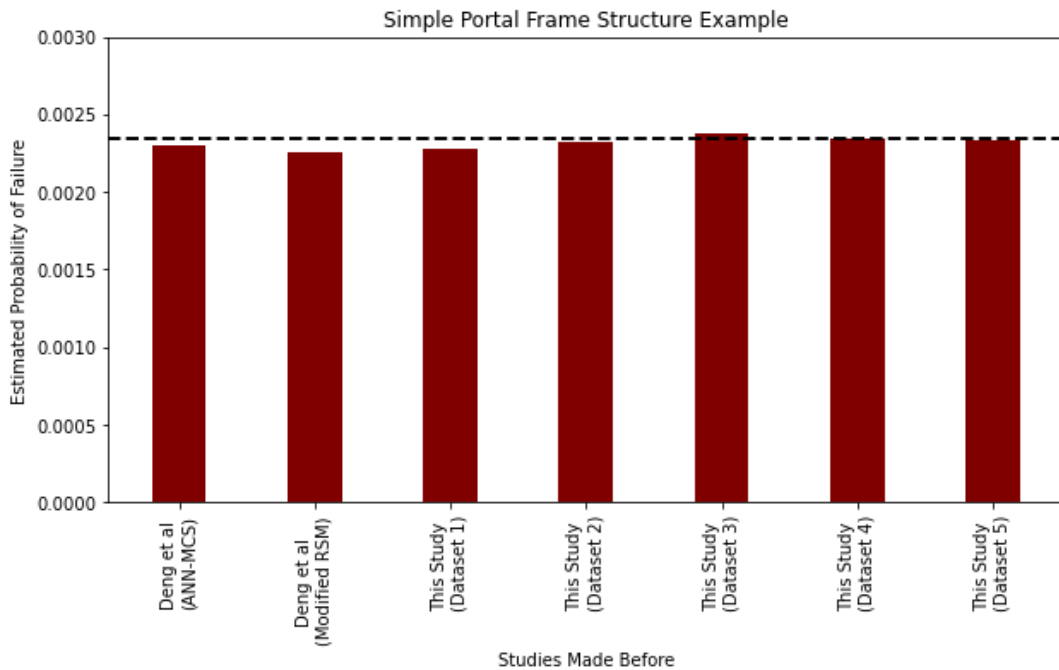


Figure 6.8. Comparison of different results obtained for the simple portal frame structure example.

### 6.5.3. Comparison of the Results for the 12-Story 3-Bay Frame Structure Example

Cheng and Xiao (2005), Cheng (2007), and Beheshti Nezhad, Miri, and Ghasemi (2019a) have been studied on this problem in their studies. The problem configuration, definition of the limit state function, and the statistical distributions and properties of the random variables are the same in all those three studies. In this study, the problem is compatible with those above-mentioned studies. The comparison of the results is

provided Figure 6.9. Based on the figure, the results of ANN-CMCS coupling provide satisfactory results in terms of accuracy.

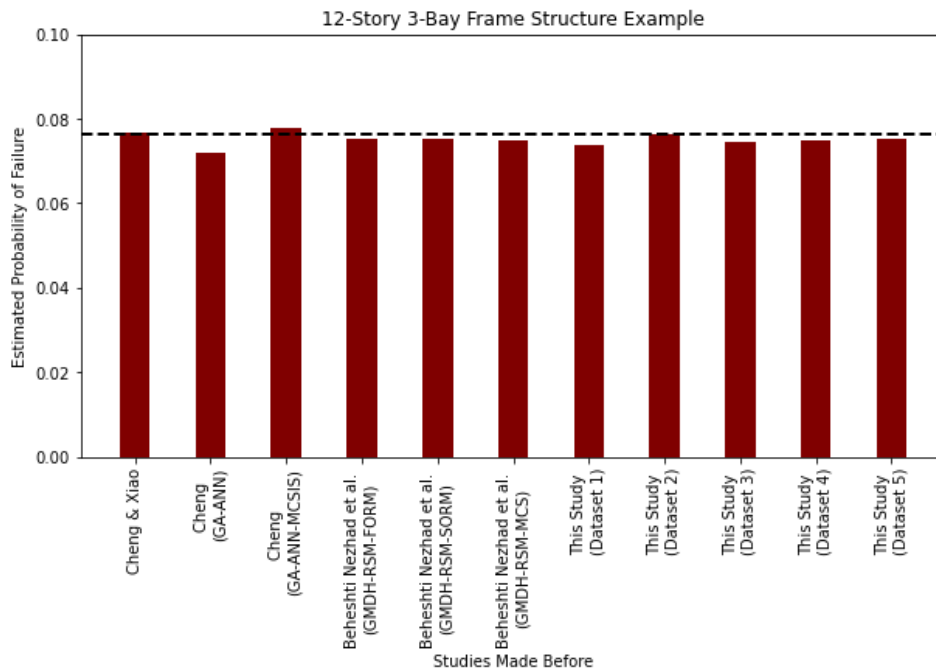


Figure 6.9. Comparison of different results obtained for the 12-story 3-bay frame structure example.

#### 6.5.4. Comparison of the Results for the 5-Story 3-Bay Correlated Frame Structure Example

The 5-story 3-bay correlated frame structure example has been investigated in several studies before as mentioned in Section 5.5. The results obtained from those studies and the results obtained from this study are provided in Figure 6.10. In the figure, the dashed line represents the result obtained from CMCS in this study. As can be shown from the figure, there is a considerable amount of difference between the results of the studies. The main reason for those differences is that the example has differences in random variables or differences in the coefficient of correlation in each mentioned study. The differences between studies related to the coefficient of correlation are given in Table 6.7 whereas the differences related to the type of random variables and threshold value used in the limit state function are given in Table 6.8.

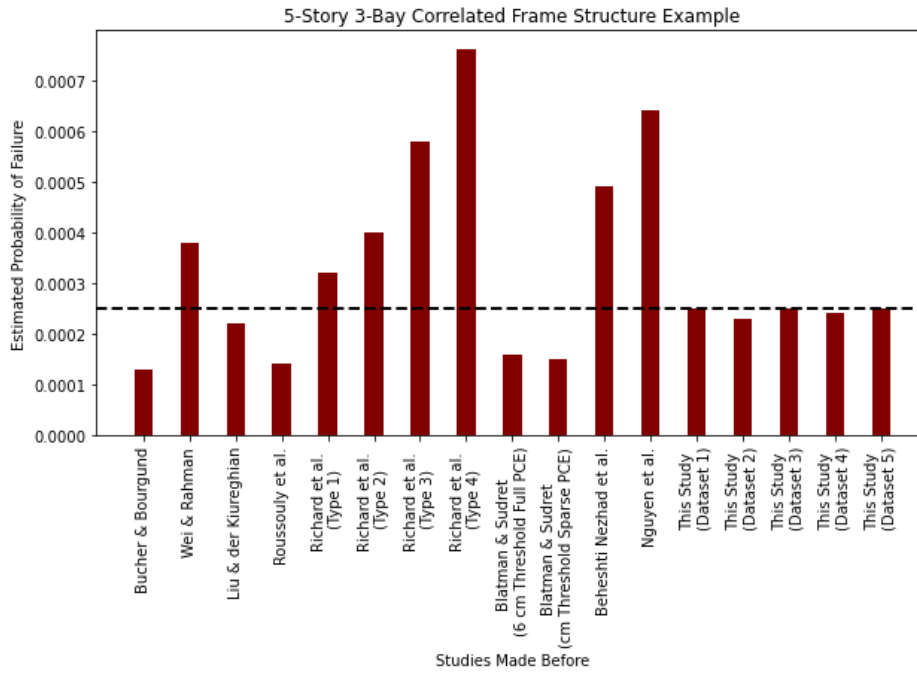


Figure 6.10. Comparison of different results obtained for the 5-story 3-bay correlated frame structure example.

Table 6.7. Differences in studies in terms of coefficient of correlation.

Authors	$\rho_{A_i A_j}$	$\rho_{I_i I_j}$	$\rho_{A_i I_j}$	$\rho_{F_{i,j}}$	$\rho_{E_{i,j}}$	$\rho_{A_i I_i}$
Beheshti Nezhad, Miri, and Ghasemi (2019b)	0.13	0.13	0.13	0.95	0.90	0.00
X. S. Nguyen et al. (2009)	0.13	0.13	0.13	0.95	0.90	0.00
Blatman and Sudret (2010)	0.13	0.13	0.13	0.00	0.90	0.95
Richard, Cremona, and Adelaide (2012)	0.13	0.00	0.00	0.95	0.90	0.00
Roussouly, Petitjean, and Salaun (2013)	0.13	0.13	0.13	0.00	0.90	0.95
Wei and Rahman (2007)	0.13	0.13	0.13	0.00	0.90	0.95
Liu and Der Kiureghian (1986)	0.13	0.13	0.13	0.50	0.90	0.95
Bucher and Bourgund (1990b)	0.13	0.13	0.13	0.90	0.90	0.00
This study	0.13	0.13	0.13	0.90	0.90	0.95

Table 6.7 and Table 6.8 show that even though the configurations are the same, the definition of the problem is different in almost all studies presented herein. Therefore, their results cannot be compared directly.

Table 6.8. Differences in studies in terms of random variables and limit state function.

Authors	Type of Loads	Type of Other Random Variables	Threshold of Limit State Function
Beheshti Nezhad, Miri, and Ghasemi (2019b)	Gumbel Max	Normal	0.061
X. S. Nguyen et al. (2009)	Gumbel Max	Normal	0.061
Blatman and Sudret (2010)	Lognormal	Truncated Normal	0.060
Richard, Cremona, and Adelaide (2012)	Gumbel Max	Normal	0.061
Roussouly, Petitjean, and Salaun (2013)	Lognormal	Normal	0.060
Wei and Rahman (2007)	Lognormal	Normal	0.06096
Liu and Der Kiureghian (1986)	Rayleigh	Normal	0.06096
Bucher and Bourgund (1990b)	Rayleigh	Normal	0.061
This study	Lognormal	Truncated Normal	0.060

## 6.6. Adaptive Algorithm for ANN-CMCS Coupling

The flowchart given in Figure 5.1 can be extended to obtain an adaptive algorithm as given in Figure 6.11. In the updated flowchart, a portion along the length of the random variables is used in the first step to obtaining a probability estimate. If the convergence criterion is not satisfied then, new samples are added and a new artificial neural network is trained for the extended dataset. This procedure continues until the convergence criterion given in equation (6.2) is satisfied. This adaptive scheme provides efficiency in

terms of computational cost by avoiding the evaluation of all random variables along their total length.

Even though the number of new samples added after the first step is indicated as “ $m$ ” in the flowchart, the number of “ $m$ ” can be changed in each step i.e. it does not have to be equal to a constant number. This flexibility in the addition of new samples can be used to provide fine-tuning when the probability estimate reaches a plateau.

The determination of the optimum number of new samples added at each step and the detailed application of the developed adaptive algorithm are open problems for future works.

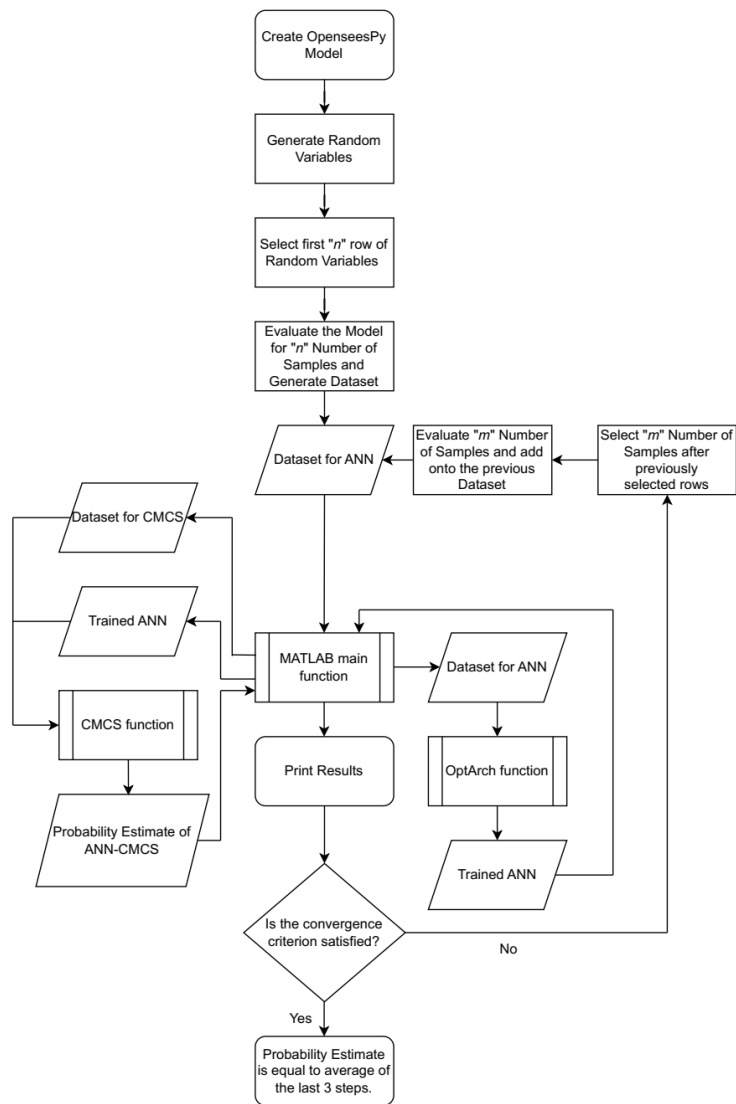


Figure 6.11. Adaptive algorithm for ANN-CMCS coupling.

## CHAPTER 7

### CONCLUSION

In this study, the theoretical background of the most widely known structural reliability methods and artificial neural networks is presented. Then, artificial neural networks have been coupled with CMCS and applied to the 4 well-known benchmark examples and one novel example derived from a real-life RC structure to reduce the computational effort made in analysis by providing a certain level of accuracy. Bayesian Regularization has been implemented in the training phase of the neural networks in order to increase the performance of the trained network. The obtained results are also compared with the commonly used structural reliability methods such as FORM, SORM, CMCS, and IS to investigate the efficiency of the introduced methodology. The results obtained for well-known benchmark examples are also compared with the selected studies from the literature in terms of accuracy. Additional effort has also been made to the important points inherent to artificial neural networks. The critical findings from this study are listed below:

- A 3-step convergence criterion is developed in this study in order to stop the training of new neural networks with a larger dataset when the probability estimate is stabilized. A coefficient of sensitivity is also proposed to provide both high accuracy and convergence guarantee to the method.
- Detailed analysis of results has revealed that the efficiency of the introduced methodology is dependent on the dataset used in the neural network. Even though the dataset affects the efficiency of the method, i.e. the size of the dataset – a larger dataset used in the training means a decrease in the efficiency due to the increase in the limit state function evaluation - the accuracy of the method still yields satisfactory results.
- A distinction is also pointed out between the performance of the trained neural network and the performance of the method. This distinction basically states that a neural network trained by a small dataset and having a high coefficient of determination does not always represent the performance of the method.

- Based on the observations made from the numerical examples and related discussions, an adaptive algorithm for ANN-CMCS coupling is suggested to reduce the computational cost further while providing a certain level of accuracy.
- The coupling of artificial neural networks and CMCS provides satisfactory results in most cases in both accuracy and efficiency based on the comparison made with the other methods and selected studies from the literature. However, FORM is a more appropriate method in terms of efficiency and accuracy when the limit state function at hand is linear. In a similar manner, SORM outperforms the other methods when the limit state function at hand is a second-order polynomial. Among all of the methods, CMCS is the most robust and reliable method when its computational cost is not prohibitive. The alternative, IS, can be used in order to decrease the computational cost of CMCS, however, it requires a design point to detect the most critical sampling region in the problem space. The main advantage of ANN-CMCS coupling does not require a design point and provides high accuracy with relatively low computational cost. Therefore, it is a preferable method when the limit state function is implicit and the dataset at hand is limited i.e. evaluating the limit state function is computationally costly.



## REFERENCES

- Anaconda Inc. 2020. “Anaconda Software Distribution.” Anaconda Inc.  
<https://www.anaconda.com/>.
- Au, S K, and J L Beck. 1999. “A New Adaptive Importance Sampling Scheme for Reliability Calculations.” *Structural Safety* 21: 135–58.  
[www.elsevier.nl/locate/strusafe](http://www.elsevier.nl/locate/strusafe).
- Beheshti Nezhad, Hossein, Mahmoud Miri, and Mohammad Reza Ghasemi. 2019a. “New Neural Network-Based Response Surface Method for Reliability Analysis of Structures.” *Neural Computing and Applications* 31 (3): 777–91.  
<https://doi.org/10.1007/s00521-017-3109-2>.
- Beheshti Nezhad, Hossein, Mahmoud Miri, and Mohammad Reza Ghasemi. 2019b. “New Neural Network-Based Response Surface Method for Reliability Analysis of Structures.” *Neural Computing and Applications* 31 (3): 777–91.  
<https://doi.org/10.1007/s00521-017-3109-2>.
- Blatman, Géraud, and Bruno Sudret. 2010. “An Adaptive Algorithm to Build up Sparse Polynomial Chaos Expansions for Stochastic Finite Element Analysis.” *Probabilistic Engineering Mechanics* 25 (2): 183–97.  
<https://doi.org/10.1016/j.probengmech.2009.10.003>.
- Breitung, Karl. 1984. “Asymptotic Approximations for Multinormal Integrals.” *Journal of Engineering Mechanics* 110 (3): 357–66.
- Breitung, Karl. 2021. “SORM, Design Points, Subset Simulation, and Markov Chain Monte Carlo.” *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering* 7 (4). <https://doi.org/10.1061/ajrua6.0001166>.
- Bucher, C G, and U Bourgund. 1990a. “A Fast and Efficient Response Surface Approach for Structural Reliability Problems.” *Structural Safety* 7: 57–66.
- Bucher, C G, and U Bourgund. 1990b. “A Fast and Efficient Response Surface Approach for Structural Reliability Problems.” *Structural Safety* 7: 57–66.

- Cardoso, João B., João R. de Almeida, José M. Dias, and Pedro G. Coelho. 2008. "Structural Reliability Analysis Using Monte Carlo Simulation and Neural Networks." *Advances in Engineering Software* 39 (6): 505–13.  
<https://doi.org/10.1016/j.advengsoft.2007.03.015>.
- Cheng, Jin. 2007. "Hybrid Genetic Algorithms for Structural Reliability Analysis." *Computers and Structures* 85 (19–20): 1524–33.  
<https://doi.org/10.1016/j.compstruc.2007.01.018>.
- Cheng, Jin, and Q. S. Li. 2008. "Reliability Analysis of Structures Using Artificial Neural Network Based Genetic Algorithms." *Computer Methods in Applied Mechanics and Engineering* 197 (45–48): 3742–50.  
<https://doi.org/10.1016/j.cma.2008.02.026>.
- Cheng, Jin, and Ru Cheng Xiao. 2005. "Serviceability Reliability Analysis of Cable-Stayed Bridges." *Structural Engineering and Mechanics*. Techno-Press.  
<https://doi.org/10.12989/sem.2005.20.6.609>.
- Choi, Seung-Kyum., R. v. Grandhi, and Robert A. Canfield. 2007. *Reliability-Based Structural Design*. Springer.
- Chojaczyk, A A, A P Teixeira, L C Neves, J B Cardoso, and C Guedes Soares. 2015. "Review and Application of Artificial Neural Networks Models in Reliability Analysis of Steel Structures." *Structural Safety* 52: 78–89.  
<https://doi.org/https://doi.org/10.1016/j.strusafe.2014.09.002>.
- Cornell, C Allin. 1969. "A Probability-Based Structural Code." *ACI Journal Proceedings* 66 (12). <https://doi.org/10.14359/7446>.
- Deák, I. 1980. "Three Digit Accurate Multiple Normal Probabilities." *Numerische Mathematik* 35: 369–80.
- Deng, Jian, Desheng Gu, Xibing Li, and Zhong Qi Yue. 2005. "Structural Reliability Analysis for Implicit Performance Functions Using Artificial Neural Network." *Structural Safety* 27 (1): 25–48. <https://doi.org/10.1016/j.strusafe.2004.03.004>.
- Ditlevsen, O, R.E. Melchers, and H Gluwer. 1990. "General Multi-Dimensional Probability Integration By Directional Simulation." *Computers & Structures* 36 (2): 355–68.

- Ditlevsen, O, R Olesen, and G Mohr. 1987. "Solution of a Class Of Load Combination Problems by Directional Simulation." *Structural Safety* 4: 95–109.
- Dudzik, Agnieszka, and Beata Potrzyszcz-Sut. 2019. "The Structural Reliability Analysis Using Explicit Neural State Functions." *MATEC Web of Conferences* 262: 10002. <https://doi.org/10.1051/matecconf/201926210002>.
- Faravelli, Lucia. 1989. "Response Surface Approach for Reliability Analysis." *Journal of Engineering Mechanics* 115 (12): 2763–81.
- Fiessler, Bernd, Hans-Joachim Neumann, and Rüdiger Rackwitz. 1979. "Quadratic Limit States in Structural Mechanics." *Journal of the Engineering Mechanics Division* 105 (EM4): 661–76.
- Foresee, F Dan, and M T Hagan. 1997. "Gauss-Newton Approximation to Bayesian Learning." In *Proceedings of International Conference on Neural Networks (ICNN'97)*, 3:1930–35 vol.3. <https://doi.org/10.1109/ICNN.1997.614194>.
- Goh, Anthony T.C., and Fred H. Kulhawy. 2003. "Neural Network Approach to Model the Limit State Surface for Reliability Analysis." *Canadian Geotechnical Journal* 40 (6): 1235–44. <https://doi.org/10.1139/t03-056>.
- Gomes, Herbert Martins, and Armando Miguel Awruch. 2004. "Comparison of Response Surface and Neural Network with Other Methods for Structural Reliability Analysis." *Structural Safety* 26 (1): 49–67. [https://doi.org/10.1016/S0167-4730\(03\)00022-5](https://doi.org/10.1016/S0167-4730(03)00022-5).
- Gomes, Herbert Martins, and Armando Miguel Awruch. 2005. "Reliability Analysis of Concrete Structures with Neural Networks and Response Surfaces." *Engineering Computations (Swansea, Wales)* 22 (1): 110–28. <https://doi.org/10.1108/02644400510572433>.
- Gomes, Wellison José De Santana. 2020. "Shallow and Deep Artificial Neural Networks for Structural Reliability Analysis." *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering* 6 (4). <https://doi.org/10.1115/1.4047636>.

- Guan, X L, and R E Melchers. 2001. "Effect of Response Surface Parameter Variation on Structural Reliability Estimates." *Structural Safety* 23 (4): 429–44. [www.elsevier.com/locate/strusafe](http://www.elsevier.com/locate/strusafe).
- Hackl, Jürgen, and Colin Caprani. 2022. "Pystra." <http://pystra.github.io/pystra/index.html>.
- Hagan, Martin T., and Mohammad B. Menhaj. 1994. "Training Feedforward Networks with the Marquardt Algorithm." *IEEE Transactions on Neural Networks* 5 (6): 989–93. <https://doi.org/10.1109/72.329697>.
- Harbitz, A. 1983. "Efficient and Accurate Probability of Failure Calculation by the Use of the Importance Sampling Technique." In *Proc. Fourth. Int. Conf. on Applications of Statistics and Probability in Soil and Structural Engineering*, 825–36. Bologna, Italy.
- Hasofer, A. M., and Niels Lind. 1974. "An Exact and Invariant First Order Reliability Format." *Journal of Engineering Mechanics*. <https://www.researchgate.net/publication/243758427>.
- Hastings, W K. 1970. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika* 57 (1): 97–109. <http://www.jstor.org>URL:<http://www.jstor.org/stable/2334940>.
- Hornik, K.M., M. Stinchcombe, and H. White. 1989. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks* 2: 359–66.
- Hosni Elhewy, A., E. Mesbahi, and Y. Pu. 2006. "Reliability Analysis of Structures Using Neural Network Method." *Probabilistic Engineering Mechanics* 21 (1): 44–53. <https://doi.org/10.1016/j.probengmech.2005.07.002>.
- Hu, Zhangli, Rami Mansour, Mårten Olsson, and Xiaoping Du. 2021. "Second-Order Reliability Methods: A Review and Comparative Study." *Structural and Multidisciplinary Optimization* 64 (6): 3233–63. <https://doi.org/10.1007/s00158-021-03013-y>.
- Hurtado, Jorge E, and Diego A Alvarez. 2001. "Neural-Network-Based Reliability Analysis: A Comparative Study." *Computer Methods in Applied Mechanics and Engineering* 191: 113–32. [www.elsevier.com/locate/cma](http://www.elsevier.com/locate/cma).

- Jain, A K, Jianchang Mao, and K M Mohiuddin. 1996. "Artificial Neural Networks: A Tutorial." *Computer* 29 (3): 31–44. <https://doi.org/10.1109/2.485891>.
- Kiureghian, Armen der, M Asce, Hong-Zong Lin, and Shyh-Jiann Hwang. 1987. "Second-Order Reliability Approximations." *Journal of Engineering Mechanics* 113 (8): 1208–25.
- Kiureghian, Armen der, and Taleen Dakessian. 1998. "Multiple Design Points in First and Second-Order Reliability." *Structural Safety* 20: 37–49.
- Koduru, S. D., and T. Haukaas. 2010. "Feasibility of FORM in Finite Element Reliability Analysis." *Structural Safety* 32 (2): 145–53. <https://doi.org/10.1016/j.strusafe.2009.10.001>.
- Köylüoğlu, Hasan Uğur, and Søren R K Nielsen. 1994. "New Approximations for SORM Integrals." *Structural Safety* 13 (4): 235–46. [https://doi.org/https://doi.org/10.1016/0167-4730\(94\)90031-0](https://doi.org/https://doi.org/10.1016/0167-4730(94)90031-0).
- Kroetz, Henrique M., Rodolfo K. Tessari, and André T. Beck. 2017. "Performance of Global Metamodeling Techniques in Solution of Structural Reliability Problems." *Advances in Engineering Software* 114 (December): 394–404. <https://doi.org/10.1016/j.advengsoft.2017.08.001>.
- Lemaire, Maurice, Alaa Chateauneuf, and Jean-Claude Mitteau. 2009. *Structural Reliability*.
- Levenberg, Kenneth. 1944. "A Method for the Solution of Certain Non-Linear Problems in Least Squares." *Quarterly of Applied Mathematics* 2 (2): 164–68. <http://www.jstor.org/stable/43633451>.
- Li, Hong Shuang, and Zi Jun Cao. 2016. "Matlab Codes of Subset Simulation for Reliability Analysis and Structural Optimization." *Structural and Multidisciplinary Optimization* 54 (2): 391–410. <https://doi.org/10.1007/s00158-016-1414-5>.
- Li, Mingyang, and Zequn Wang. 2020. "Deep Learning for High-Dimensional Reliability Analysis." *Mechanical Systems and Signal Processing* 139 (May). <https://doi.org/10.1016/j.ymsp.2019.106399>.
- Lieu, Qui X., Khoa T. Nguyen, Khanh D. Dang, Seunghye Lee, Joowon Kang, and Jaehong Lee. 2022. "An Adaptive Surrogate Model to Structural Reliability

- Analysis Using Deep Neural Network.” *Expert Systems with Applications* 189 (March). <https://doi.org/10.1016/j.eswa.2021.116104>.
- Liu, Pei-Ling, and Armen Der Kiureghian. 1986. “Optimization Algorithms for Structural Reliability Analysis.” *UCB/SESM-86/09*. California.
- Liu, Pei-Ling, and Armen Der Kiureghian. 1991. “Optimization Algorithms for Structural Reliability.” *Structural Safety* 9: 161–77.
- Mackay, David J C. 1992. “Bayesian Interpolation.” *Neural Computation* 4 (3): 415–47.
- Marquardt, Donald W. 1963. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters.” *Journal of the Society for Industrial and Applied Mathematics* 11 (2): 431–41.
- Melchers, Robert E., and André T. Beck. 2018. *Structural Reliability Analysis and Prediction*. 3rd ed. Wiley.
- Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. “Equation of State Calculations by Fast Computing Machines.” *The Journal of Chemical Physics* 21 (6): 1087–92. <https://doi.org/10.1063/1.1699114>.
- Ministry of Interior, Disaster and Emergency Management Presidency. 2018. “TSC2018, Turkish Seismic Code.” Ankara: Government of Republic of Türkiye.
- Nataf, Andre. 1962. “Détermination Des Distribution Don’t Les Marges Sont Donnees.” *Comptes Rendus de l’Académie Des Sciences* 225: 42–43.
- Nguyen, D, and B Widrow. 1990. “Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights.” In *1990 IJCNN International Joint Conference on Neural Networks*, 21–26 vol.3. <https://doi.org/10.1109/IJCNN.1990.137819>.
- Nguyen, Xuan Son, Alain Sellier, Frédéric Duprat, and Gérard Pons. 2009. “Adaptive Response Surface Method Based on a Double Weighted Regression Technique.” *Probabilistic Engineering Mechanics* 24 (2): 135–43. <https://doi.org/10.1016/j.probengmech.2008.04.001>.

- Nie, Jinsuo, and Bruce R Ellingwood. 2000. "Directional Methods for Structural Reliability Analysis." *Structural Safety* 22: 233–49.  
[www.elsevier.nl/locate/strusafe](http://www.elsevier.nl/locate/strusafe).
- Nie, Jinsuo, and Bruce R. Ellingwood. 2004. "A New Directional Simulation Method for System Reliability. Part II: Application of Neural Networks." *Probabilistic Engineering Mechanics* 19 (4): 437–47.  
<https://doi.org/10.1016/j.probengmech.2004.03.005>.
- Nowak, Andrzej S., and Kevin Collins. 2000. *Reliability of Structures*. 1st ed. McGraw-Hill Science/Engineering/Math.
- Oparaji, Uchenna, Rong Jiun Sheu, and Edoardo Patelli. 2017. "Robust Artificial Neural Network for Reliability Analysis." In *UNCECOMP 2017 - Proceedings of the 2nd International Conference on Uncertainty Quantification in Computational Sciences and Engineering*, 2017-January:651–62. National Technical University of Athens. <https://doi.org/10.7712/120217.5400.17104>.
- Papadopoulos, Vissarion, Dimitris G. Giovanis, Nikos D. Lagaros, and Manolis Papadrakakis. 2012. "Accelerated Subset Simulation with Neural Networks for Reliability Analysis." *Computer Methods in Applied Mechanics and Engineering* 223–224 (June): 70–80. <https://doi.org/10.1016/j.cma.2012.02.013>.
- Papadrakakis, Manolis, Vissarion Papadopoulos, and Nikos D. Lagaros. 1996. "Structural Reliability Analysis of Elastic-Plastic Structures Using Neural Networks and Monte Carlo Simulation." *Computer Methods in Applied Mechanics and Engineering* 136 (1–2): 145–63. [https://doi.org/10.1016/0045-7825\(96\)01011-0](https://doi.org/10.1016/0045-7825(96)01011-0).
- Papaioannou, Iason. 2012. "Non-Intrusive Finite Element Reliability Analysis Methods." PhD diss., Technische Universität München.
- Peter Bjerager, By. 1988. "Probability Integration by Directional Simulation." *Journal of Engineering Mechanics* 114 (8): 1285–1320.
- Rackwitz, Rüdiger, and Bernd Fiessler. 1978. "Structural Reliability under Combined Random Load Sequences." *Computers & Structures* 9 (5): 489–94.  
[https://doi.org/https://doi.org/10.1016/0045-7949\(78\)90046-9](https://doi.org/https://doi.org/10.1016/0045-7949(78)90046-9).

- Rajashekhar, Malur R, and Bruce R Ellingwood. 1993. "A New Look at the Response Surface Approach for Reliability Analysis." *Structural Safety* 12: 205–20.
- Ren, Yuan, and Guangchen Bai. 2011. "New Neural Network Response Surface Methods for Reliability Analysis." *Chinese Journal of Aeronautics* 24 (1): 25–31. [https://doi.org/10.1016/S1000-9361\(11\)60004-6](https://doi.org/10.1016/S1000-9361(11)60004-6).
- Richard, Benjamin, Christian Cremona, and Lucas Adelaide. 2012. "A Response Surface Method Based on Support Vector Machines Trained with an Adaptive Experimental Design." *Structural Safety* 39 (November): 14–21. <https://doi.org/10.1016/j.strusafe.2012.05.001>.
- Rosenblatt, Murray. 1952. "Remarks on a Multivariate Transformation." *The Annals of Mathematical Statistics* 23 (3): 470–72.
- Roussouly, N., F. Petitjean, and M. Salaun. 2013. "A New Adaptive Response Surface Method for Reliability Analysis." *Probabilistic Engineering Mechanics* 32 (April): 103–15. <https://doi.org/10.1016/j.pro bengmech.2012.10.001>.
- Shao, S, and Y Murotsu. 1997. "Structural Reliability Analysis Using a Neural Network." *JSME International Journal* 40 (3): 242–46.
- The MathWorks Inc. 2019. "MATLAB." Natick, Massachusetts: The MathWorks Inc. <https://www.mathworks.com>.
- Tvedt, L. 1983. "Two Second-Order Approximations to the Failure Probability." *Veritas Rep. RDIV/20-004083, Det Norske Veritas*.
- Wei, D., and S. Rahman. 2007. "Structural Reliability Analysis by Univariate Decomposition and Numerical Integration." *Probabilistic Engineering Mechanics* 22 (1): 27–38. <https://doi.org/10.1016/j.pro bengmech.2006.05.004>.
- Yu, Hao, and Bogdan M Wilamowski. 2018. "Levenberg–Marquardt Training." In *Intelligent Systems*, 11–12. CRC Press.
- Zhao, Yan-Gang, and Tetsuro Ono. 1999a. "New Approximations for SORM: Part I." *Journal of Engineering Mechanics* 125 (1): 79–85.
- Zhao, Yan-Gang, and Tetsuro Ono. 1999b. "New Approximations for SORM: Part II." *Journal of Engineering Mechanics* 125 (1): 86–93.



Zhu, Minjie, Frank McKenna, and Michael H. Scott. 2018. "OpenSeesPy: Python Library for the OpenSees Finite Element Framework." *SoftwareX* 7 (January): 6–11. <https://doi.org/10.1016/j.softx.2017.10.009>.

Zuev, Konstantin M., James L. Beck, Siu Kui Au, and Lambros S. Katafygiotis. 2012. "Bayesian Post-Processor and Other Enhancements of Subset Simulation for Estimating Failure Probabilities in High Dimensions." *Computers and Structures* 92–93 (February): 283–96. <https://doi.org/10.1016/j.compstruc.2011.10.017>.