Title
Optimization Algorithms for Performance-based Design and Structural Model Updating

Author
Choi, Euihyun

Publication Date
2023

Peer reviewed|Thesis/dissertation

Optimization Algorithms for Performance-based Design and Structural Model Updating

By

Euihyun Choi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Khalid M. Mosalam, Chair
Professor Filip C. Filippou
Professor Adityanand Guntuboyina

Spring 2023

Optimization Algorithms for Performance-based Design and Structural Model Updating

Abstract

Optimization Algorithms for Performance-based Design and Structural Model Updating

by

Euihyun Choi

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Professor Khalid M. Mosalam, Chair

Many problems in structural engineering can be defined as optimization problems and be solved by adopting proper optimization algorithms. This dissertation focuses on two optimization problems in structural engineering, namely, Performance-based Design Optimization (PDO) and Structural Model Updating (SMU).

In recent years, there have been many studies on the sustainability and resilience of building systems during their life-cycle. In particular, the holistic design framework using a Performance-Based Engineering (PBE) approach combined with the Multi-Attribute Utility Theory (MAUT), namely PBE-MAUT, is developed to provide a robust evaluation of the building performance in terms of sustainability and resiliency attributes. In PDO, the PBE-MAUT allows engineers to rank the design alternatives through the Generalized Expected Utility (GEU) (objective function to be maximized) containing the information about the risk attitude (or perception) of the decision makers. This dissertation proposes the PDO framework for seismic hazard using two meta-heuristic algorithms, namely, Genetic Algorithm (GA), abbreviated as PDO-GA and Bayesian Optimization Algorithm (BOA), abbreviated as PD-BOA. In this framework, probabilistic approaches are used to quantify the uncertainties in the different stages of the Performance-Based Earthquake Engineering (PBEE). For the *seismic hazard*, artificial accelerograms compatible with the response spectrum are generated by evolutionary Power Spectral Density (PSD) for seismic loading. For the *structural analysis*, distributions of the Engineering Demand Parameters (EDPs) are determined by using the Kernel Density Maximum Entropy Method (KDMEM), which provides the least biased probability density function from available data sets. For the combined *damage analysis* and *loss estimation*, GEU is used to evaluate the utility of the design alternatives based on the MAUT. The proposed framework adopts a Probability of Improvement (PI) function for the acquisition function of BOA. A hypothetical three-bay, five-story steel Moment Resisting Frame (MRF) building and three-bay, nine-story steel MRF building examples demonstrate the performances of the proposed framework.

Numerical models are very powerful tools used in simulation, damage detection, and evaluating physical structures. Accurate modeling of complex structures, however, remains challenging due to incomplete information (uncertainties) about the existing structure, which results in a difference between the responses of the numerical model and the measured responses of the actual "instrumented" structure. SMU is a process for improving the accuracy of a numerical model by reducing or even closing the gap between its prediction and the measured response of its physical counterpart through model parameter optimization. In this dissertation, the ABAQUS-Python Model Updating framework for overhead box beam highway sign structures is proposed. This is a Python-based framework that uses the ABAQUS software to create and analyze Finite Element (FE) models. The performance of the proposed framework is demonstrated by application to a single-post butterfly type overhead box beam sign structure on the California State Route 113, near the city of Davis, CA.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my gratitude to my advisor, Professor Khalid M. Mosalam, for his insightful guidance and thoughtful support. Professor Mosalam has been a true advisor, leader, and person, providing me with thoughtful counsel and motivation during challenging times.

I would also like to extend my gratitude to Professor Filip C. Filippou and Professor Adityanand Guntuboyina for their service on my dissertation and qualifying exam committees, and to Professor Matthew DeJong and Professor Laurent El Ghaoui for their participation on my qualifying exam committee. Thank you for generously sharing your valuable time and providing constructive feedback on my research.

As a Graduate Student Instructor, it was a great honor to work with Professor Armen Der Kiureghian and Professor Francisco Armero. Their enthusiasm for academia and devotion to teaching inspired me and inflamed my passion for learning.

I was fortunate to have the opportunity to intern at AECOM, and I am grateful to my supervisor, Fariborz Vossoughi, for providing me with valuable experiences and opportunities during and after the internship. I extend my gratitude to Bhosh Ravi Chandran, Julian Jaramillo, Larry Taicz, Mital Patel, Brendan O'Rourke, Ted Feldsher, and the entire team in the AECOM Oakland office for being wonderful colleagues and friends.

Without the friendships I have made at Berkeley, my graduate life at UC Berkeley would have been significantly more challenging. I would like to express my gratitude to Chrystal Chern, Jorge Archbold, Chuyang Chen, Fan Hu, Roberto Andreotti, Juan Meriles, Umberto Alibrandi, Selim Günay, Amarnath Kasalanati, and all members of Professor Mosalam's research group, STAIRlab and PEER Center. I am also grateful to my Korean friends, including Hahyung Park, Suhong Moon, Dayeol Lee, Hoyoung Lee, Wonjun Lim, Eddie Kim, and Minyoung Kim, for their unwavering support and friendship. I would like to extend special thanks to Soo Hyun Shin, Donghoon Kim, and Sebin Oh, who were not only my roommates but also my dear friends. The time we spent together at the Highland and Village will always remain in my heart.

Finally, I would like to dedicate this dissertation to my parents and my beloved wife. My parents, Sangjin Choi and Heesoo Lim, have always supported me with unconditional love. My wife, Jieun Byun, has been standing by my side throughout my journey to PhD with constant support and encouragement. I am so lucky to have you in my life.

Funding: I would like to acknowledge that my graduate studies and research have been financially supported by the Taisei Chair of Civil Engineering at the University of California-Berkeley, Kwanjeong Educational Foundation, and Harry H. Hilp Scholarship. The research in Chapter 5 received funding support from California Department of Transportation (Caltrans) for the "In-Service Structural Evaluation of Box Beam Overhead Sign Structures" project, Task Order 008 of the agreement number 65A0774.

Chapter 1

Introduction

1.1 Motivation

Seismic building design is a highly complex process that involves sophisticated structural analysis (e.g., nonlinear time-history analysis) and has various sources of uncertainties. Moreover, engineers should consider not only damage in structural elements of buildings, but also other sources of economic loss (e.g., repair cost and downtime). Therefore, various design tools and frameworks for supporting engineer's decision making are developed within the earthquake engineering community. The Pacific Earthquake Engineering Research (PEER) Center, `https://peer.berkeley.edu/`, developed the Performance-Based Earthquake Engineering (PBEE) methodology which is a robust framework for evaluating system performance measures in the interest of various stakeholders, such as monetary losses, downtime, or casualties [21]. In Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) program, a powerful Decision Support Tool for holistic building design is also developed, which allows engineers to determine the best design alternative in terms of holistic design (including not only safety but also resiliency and sustainability) [4] [28].

The primary objective of the engineer in design is to solve an optimization problem and find the "best" design. The PEER PBEE methodology is a robust method to determine the model's utility, which is an effective metric to decide on the best design among the design alternative sets. The best way to determine the optimal design is to evaluate every possible design alternative; however, this requires a large number of sophisticated analyses, which costs significant computational time and makes the optimization problem nonlinear and nonconvex. Therefore, a proper optimization algorithm needs to be developed for extension of the PEER PBEE methodology into the design space.

The performances of structural systems should be monitored during their life-cycle to ensure their sustainability and resilience. There are various methodologies for monitoring the structures, but creating a numerical model of the structure is a fundamental requirement [16]. However, accurate modeling of real structures is highly challenging due to the pres-

ence of uncertainties even in the information from various advanced inspection equipment and detailed as-built drawings [10] [29]. This discrepancy causes differences between the structural responses of the actual structure and that of the numerical model. Structural Model Updating (SMU) is used to reduce or even close the gap between the responses of the actual "physical" structure and the numerical model. This process involves parameterizing the numerical model by setting design variables and finding an optimized numerical model fitting the measured structural responses of the actual structure. As the evaluation of the numerical model requires complex Finite Element Analysis (FEA), like the PEER PBEE methodology, the development of a suitable optimization algorithm is crucial due to the required significant computational cost [6]. Additionally, the integration of the FEA software and a parameter-friendly code environment, such as Python or Matlab, requires a framework that accommodates various software programs.

1.2 Overview of the dissertation

This dissertation consists of 6 chapters including this introduction chapter. A description of each chapter is presented as follows:

- In **Chapter 2**, the Probabilistic Performance-Based Design (PPBD) framework which consists of four analysis steps is introduced with probabilistic approaches to determine the Engineering Demand Parameter (EDP) distribution and to apply the Multi Attribute Utility Theory (MAUT). The first section of this chapter reviews hazard analysis, structural analysis, damage analysis, and loss analysis in the PEER PBEE methodology. In the second section, the theoretical background for stochastic dynamic analysis and Kernel Density Maximum Entropy Method (KDMEM) are introduced. In the final section of this chapter, the background and the concept of Generalized Expected Utility (GEU) are introduced.

- In **Chapter 3**, two meta-heuristic algorithms, Genetic Algorithm (GA) and Bayesian Optimization Algorithm (BOA), are introduced and Performance-based Design Optimization (PDO) framework using either GA or BOA, i.e., PDO-GA or PD-BOA, is proposed.

- In **Chapter 4**, the applications of the proposed PDO framework to two steel Moment Resisting Frame (MRF) buildings are presented. These applications demonstrate the performance of the proposed PDO framework.

- In **Chapter 5**, the concept of the Structural Model Updating (SMU) is introduced and the ABAQUS-Python model updating framework for overhead highway sign structures is proposed. The application to a single-post butterfly sign structure is presented as a practical demonstration of the performance of the proposed SMU framework.

- In **Chapter 6**, the conclusions of this dissertation are made and suggestions for future research directions are proposed.

The dissertation also includes three appendices. Appendix A lists the properties of the AISC sections used in Chapter 4. Appendix B is the code listings used in Chapter 4. Finally, Appendix C is the code listings used in Chapter 5.

Chapter 2

Performance-based Design Evaluation Framework

2.1 Probabilistic Performance-based Design

The Probabilistic Performance-Based Design (PPBD) framework based on the PEER PBEE methodology is used for evaluating the seismic performances of structures or facilities. In this framework, there are four consecutive steps: (i) hazard analysis, (ii) structural analysis, (iii) damage analysis, and (iv) loss analysis. Figure 2.1 explains the analysis steps in the PPBD framework.

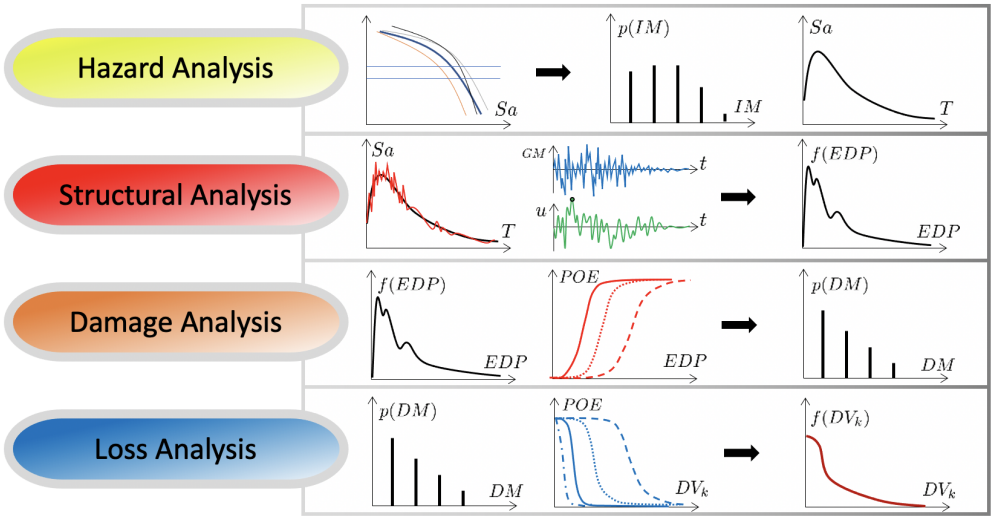


Figure 2.1: Analysis steps of the PPBD framework.

Hazard Analysis

In hazard analysis, we evaluate a seismic hazard considering the uncertainties in the seismic environment, such as nearby faults, their magnitude-frequency recurrence rates, fault mechanism, and site properties (e.g., distance and soil/site conditions). In that regard, we define an Intensity Measure (IM) which is a parameter of ground motions, e.g., spectral acceleration at the first natural period, T_1, i.e., $S_a(T_1)$, or Peak Ground Acceleration (PGA). In this process, we can obtain site-specific hazard curve, which shows a mean annual frequency of exceedance of the IM, Figure 2.2.

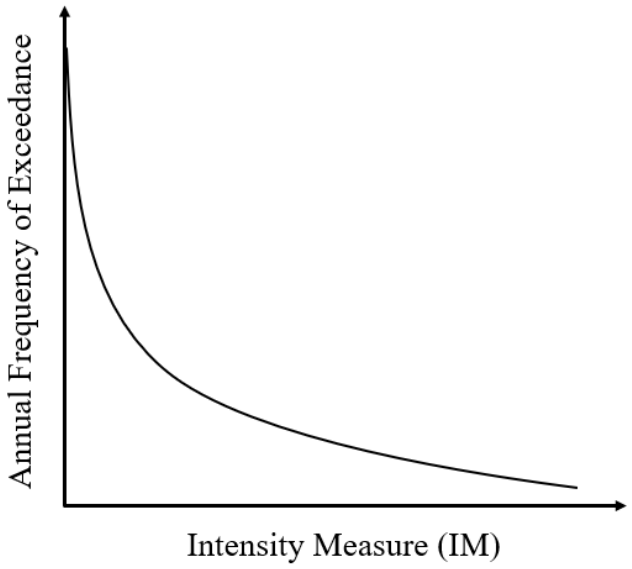


Figure 2.2: General hazard curve, annual frequency of exceedance of IM.

Since the occurrence of an earthquake is usually modeled by Poisson's model, the probability of exceedance of IM_0 in t years is expressed as follows,

$$P(IM \geq IM_0) = 1 - e^{-\lambda_{IM_0} t}, \tag{2.1}$$

where λ_{IM_0} is the mean annual frequency of exceedance of IM_0. Thus, the Probability Mass Function (PMF) of IM is

$$\begin{cases} p(IM_i) = e^{-\lambda_{IM_{i+1}} t} - e^{-\lambda_{IM_i} t}, & \forall\, i = 1, 2, \ldots, n-1, \\ p(IM_n) = 1 - e^{-\lambda_{IM_n} t}. \end{cases} \tag{2.2}$$

If multiple hazard curves for various natural periods of vibration of a structural system are available, the response spectrum of certain level of ground motion can be obtained. Figure 2.3 is an example of hazard curves and uniform hazard response spectrum of a Berkeley site[1].

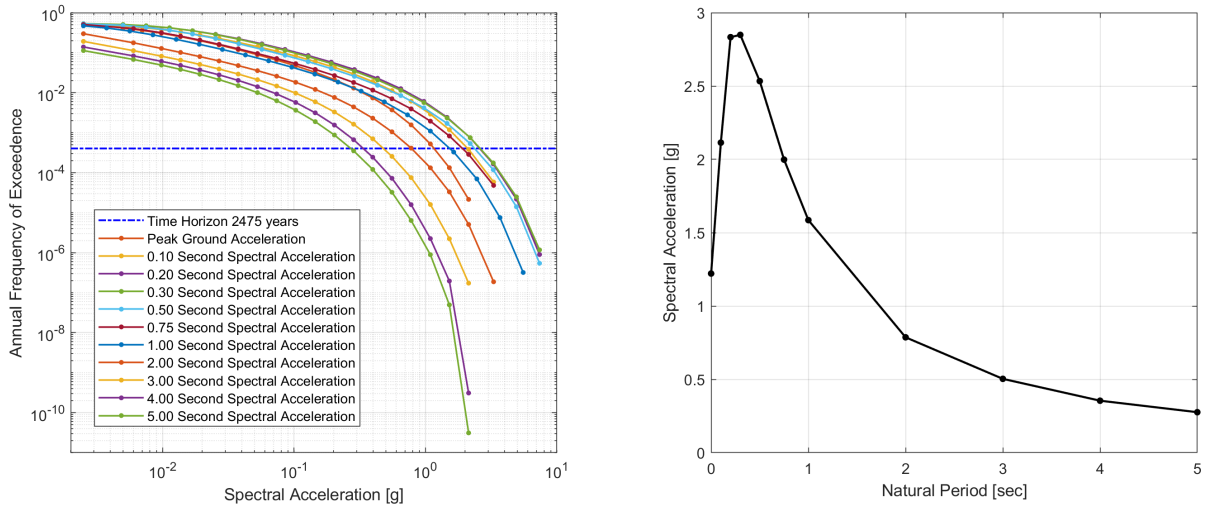


Figure 2.3: Hazard curves (left) and uniform hazard response spectrum (2% in 50 years, recurrence interval of 2,475 yr) (right).

Structural Analysis

From the site-specific hazard curves and response spectra, ground motions (seismic loads) are selected. Engineers perform structural analysis to obtain the responses of buildings to the selected seismic loads from each IM. Since the structures are under seismic loads, nonlinear time-history analyses are used in this step. The PPBD formulation takes parameters called Engineering Demand Parameters (EDPs) from the structural responses, which could be floor accelerations, inter-story drifts, or the entire time-history responses.

In the structural model, the engineer needs to consider the sources of uncertainties because the model typically does not perfectly represent the building in the real world. Uncertainties also arise from the probabilistic nature of the earthquake. Due to the uncertainties in the structural model and the seismic load (epistemic and aleatory uncertainties[2]), structural analysis also needs to be conducted in a probabilistic manner. Some probabilistic approaches are introduced in Chapter 3.

The final goal of this step is to obtain the distribution of EDP from each IM. From the results, the Probability Density Function (PDF) of EDP, $f(EDP)$, can be constructed by

[1]Source: USGS Unified Hazard Tool, `https://earthquake.usgs.gov/hazards/interactive/`.

[2]Epistemic uncertainty refers to modeling uncertainty due to lack of knowledge or data. Aleatory uncertainty refers to inherent randomness that cannot be explained.

the law of total probability. If we are targeting several EDPs, all of their distributions need to be determined in this step.

$$f(EDP_j) = \sum_{i=1}^{n} f(EDP_j|IM_i)p(IM_i) \qquad \forall\, j = 1, 2, \ldots, m, \tag{2.3}$$

where $f(EDP_j|IM)$ denotes the conditional probability of the j-th EDP given IM.

Damage Analysis

In earthquake design, EDPs are usually peak values of structural response, e.g., maximum story drift. The peak value is a desirable criterion to estimate the structure's damage, but this parameter cannot represent the entire response history. For this reason, engineers should account for the uncertainties in determining the damage state from EDPs. One of the efficient methods to account for this uncertainty is to define several Damage States (DSs). Each DS has quantitative information about the structural and nonstructural elements, and from corresponding fragility curves, Figure 2.4, the engineers can determine the probability of being in each DS at a certain EDP value.

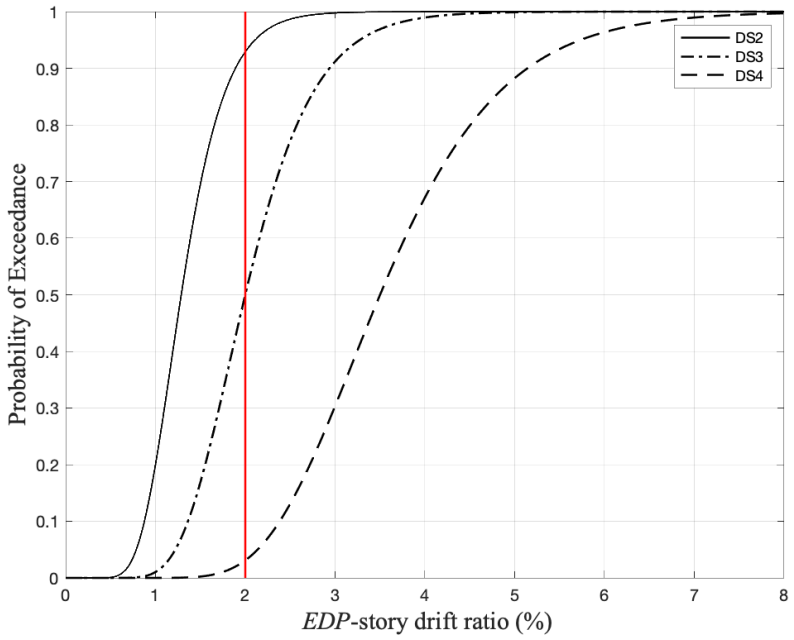


Figure 2.4: Example of fragility curves for the boundaries of four damage states where DS1, DS2, DS3, and DS4 correspond to no, minor, moderate, and major damages, respectively.

Figure 2.4 shows an example where each fragility curve follows a lognormal Cumulative Distribution Function (CDF) with the lognormal distribution parameters $\lambda_{DS2} = 0.252$,

$\lambda_{DS3} = 0.693$ and $\lambda_{DS4} = 1.253$, and $\zeta = 0.30$, i.e. $p(DS_i) \sim LN(\lambda_{DS_j}, \zeta)$, where λ and ζ are respectively the mean and standard deviation of the variable's (EDP's) natural logarithm. In this figure, the probability of being in DS1 (no damage) or worse is not plotted since that is always 100%. For the 2% story drift ratio, the red line in Figure 2.4 shows that the probability of being in DS1 (no damage), DS2, DS3, and DS4 is 0.07, 0.43, 0.47, and 0.03, respectively. With the PDF of an EDP and the fragility curves, we can obtain the PMF of a DS as follows,

$$p(DS_i) = \int_{EDP} p(DS_i|EDP) f(EDP) dEDP \qquad \forall\, i = 1, 2, \ldots, N. \tag{2.4}$$

Loss Analysis

In loss analysis, we determine the distributions of various Decision Variables (DVs) or performances which are critical variables in the decision-making, e.g., repair cost, downtime, or CO2 emission. With the loss curves as shown in Figure 2.5, the PDF of the DVs can be obtained by the law of total probability as follows,

$$p(DV_k) = \sum_{i=1}^{N} f(DV_k|DS_i) p(DS_i) \qquad \forall\, k = 1, 2, \ldots, p. \tag{2.5}$$

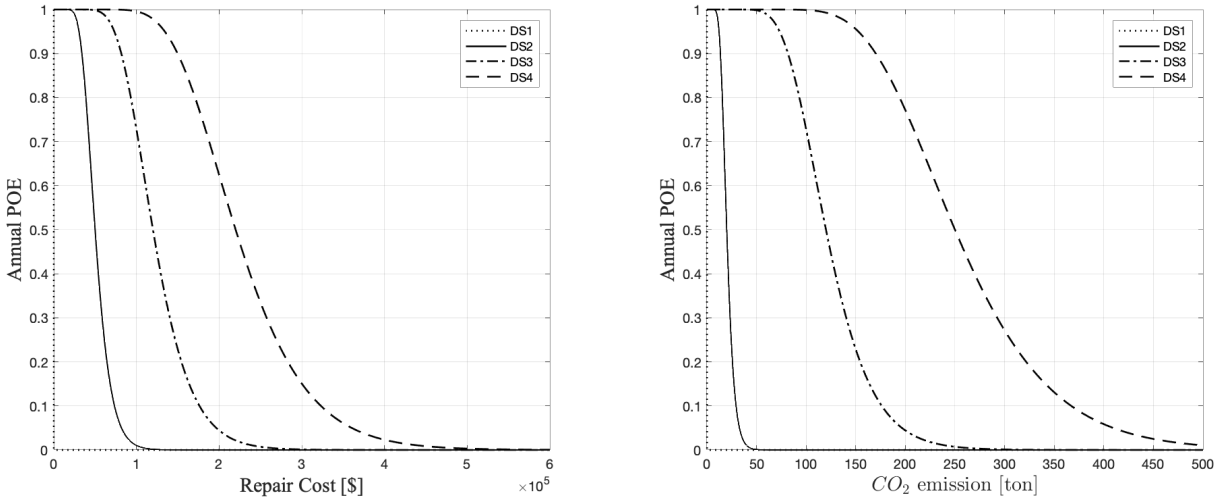


Figure 2.5: Loss curve examples of repair cost (left) and CO2 emission (right) [POE: Probability of Exceedance].

In reference [44], one of the DVs, expected total Repair Cost (RC), is determined as

$$RC = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} c_j(q_{j|DS_i})\, q_{j|DS_i}\, p(DS_i) \right), \tag{2.6}$$

where c_j is a unit cost of the j-th element, $q_{j|DS_i}$ is a quantity of the j-th element when the damage state is DS_i, and $p(DS_i)$ is the probability that the damage state is DS_i. Moreover, Yang et al. [44] suggested to consider the uncertainties in the unit repair costs by making them random variables as shown in Figure 2.6.

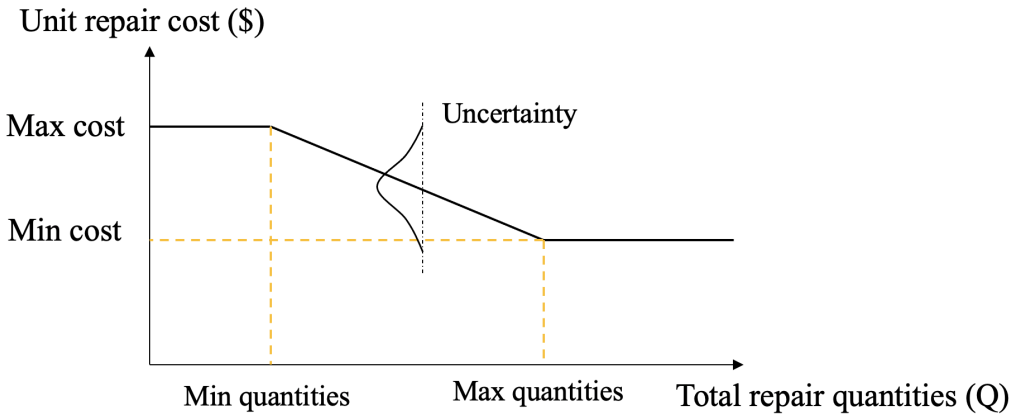


Figure 2.6: The unit repair cost function.

2.2 Probabilistic Approaches to EDP Distributions

Stochastic Dynamic Analysis

As the result of hazard analysis, response spectrum for each IM is obtained. For the next step (structural analysis) of the PPBD framework, a number of ground motions are selected based on the response spectrum. In general, engineers pick ground motions from a database and scale them to the target response spectrum. However, this procedure requires some efforts in selecting and scaling the original ground motions to a specific response spectrum [3]. Thus, in this framework, we use the stochastic model, which does not require that engineers conduct probabilistic seismic hazard analysis involving the process of selecting ground motions from a database. This stochastic dynamic analysis method generates artificial accelerograms without the need for any data processing.

Stationary Model

The common approaches in generating artificial ground motion is using Gaussian models, which can be defined by their evolutionary Power Spectral Density (PSD) as follows,

$$\begin{cases} G_{\ddot{u}_g}(t,\omega) = 0, & \text{if } \omega < 0, \\ G_{\ddot{u}_g}(t,\omega) = \left|\varphi(t,\omega)\right|^2 G_{\ddot{u}_g}(\omega), & \text{if } \omega \geq 0, \end{cases} \tag{2.7}$$

where $\varphi(t,\omega)$ is the frequency-dependent modulating function and $G_{\ddot{u}_g}(\omega)$ is the one-sided PSD of the stationary part of the ground motion acceleration time series $\ddot{u}_g(t)$.

Cacciola et al. [11][12] proposed the model for stationary PSD. Following a recursive expression, the stationary PSD can be determined as

$$\begin{cases} G_{\ddot{u}_g}(t,\omega_i) = 0, & \text{if } 0 \leq \omega_i \leq \omega_s, \\ G_{\ddot{u}_g}(t,\omega_i) = \dfrac{4\xi}{\pi\omega_i - 4\xi\omega_i} \times \left[\dfrac{S_a(\omega_i)^2}{\eta_U(\omega_i,\xi)^2} - \Delta\omega \sum_{j=1}^{i-1} G_{\ddot{u}_g}(\omega_j) \right], & \text{if } \omega_i > \omega_s, \end{cases} \tag{2.8}$$

where $\Delta\omega \leq 2\pi/t_s$, t_s is time observing window, $S_a(\omega_i)$ is the spectral acceleration obtained from the response spectrum for frequency ω_i, and $\eta_U(\omega_i,\xi)^2$ denotes the peak factor under the *hypothesis of a barrier outcrossing in clumps*, which depends on ω_i and the damping ratio ξ, and can be evaluated according to [12] as follows,

$$\eta_U(\omega_i,\xi) = \sqrt{2\log\left[2N_U\left\{1-\exp\left(-\delta_U^{1.2}\sqrt{\pi\log(2N_U)}\right)\right\}\right]}, \tag{2.9}$$

with the parameter N_U and the spread factor δ_U of the response U defined in [12] as follows,

$$N_U = \frac{t_s}{2\pi}\omega_i(-\log p)^{-1}, \tag{2.10}$$

and

$$\delta_U = \sqrt{1 - \frac{1}{\xi^2}\left(1 - \frac{2}{\pi}\arctan\left(\frac{\xi}{\sqrt{1-\xi}}\right)\right)^2}. \tag{2.11}$$

For a stationary model, the modulating function $\varphi(t,\omega) = 1$. Thus, the stochastic ground motion $\ddot{u}_g(t)$ can be defined by following discrete Fourier series [3][12] as follows,

$$\begin{aligned} \ddot{u}_g(t,\boldsymbol{u}) &= \sum_{k=1}^{n} \sqrt{G_{\ddot{u}_g}(\omega_k)\Delta\omega}\left[\cos(\omega_k t)\boldsymbol{u_k^c} + \sin(\omega_k t)\boldsymbol{u_k^s}\right], \\ &= \sum_{k=1}^{n} s_k^c(t)\boldsymbol{u_k^c} + s_k^s(t)\boldsymbol{u_k^s}, \\ &= \boldsymbol{s}(t,\omega)\cdot\boldsymbol{u}, \end{aligned} \tag{2.12}$$

where $\boldsymbol{u} = \left\{\boldsymbol{u_k^c}, \boldsymbol{u_k^s}\right\}$ is a vector of normal standard random variables, and $\boldsymbol{s}(t,\omega) = \left\{s_k^c(t), s_k^s(t)\right\} = \left\{\sigma_k(t)\cos(\omega_k t), \sigma_k(t)\sin(\omega_k t)\right\}$ with $\sigma_k = \sqrt{G_{\ddot{u}_g}(\omega_k)\Delta\omega}$.

Quasi-Stationary Model

For the quasi-stationary model, the modulating function only depends on time t. Thus, $\varphi(t,\omega) = \phi(t)$. The modulating function proposed in [25] is as follows,

$$\phi_J(t) = \begin{cases} \left(\frac{t}{t_1}\right)^2, & \text{if } t < t_1, \\ 1, & \text{if } t_1 \le t < t_2, \\ \exp[-\beta_J(t-t_2)], & \text{if } t_2 \le t, \end{cases} \tag{2.13}$$

where $t_1 = \frac{2.5}{\beta_J}$ and $t_2 = \frac{11.5}{\beta_J}$ with $\beta_J = 0.9$. With the modulating function, $\phi_J(t)$, the stochastic ground motion $\ddot{u}_g(t)$ following the quasi-stationary model is obtained using Eqs 2.12 but with $G_{\ddot{u}_g}(\omega_k)$ replaced by $G_{\ddot{u}_g}(t,\omega_k)$. Examples of artificial accelerograms and corresponding response spectra by the stationary and Jennings et al. [25] quasi-stationary models are shown in Figure 2.7.

Kernel Density Maximum Entropy Method

In the second step of the PPBD framework, engineers need to determine the PDF of an EDP from structural analysis considering existing uncertainties. For the classical approach of this step, engineers select a suitable probability distribution for the PDF of the EDP. In particular, the shape of a lognormal CDF is generally used for the story drift ratio. However, in some cases, the determined parametric distributions cannot fit the data well; thus, the validity of such assumptions remain questionable [5] [37].

This section presents the Kernel Density Maximum Entropy Method (KDMEM) to estimate the PDF of the selected EDP. This method provides the least biased PDF from available data sets and has benefits in computational cost since the associated optimization problem is *convex* [5], which can be solved with the open source module in Python, CVXPY [18].

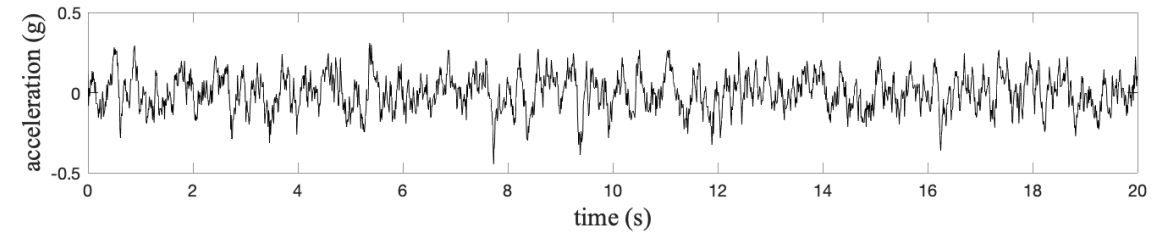
Shannon's Entropy

In 1948, Shannon [35] found a unique quantity H measuring the uncertainties of an information source, which is named *entropy*, and the Shannon's entropy functional of the discrete distribution $\boldsymbol{p} = \{p_1, p_2, \ldots, p_N\}$ is defined as follows,

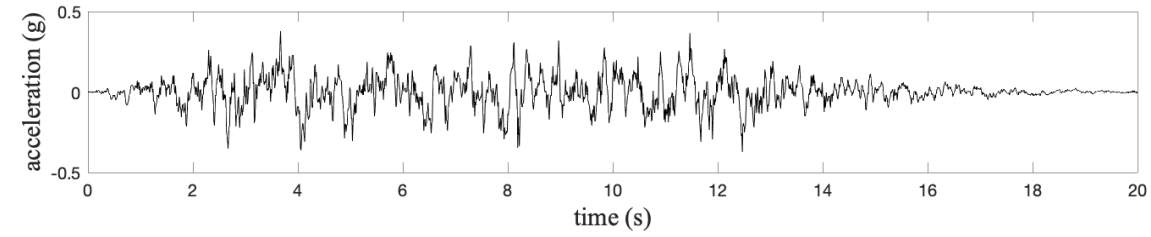
$$H(p_1, \cdots, p_N) = -\sum_{i=1}^{N} p_i \log p_i, \tag{2.14}$$

where "log" in this dissertation is the natural logarithm.

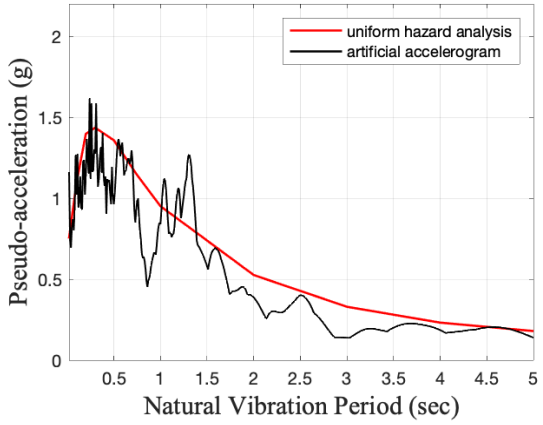
The larger entropy represents larger randomness, which means it is more unpredictable [2]. On the other hand, the entropy of a continuous random variable X with PDF $f_X(x)$ and



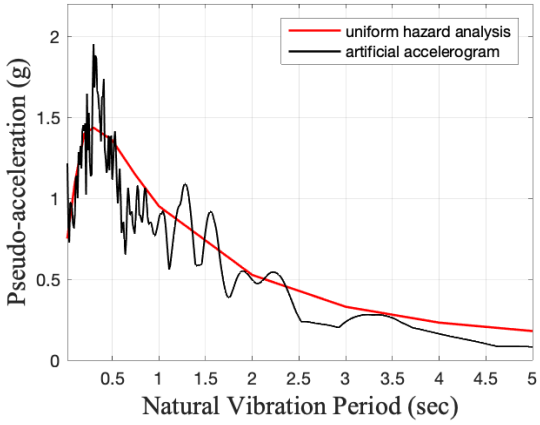
(a) Stationary model



(b) Quasi-stationary model



(c) Stationary model



(d) Quasi-stationary model

Figure 2.7: Artificial accelerograms and corresponding response spectra by stationary and Jennings et al. [25] quasi-stationary models.

support (i.e., the set of values that the random variable can take), $\mathcal{X}$, is defined as follows,

$$H(f) = -\mathbb{E}[\log f_X(x)] = -\int_{\mathcal{X}} f_X(x) \log f_X(x) dx, \tag{2.15}$$

where $\mathbb{E}[\log f_X(x)]$ is the expected value of $\log f_X(x)$.

Maximum Entropy Probability Density Estimation

For a discrete random variable, X, if we maximize the entropy of a probability distribution, $p_i = P(X = x_i), \forall\ i = 1, 2, \ldots, N$, and $\sum_{i=1}^{N} p_i = 1$, with given constraints, $G_k = \sum_{i=1}^{N} g_k(x_i)\ p_i, \forall\ k = 1, 2, \ldots, M$, the problem can be expressed as follows,

$$\begin{aligned} \boldsymbol{p}^* &= \underset{\boldsymbol{p}}{\operatorname{argmax}}\, H(\boldsymbol{p}), \\ \text{s.t. } &\sum_{i=1}^{n} p_i = 1, \\ &\sum_{i=1}^{n} g_k(x_i)\ p_i = G_k \quad \forall\ k = 1, 2, \ldots, M. \end{aligned} \tag{2.16}$$

If there is no constraint, the solution will be $p_i = \frac{1}{N}, i = 1, 2, \ldots, N$. We can build a new function $\mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda})$ with Lagrange multipliers, $\lambda_k, k = 0, 1, \ldots, M$, as follows,

$$\mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda}) = H(\boldsymbol{p}) - (\lambda_o - 1)\left(\sum_{i=1}^{N} p_i - 1\right) - \sum_{k=1}^{M} \lambda_k \left(\sum_{i=1}^{N} g_k(x_i) p_i - G_k\right). \tag{2.17}$$

The above Lagrangian, $\mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda})$, provides an upper bound of the solution according to [13], which is stated as follows,

$$\max_{\boldsymbol{p}} H(\boldsymbol{p}) = \max_{\boldsymbol{p}} \min_{\boldsymbol{\lambda}} \mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda}) \leq \min_{\boldsymbol{\lambda}} \max_{\boldsymbol{p}} \mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda}) = \min_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \tag{2.18}$$

where the dual function $g(\boldsymbol{\lambda})$ is expressed as follows,

$$g(\boldsymbol{\lambda}) = \max_{\boldsymbol{p}} \mathcal{L}_p(\boldsymbol{p}, \boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{p}^*, \boldsymbol{\lambda}). \tag{2.19}$$

Accordingly,

$$\frac{\partial}{\partial p_i} \mathcal{L}(\boldsymbol{p}, \boldsymbol{\lambda})\bigg|_{\boldsymbol{p}=\boldsymbol{p}^*} = -\log p_i^* - \lambda_0 - \sum_{k=1}^{M} \lambda_k g_k(x_i) = 0 \qquad \forall\ i = 1, 2 \ldots, N. \tag{2.20}$$

Using superscript ME for the Maximum Entropy, we obtain,

$$\log p_i^* = -\lambda_0 - \nu_i(\boldsymbol{\lambda}) \qquad \forall\ i = 1, 2, \ldots, N, \tag{2.21}$$

$$p_i^{ME}(\boldsymbol{\lambda}) = \exp(-\lambda_0)\exp\left(-\nu_i(\boldsymbol{\lambda})\right) \qquad \forall\, i = 1, 2, \ldots, N, \tag{2.22}$$

where

$$\nu_i(\boldsymbol{\lambda}) = \sum_{k=1}^{M} \lambda_k g_k(x_i) \qquad \forall\, i = 1, 2, \ldots, N. \tag{2.23}$$

From Eqs 2.14, 2.17, 2.19, and 2.22, we obtain,

$$\begin{aligned} g(\boldsymbol{\lambda}) = &-\sum_{i=1}^{N}\left(\exp(-\lambda_0)\exp\left(-\nu_i(\boldsymbol{\lambda})\right)\left(-\lambda_0 - \nu_i(\boldsymbol{\lambda})\right)\right) \\ &-(\lambda_0 - 1)\left(\exp(-\lambda_0)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda})\right) - 1\right) \\ &-\sum_{k=1}^{M}\lambda_k\left(\sum_{i=1}^{N}\left(g_k(x_i)\exp(-\lambda_0)\exp\left(-\nu_i(\boldsymbol{\lambda})\right)\right) - G_k\right). \end{aligned} \tag{2.24}$$

Accordingly,

$$\begin{aligned} \frac{\partial}{\partial\lambda_0} g(\boldsymbol{\lambda})\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^*} &= \exp(-\lambda_0^*)\sum_{i=1}^{N}\left(\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right)\left(-\lambda_0^* - \nu_i(\boldsymbol{\lambda}^*)\right)\right) \\ &\quad + \exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \\ &\quad - \left(\exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) - 1\right) \\ &\quad + (\lambda_0^* - 1)\exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \\ &\quad - \sum_{k=1}^{M}\lambda_k^*\left(-\exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right)g_k(x_i)\right) \\ &= 1 - \exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \\ &= 0. \end{aligned} \tag{2.25}$$

Thus,

$$\exp(-\lambda_0^*)\sum_{i=1}^{N}\exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) = 1, \tag{2.26}$$

$$\lambda_0^*(\lambda_1, \lambda_2, \ldots, \lambda_M) = \log\left(\sum_{i=1}^{N} \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right)\right). \tag{2.27}$$

Moreover,

$$\begin{aligned}
\frac{\partial}{\partial \lambda_k} g(\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^*} &= \sum_{i=1}^{N} \left(g_k(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \left(1 - \lambda_0^* - \nu_i(\boldsymbol{\lambda}^*)\right)\right) \\
&\quad - \sum_{i=1}^{N} \left(g_k(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) (1 - \lambda_0^*)\right) \\
&\quad - \sum_{i=1}^{N} g_k(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) + G_k \\
&\quad + \sum_{i=1}^{N} \nu_i(\boldsymbol{\lambda}^*) g_k(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right), \\
&= G_k - \sum_{i=1}^{N} g_j(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right), \\
&= 0 \qquad\qquad \forall\, k = 1, 2, \ldots, M.
\end{aligned} \tag{2.28}$$

Thus,

$$G_k = \sum_{i=1}^{N} g_k(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \qquad\qquad \forall\, k = 1, 2, \ldots, M. \tag{2.29}$$

From Eq. 2.24 with $\boldsymbol{\lambda}$ replaced by $\boldsymbol{\lambda}^*$, and making use of Eqs 2.26 and 2.29, we obtain,

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}) = g(\boldsymbol{\lambda}^*) &= \sum_{i=1}^{N} \left(-\exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \left(-\lambda_0^* - \nu_i(\boldsymbol{\lambda}^*)\right)\right), \\
&= \sum_{i=1}^{N} \lambda_0^* \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right) \\
&\quad + \sum_{k=1}^{M} \lambda_k^* \sum_{i=1}^{N} g_j(x_i) \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda}^*)\right), \\
&= \lambda_0^*(\lambda_1, \lambda_2, \ldots, \lambda_M) + \sum_{k=1}^{M} \lambda_k^* G_k.
\end{aligned} \tag{2.30}$$

It is noted that the final equation in Eqs 2.30 makes use of Eqs 2.27 and 2.29. Thus, if there is a feasible solution, the uniqueness of this solution is guaranteed by the following

unconstrained convex functional [2][5][38],

$$\Gamma^{ME}(\lambda_1, \lambda_2, \ldots, \lambda_M) = \lambda_0^*(\lambda_1, \lambda_2, \ldots, \lambda_M) + \sum_{k=1}^{M} \lambda_k G_k, \tag{2.31}$$

where $\lambda_0^*(\lambda_1, \lambda_2, \ldots, \lambda_M)$ is obtained from Eq. 2.27. Thus, the solution for the maximum entropy discrete distribution is obtained from Eq. 2.22 with λ's replaced by λ^*'s, as follows,

$$p_i^* = p_i^{ME}(\boldsymbol{\lambda^*}) = \exp(-\lambda_0^*) \exp\left(-\nu_i(\boldsymbol{\lambda^*})\right), \tag{2.32}$$

where

$$\boldsymbol{\lambda^*} = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}}\, \Gamma^{ME}(\boldsymbol{\lambda}). \tag{2.33}$$

Similarly, for a continuous random variable, the optimization problem is stated as follows,

$$\begin{aligned} f^* &= \underset{f}{\operatorname{argmax}}\, H(f), \\ \text{s.t.} &\int_{\mathcal{X}} f_X(x)dx = 1, \\ &\int_{\mathcal{X}} g_k(x) f_X(x)dx = G_k \quad \forall\, k = 1, 2, \ldots, M. \end{aligned} \tag{2.34}$$

The solution (with $\boldsymbol{\lambda^*}$ from Eq. 2.33) of this problem is also similarly expressed follows,

$$f_{ME}(x) = \exp(-\lambda_0^*) \exp\left(-\nu(x, \boldsymbol{\lambda^*})\right), \tag{2.35}$$

where

$$\lambda_0^*(\lambda_1, \lambda_2, \ldots, \lambda_M) = \log\left(\int_{\mathcal{X}} \exp\left(-\nu(x, \boldsymbol{\lambda^*})\right) dx\right), \tag{2.36}$$

and

$$\nu(x, \boldsymbol{\lambda^*}) = \sum_{k=1}^{M} \lambda_k^* g_k(x) \qquad \forall\, i = 1, 2, \ldots, N. \tag{2.37}$$

Maximum Entropy Distribution with Power Moments

In general, the given constraints are power moments, i.e., $G_k = \mathbb{E}[X^k]$ and $g_k(x_i) = x_i^k$. Typically, k is a positive integer number, i.e., $k \in \mathbb{Z}^+$, where $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$. The maximum entropy distribution generated by power moments requires higher k of moments to obtain accurate description of the tails of the distribution. However, when k of moment is high, the optimization problem can be ill-posed, which may induce instability in the maximum entropy algorithms [43]. For this reason, Alibrandi and Mosalam [5] used fractional moment $\mathbb{E}[X^\alpha]$, where α is a real number, i.e., $\alpha \in \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers.

The Taylor series expansion of x^α at $x = b$ is expressed as follows,

$$x^\alpha = b^\alpha + \frac{\alpha}{1!}b^{\alpha-1}(x-b) + \frac{\alpha(\alpha-1)}{2!}b^{\alpha-2}(x-b)^2 + \cdots, \tag{2.38}$$

$$= \sum_{i=0}^{\infty} \binom{\alpha}{i} b^{\alpha-i}(x-b)^i. \tag{2.39}$$

It is noted that the combination $C(\alpha, i) = \binom{\alpha}{i} = \alpha!/\left(i!\,(\alpha-i)!\right)$. Moreover, the fractional moments can be defined with power moments as follows,

$$\mathbb{E}\left[X^\alpha\right] = \sum_{i=0}^{\infty}\sum_{k=0}^{i}(-1)^{i-k}\binom{\alpha}{i}\binom{i}{k}b^{\alpha-k}\mathbb{E}\left[X^k\right]. \tag{2.40}$$

Thus, we can have information from numerous power moments with few fractional moments. With the selection of real numbers for the powers, $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_k, \ldots, \alpha_M\}$, the maximum entropy distribution, making use of Eqs 2.35 and 2.37, is as follows,

$$f_{ME}(x; \boldsymbol{\lambda^*}, \boldsymbol{\alpha}) = \exp(-\lambda_0^*)\exp\left(-\sum_{k=1}^{M}\lambda_k x^{\alpha_k}\right), \tag{2.41}$$

where, from Eq. 2.36, we have,

$$\lambda_0^*(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \log\left(\int_{\mathcal{X}}\exp\left(-\sum_{k=1}^{M}\lambda_k x^{\alpha_k}\right)dx\right), \tag{2.42}$$

and, expanding Eq. 2.33, we can write,

$$(\boldsymbol{\lambda^*}, \boldsymbol{\alpha^*}) = \underset{\boldsymbol{\lambda},\boldsymbol{\alpha}}{\operatorname{argmin}}\,\Gamma_\alpha^{ME}(\boldsymbol{\lambda}, \boldsymbol{\alpha}). \tag{2.43}$$

It is to be noted that Γ^{ME} is a convex functional in $\boldsymbol{\lambda}$, but not in $\boldsymbol{\alpha}$. Therefore, we need to check the solution for several $\boldsymbol{\alpha}$ alternatives to determine the optimal solution.

Kernel Density Estimation

The Kernel Density (KD) estimation is non-parametric estimator of the probability density function of a random variable [27][36]. For a random variable, X, having $x_i, i = 1, 2, \ldots, N$, identically distributed and independent samples with kernel function f_X, the kernel density estimator is expressed as follows,

$$f_{KD}(x) = \frac{1}{N}\sum_{i=1}^{N} f_X(x; x_i, h), \tag{2.44}$$

where h is the bandwidth. The choice of h is very important for the entire distribution [40].

For the kernel density estimation, we need to select a kernel function. In general, such function depends on the domain of the random variable. If the target distribution has unbounded support, Gaussian is possible as a kernel function and if the target distribution has semi-bounded support, the lognormal distribution is usually adopted. This latter choice is expressed as follows,

$$f_{KD}(x) = \sum_{i=1}^{N} p_i f_X^{LN}(x; \hat{\lambda}_i, \hat{\zeta}_i), \tag{2.45}$$

where

$$f_X^{LN}(x; \hat{\lambda}_i, \hat{\zeta}_i) = \frac{1}{x\hat{\zeta}_i\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\log x - \hat{\lambda}_i}{\hat{\zeta}_i}\right)^2\right), \tag{2.46}$$

and

$$\hat{\lambda}_i = \log\left(\frac{x_i^2}{\sqrt{x_i^2 + h^2}}\right), \quad \hat{\zeta}_i = \log\left(1 + \left(\frac{h}{x_i}\right)^2\right) \quad \forall\, i = 1, 2, \ldots, N. \tag{2.47}$$

Kernel Density Maximum Entropy Algorithm

Algorithm 1 is used in the proposed framework for the implementation of the KDMEM following the steps discussed in [5]. An example of the use of the implementation of Algorithm 1 is illustrated in Figure 2.8, where $\mathcal{P}(\cdot)$ is the power set.

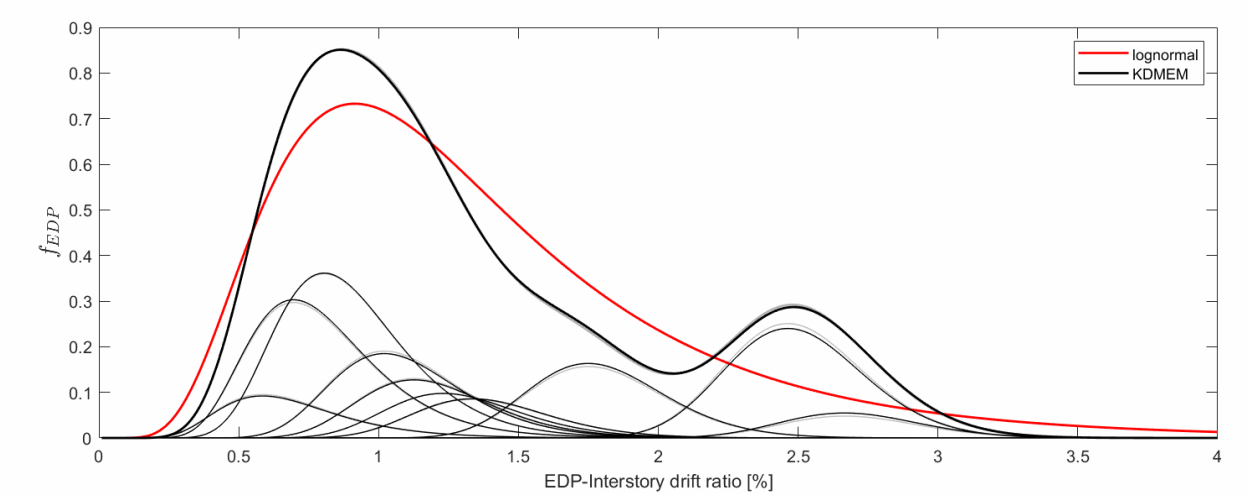


Figure 2.8: Example of KDMEM with 11 data points, $\mathbf{x}$ =[0.5, 0.7, 0.8, 0.9, 1.1, 1.2, 1.3, 1.4, 1.8, 2.5, 2.7].

Algorithm 1: Kernel Density Maximum Entropy Method.

Input : $\Delta\alpha$,α_{max},M

$Q = \alpha_{max}/\Delta\alpha$

$\boldsymbol{\alpha_{set}} = \{\alpha_n = n\Delta\alpha | n \in \{1, 2, \ldots, Q\}\}$

$\mathbf{A_{searched}} = \emptyset$

$H^* \leftarrow 0$

for $j = 1, 2, \ldots, \binom{Q}{M}$ **do**

  Select one $\boldsymbol{\alpha^j} \in \{\boldsymbol{\alpha} \in \mathcal{P}(\boldsymbol{\alpha_{set}}) | \, |\boldsymbol{\alpha}| = M, \boldsymbol{\alpha} \notin \mathbf{A_{searched}}\}$

  $G_k = \frac{1}{N}\sum_{i=1}^{N} x_i^{\alpha_k^j}$, $\forall\, k = 1, 2, \ldots, M$

  $g_{ik} = \int_X x^{\alpha_k^j} f_X(x; x_i, h)dx$, $i = 1, 2, \ldots, N$; $k = 1, 2, \ldots, M$

  $\boldsymbol{p_j^*} = \underset{\boldsymbol{p}}{\operatorname{argmax}}\, H(\boldsymbol{p}, \boldsymbol{\alpha^j}) : \sum_{i=1}^{N} p_i = 1, \sum_{i=1}^{N} g_{ik} p_i = G_k \;\; \forall\, k = 1, 2, \ldots, M$

  if $H^* < H(\boldsymbol{p_j^*}, \boldsymbol{\alpha^j})$ **then**

    $H^* \leftarrow H(\boldsymbol{p_j^*}, \boldsymbol{\alpha^j})$

    $\boldsymbol{\alpha^*} \leftarrow \boldsymbol{\alpha^j}$

    $\boldsymbol{p^{ME}} \leftarrow \boldsymbol{p_j^*}$

  $\mathbf{A_{searched}} \leftarrow \mathbf{A_{searched}} \cup \{\boldsymbol{\alpha^j}\}$

Output: $\boldsymbol{p^{ME}}$

2.3 Multi Attribute Utility Theory

In performance-based design, engineers should consider various performances in decision-making. In addition, due to the probabilistic nature of the PBEE methodology, the evaluated performances are not deterministic. The outcomes of the four-consecutive analyses are the inputs to this multi-objective probabilistic problem. Therefore, it is difficult to rank different design alternatives in the setting of such decision-making problem. For one design alternative to be better than another without risks, one should Pareto dominates[3] the other [15].

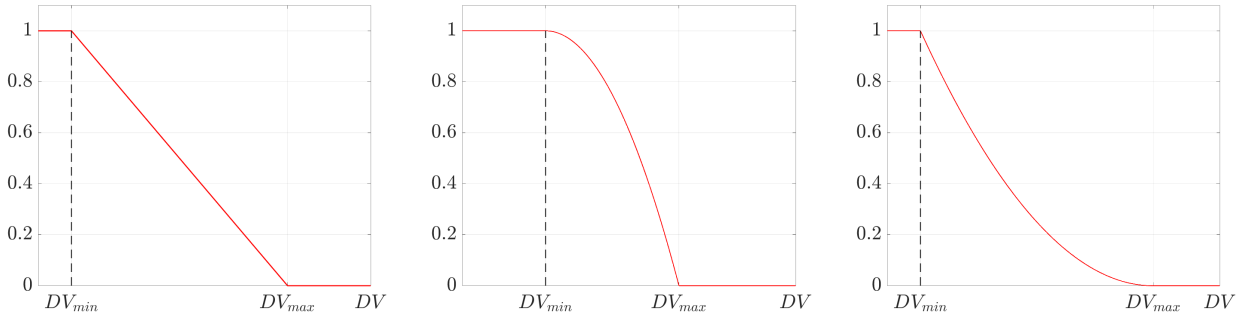
Finding the one design alternative dominating the others is not possible in some cases of design problems. For example, if an engineer is considering both construction cost and down-time due to earthquakes in a building design, there will be obvious trade-off between the two performances. Since negative correlation between the construction cost and the downtime is expected, it is difficult to determine the design that is both cheaper (less construction cost) and more resilient (short functional recovery time following an extreme event, e.g., a major

[3]Pareto optimal solution is defined as a set of "non-inferior" solutions in the objective space defining a boundary beyond which none of the objectives can be improved without sacrificing at least one of the other objectives.

earthquake) than the others.

Generalized Expected Utility

As discussed above, it is difficult to decide which is the best design alternative when there are multiple criteria. Alibrandi and Mosalam [4][28] developed the Multi-Attribute Utility Theory (MAUT) in conjunction with PBE, namely PBE-MAUT. The proposed PDO framework adopts PBE-MAUT for defining the objective functions in the design optimization problems. For each DV (performance), DV_k, $k = 1, 2, \ldots, p$, we can quantify its utility with single-attribute utility functions $u_k(DV_k)$, $k = 1, 2, \ldots, p$, such that $u_k(DV_k) : \mathcal{X}_{DV_k} \rightarrow [0, 1]$ where $\mathcal{X}_{DV_k}$ is the domain of DV_k. The utility functions reflect the stakeholders' risk management strategies towards the decision-making. Figure 2.9 shows examples of linear (risk-neutral), concave down (risk-averse), and concave up (risk-seeking) single attribute utility functions.



(a) Linear (risk-neutral) (b) Concave down (risk-averse) (c) Concave up (risk-seeking)

Figure 2.9: Examples of single attribute utility functions.

The multi-attribute utility function, or total utility function, can be expressed as a combination of single attribute utility functions with weights $0 \leq w_k \leq 1$, $k = 1, 2, \ldots, p$ and $\sum_{k=1}^{p} w_k = 1$ as follows,

$$u(DV_1, DV_2, \ldots, DV_p) = \sum_{k=1}^{p} w_k u_k(DV_k). \tag{2.48}$$

From the distributions of the DVs, resulting from the loss analysis, we can determine the PDF of the total utility as follows,

$$f_U(u) = \int_{DV:u(DV)=u} f(DV_1, DV_2, \ldots, DV_p)\, dDV. \tag{2.49}$$

The final step of the PBE-MAUT adopts the Generalized Expected Utility (GEU), Eq. 2.50, to quantify the utility level of the PDF given in Eq. 2.49.

$$GEU(\alpha) = \frac{1}{\alpha} \int_0^{\alpha} u\, dF_U = \frac{1}{\alpha} \int_0^{F_U^{-1}(\alpha)} f_U(u)\, du = \mathbb{E}[U | 0 \leq U \leq F_U^{-1}(\alpha)], \tag{2.50}$$

where $F_U(u)$ is a CDF of the total utility. Accordingly, the GEU is the α-superquantile which depends on α, and $GEU \in [0, 1]$. In Figure 2.10, we can observe that the GEU is the first moment of the shaded (cyan colored) area.

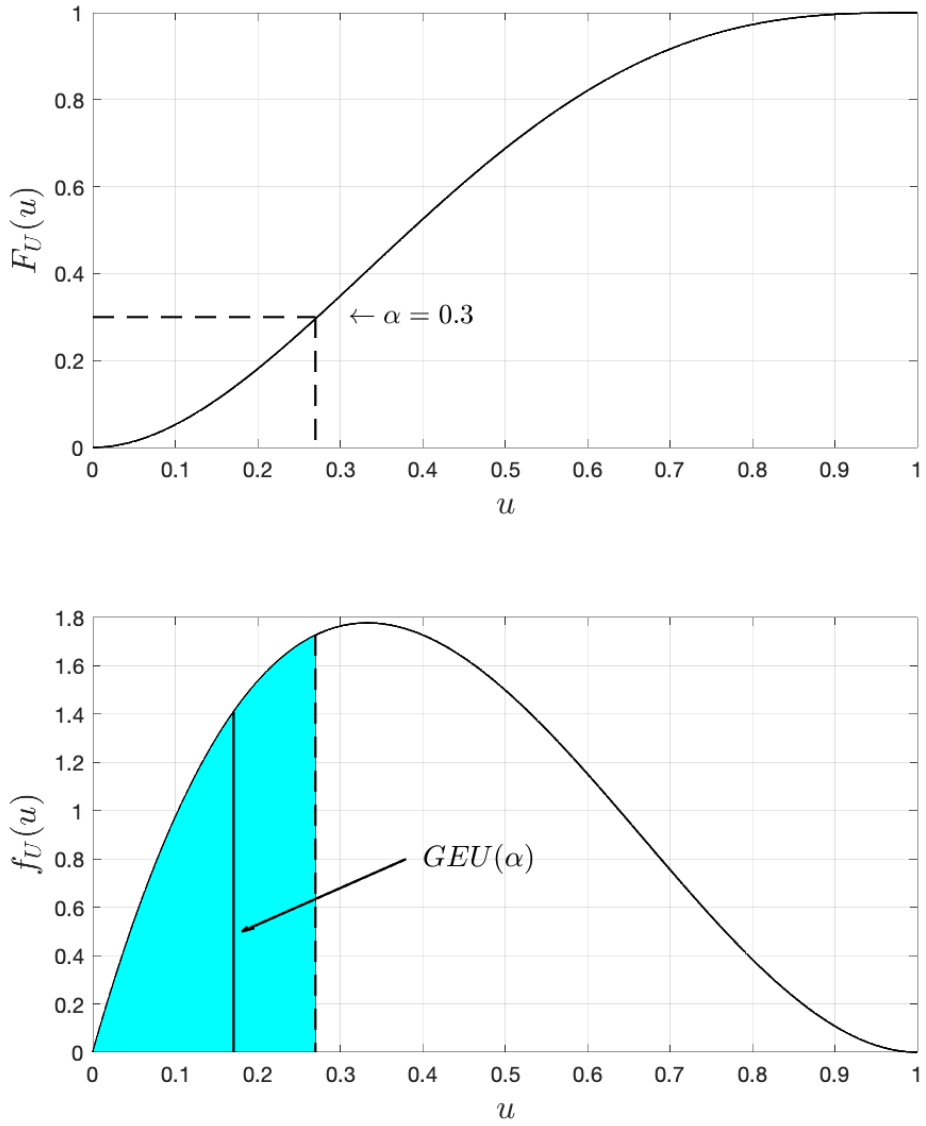


Figure 2.10: CDF of multi-attribute utility (Top) and PDF of multi-attribute utility (Bottom) where for example $F_U^{-1}(\alpha = 0.3) = 0.270$ and $GEU(\alpha = 0.3) = 0.171$.

Chapter 3

Performance-based Design Optimization Algorithm

3.1 Introduction

Optimization is the process for selecting the best element from some set of feasible alternatives. The optimization problem can be expressed in a standard form [13] as follows,

$$\begin{aligned} p^* &= \min_{\boldsymbol{x}} f_0(\boldsymbol{x}), \\ \text{s.t. } & f_i(\boldsymbol{x}) \leq 0 \quad \text{for } i = 1, 2, \ldots, m, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the design (decision) variable, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective (cost) function, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, m$, represent the constraints, and p^* is the optimal value. This form of the optimization problem can be adopted for the PPBD framework discussed in the previous chapters. Through the PPBD evaluation, we can evaluate the performances and finally obtain the GEU of the design alternatives. Thus, this design optimization problem can simply be expressed as follows,

$$\boldsymbol{d^*} = \underset{\boldsymbol{d} \in \mathcal{D}}{\operatorname{argmax}} \mathcal{U}_s(\boldsymbol{d}), \tag{3.2}$$

where $\boldsymbol{d}$ is a design variable, $\mathcal{D}$ is a domain of design set, and $\mathcal{U}_s : \mathcal{D} \to [0, 1]$ is the PPBD evaluation function considering certain site information $s \in \mathcal{S}$.

The entire process of the PPBD framework is very complicated, which makes the above optimization problem *non-convex*. In non-convex optimization problems, there exists local optima and iterative optimization algorithms like the *gradient descent method* are easily stuck in such local optima. Thus, we need to adopt optimization algorithms containing some randomness to search for the global optimum (at least have the possibilities to find a better local optimum point). In this chapter, two *meta-heuristic algorithms* are introduced for solving this complex function of the optimization problem.

3.2 Genetic Algorithm

Basic Terminology

The Genetic Algorithm (GA) is a computational model inspired by natural evolution. Therefore, most of the terminologies in a GA are from Biology. The following is a list of the basic terminologies commonly used in GAs [14]:

- Population of chromosomes (or initial population);
- Fitness function;
- Natural selection;
- Crossover to produce next generation;
- Random mutation of chromosomes.

The *chromosome* is one of the candidate solutions. In the framework of PDO, design alternatives are chromosomes (refer to Figure 3.1). The *fitness function*, which is from evolutionary theory, is the function that the algorithm is trying to optimize [41]. It corresponds to the objective or the cost function in the optimization problem. *Natural selection* is the criterion for selecting superior chromosomes over the others. In general, it is finding the maximum or the minimum value from the fitness function. *Crossover* and *random mutation* are common methods for generating offspring, which are discussed later in this chapter.

The GA begins with a population of chromosomes. In general, they are generated randomly. Each chromosome is evaluated and usually ranked by a fitness function, and subsequently the chromosomes are selected by natural selection where the fittest alternatives are selected for reproduction in order to produce offsprings of the next generation. Superior chromosomes which fit well the fitness function can get higher chance to reproduce and survive [14][41]. The selection criteria may differ from one problem to another, but eventually some chromosomes are chosen, and they form the next generation with a new population of chromosomes. Generation by generation, the population of chromosomes eventually become superior.

Crossover

In a GA, crossover (recombination) is a genetic operator for combining the genetic information from two parents to generate an offspring in the next generation. There are numerous types of crossover operators in GAs. In general, the engineer designs the problem with specific algorithms for their problems at hand. For example, the most popular crossover operator, namely *one-point crossover*, is introduced in this section (Figures 3.2 and 3.3). In one-point crossover, one arbitrary combination point is selected for both parents' chromosomes. Subsequently, the genetic information of the two parents up to the selected point are

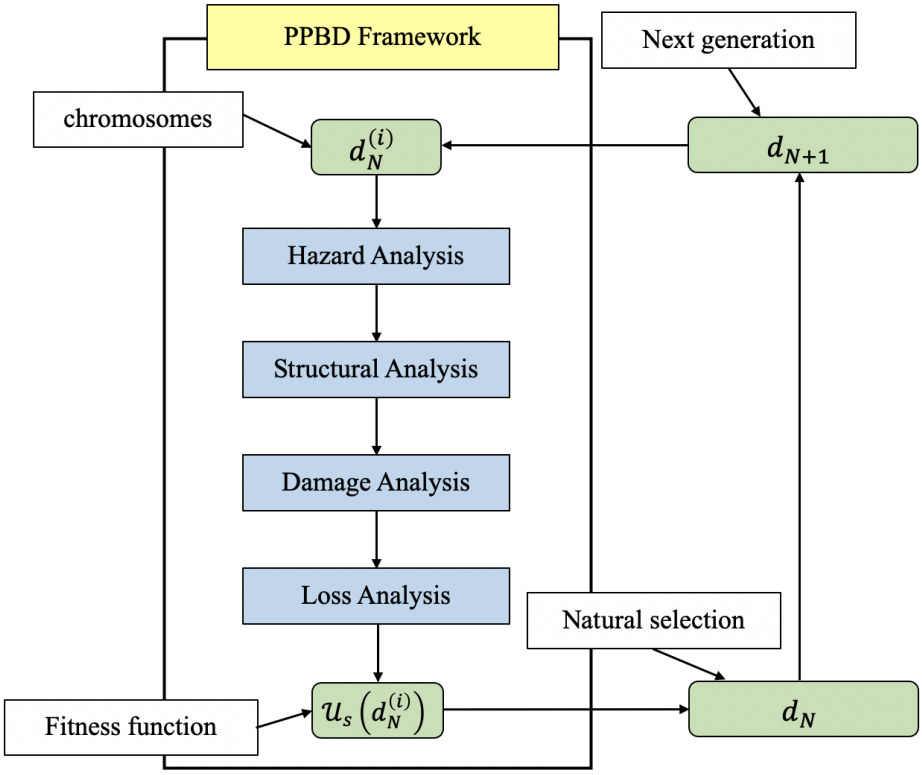


Figure 3.1: GA terminologies applied to the PPBD framework.

swapped to generate an offspring. When the data structure is a bit array, the result of the crossover is as shown in Figure 3.2. If the data set is real numbers or integers, the offspring can take the value between two parents (could be an average or some weighted averages), refer to Figure 3.3.

Random Mutation

Random mutation is a very intuitive and simple operator in the GAs. This method is advantageous when the solution is stuck at the local optimal point. The genetic information of the parent is randomly selected and randomly modified (e.g., by addition a constant $a \in [-1, 1]$, to generate mutants (Figure 3.4). Since this method significantly depends on how many information sets will be selected and how much they will be modified, the engineer should consider the characteristics of the problem at hand in designing the adopted random mutation algorithms.

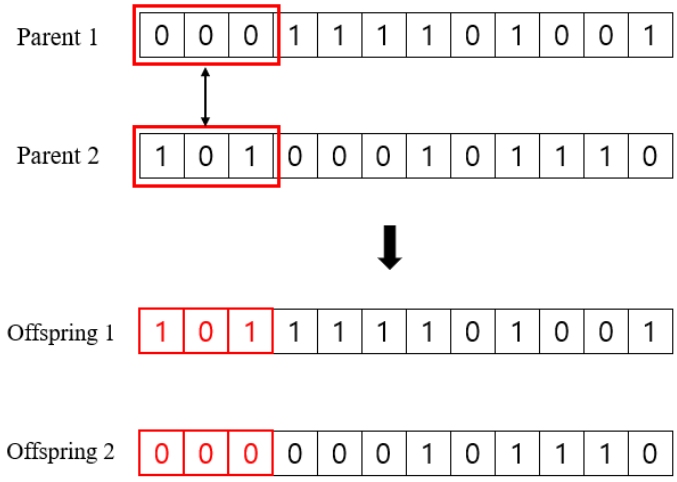


Figure 3.2: Crossover operator for a bit array.

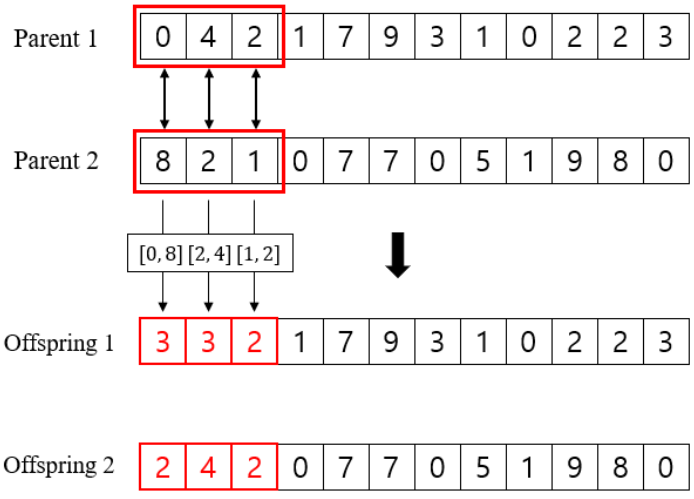


Figure 3.3: Crossover operator for an integer array.

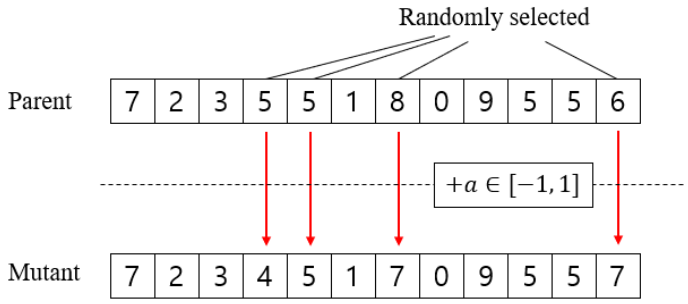


Figure 3.4: Random mutation.

3.3 Bayesian Optimization Algorithm

The Bayesian Optimization Algorithm (BOA) is for optimizing objective functions that require an extensive computational cost to evaluate. It is a well-known algorithm in the Machine Learning (ML) community for tuning hyper-parameters in ML algorithms, especially Deep Convolutional Neural Networks (DCNNs). However, BOA is also a powerful algorithm to solve many other black-box optimization problems of the following general form:

$$\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{3.3}$$

where $\mathcal{X}$ is a feasible set and $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ is the objective function that takes a long time to evaluate. In the BOA, the feasible set and objective function have the following typical properties [20]:

- The feasible set $\mathcal{X}$ is a simple set, e.g., a hyper-rectangle $\{\boldsymbol{x} \in \mathbb{R}^d : a_i \leq x_i \leq b_i, i = 1, 2, \ldots, d\}$ or the d-dimensional simplex $\{\boldsymbol{x} \in \mathbb{R}^d : \sum_{i=1}^{d} x_i = 1\}$.
- The dimension, d, of the input variable $\boldsymbol{x} \in \mathbb{R}^d$ is not too large ($d \leq 20$).
- The objective function f is continuous.
- The function f is a black-box function, i.e., we do not know its special structure (e.g., linearity or convexity), where such structure usually makes it easy to optimize using efficient optimization techniques.
- The function f requires a large computational cost which limits the number of evaluations we can perform, e.g., $N \leq 1000$.

The BOA consists of two main components, a *surrogate model* (Bayesian statistical model in this study) for modeling the objective function, and an *acquisition function* for determining the next sample point to evaluate. Algorithm 2 shows the general procedure of the BOA.

Algorithm 2: Pseudo-code for BOA.

Construct Gaussian process prior on f.
Select N random points, $\boldsymbol{x_i}$ for $i = 1, 2, \ldots, N$.
Evaluate $f(\boldsymbol{x_i})$ for $i = 1, 2, \ldots, N$.
$n \leftarrow N$
while $n \leq M$ **do**
  Update the Gaussian process posterior on f using all data.
  $f_n^* \leftarrow \max_{i=1,2,\ldots,n} f(\boldsymbol{x_i})$
  Choose most promising point for next evaluation using an Acquisition function α.
  $\boldsymbol{x_{next}} \leftarrow \operatorname{argmax}_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x}, f_n^*)$
  Evaluate $f(\boldsymbol{x_{next}})$
  $n \leftarrow n + 1$
Output: f_n^*

Gaussian Process Regression

In most of the BOAs, Gaussian Process Regression (GPR) model, which can make predictions incorporating prior knowledge (kernels) and provide uncertainty measures over predictions [32] [33], is used as surrogate models. Gaussian process defines a distribution over a function, $p(f)$, where the function is mapping the input space, $\mathcal{X}$, to $\mathbb{R}$.

> ***Definition of Gaussian Process:*** $p(f)$ is a Gaussian process (GP) if for any finite subset $\{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset has a multivariate Gaussian distribution.

The random variables that follows the multivariate Gaussian distribution in d-dimensional space with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$, i.e., $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, have a PDF as follows,

$$f_X(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right]. \tag{3.4}$$

GPs are parameterized by a mean function, $m(\cdot)$, and a covariance function, or kernel, $k(\cdot, \cdot)$. The mean function constructs a mean of the multivariate Gaussian distribution and the kernel constructs the covariance matrix as follows,

$$\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X'}) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}'_1) & k(\boldsymbol{x}_1, \boldsymbol{x}'_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}'_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}'_1) & k(\boldsymbol{x}_2, \boldsymbol{x}'_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}'_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}'_1) & k(\boldsymbol{x}_N, \boldsymbol{x}'_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}'_N) \end{bmatrix}. \tag{3.5}$$

The covariance matrix $\boldsymbol{\Sigma(x, x')}$ must be positive definite, i.e., $\boldsymbol{a}^T\Sigma(\boldsymbol{x}, \boldsymbol{x'})\boldsymbol{a} > 0, \forall \boldsymbol{a} \neq \boldsymbol{0}$. The Squared Exponential (SE) kernel function is commonly used and is defined as follows,

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x}') = \sigma \exp\left(-\frac{1}{2l^2}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right), \tag{3.6}$$

where σ and l are parameters of the squared exponential kernel function, and $\|\bullet\|$ is the Euclidean norm induced by the inner product of $\bullet$. Therefore, for any finite set of elements $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \in \mathcal{X}$ in the GP model with $m(\cdot)$ and $k(\cdot, \cdot)$, the corresponding finite set of random variables $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)$ have the following distribution,

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{x}_1) \\ m(\boldsymbol{x}_2) \\ \vdots \\ m(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right). \tag{3.7}$$

We denote this using the following compact notation,

$$f(\cdot) \sim \mathcal{GP}\left(m(\cdot), k(\cdot, \cdot)\right). \tag{3.8}$$

Let $D_T = \{\boldsymbol{X}, \boldsymbol{f}\}$ be the training data where $\boldsymbol{f}$ denotes the vector of training observations. From the observed data set, we have a GP prior as follows,

$$\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{X}), \boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X})\right). \tag{3.9}$$

For a test set $(\boldsymbol{x}_*, f_*)$, we have

$$f_*|\boldsymbol{x}_* \sim \mathcal{N}\left(m(\boldsymbol{x}_*), k(\boldsymbol{x}_*, \boldsymbol{x}_*)\right). \tag{3.10}$$

The joint distribution of $\boldsymbol{f}$ and f_* would be also multivariate normal distribution as follows,

$$\begin{bmatrix} \boldsymbol{f}|\boldsymbol{X} \\ f_*|\boldsymbol{x}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{X}) \\ m(\boldsymbol{x}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X}) & \boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{x}_*) \\ \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{X}) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right). \tag{3.11}$$

Then, the GP posterior is expressed as follows,

$$\begin{aligned} f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f} \sim &\mathcal{N}\left(m(\boldsymbol{x}_*) + \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{X})\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X})^{-1}(\boldsymbol{f} - \boldsymbol{\mu}(\boldsymbol{X})),\right. \\ &\left. k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{X})\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{x}_*)\right). \end{aligned} \tag{3.12}$$

Thus,

$$\mu_{f_*|\boldsymbol{X},\boldsymbol{f}}(\boldsymbol{x}_*) = m(\boldsymbol{x}_*) + \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{X})\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X})^{-1}(\boldsymbol{f} - \boldsymbol{\mu}(\boldsymbol{X})), \tag{3.13}$$

and

$$\sigma_{f_*|\boldsymbol{X},\boldsymbol{f}}(\boldsymbol{x}_*, \boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{X})\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{\Sigma}(\boldsymbol{X}, \boldsymbol{x}_*). \tag{3.14}$$

Acquisition Function

Acquisition functions are crucial to the BOA. It is the function used for choosing the most promising point for the next evaluation in the BOA as follows,

$$\boldsymbol{x}_{next} = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}}\, \alpha(\boldsymbol{x}). \tag{3.15}$$

There are a variety of options to adopt and each acquisition function has its own strength and weakness, and the choice of the function depends on the specific problem being optimized. Three acquisition functions are widely used, namely Probability of Improvement (PI), Expected Improvement (EI), and Upper Confidence Bound (UCB). These three functions are introduced in this section.

Probability of Improvement

PI acquisition fuction chooses the next point as the one which has the highest *probability of improvement* over the current maximum, $f_{max} = \max_{i=1,2,\ldots,n} f(\boldsymbol{x}_i)$:

$$\alpha_{PI}(\boldsymbol{x}) = P(f(\boldsymbol{x}) \geq f_{max}) \tag{3.16}$$

Thus,

$$\alpha_{PI}(\boldsymbol{x}) = \int_{f_{max}}^{\infty} \mathcal{N}\left(f; \mu(\boldsymbol{x}), \sigma(\boldsymbol{x}, \boldsymbol{x})\right) df = 1 - \Phi(f_{max}; \mu(\boldsymbol{x}), \sigma(\boldsymbol{x}, \boldsymbol{x})). \tag{3.17}$$

where Φ is the CDF of the standard normal distribution.

Expected Improvement

EI acquisition function is defined as follows,

$$\alpha_{EI}(\boldsymbol{x}) = \mathbb{E}[\max(0, f(\boldsymbol{x}) - f_{max})]. \tag{3.18}$$

Upper Confidence Bound

UCB acquisition function is typically known as GP-UCB and is defined as follows,

$$\alpha_{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) - \beta\sigma(\boldsymbol{x}), \tag{3.19}$$

where $\beta > 0$ is a tradeoff parameter and $\sigma(\boldsymbol{x})$ is the marginal standard deviation of $f(\boldsymbol{x})$.

3.4 Proposed Performance-based Design Optimization Algorithm

In this section, we propose the Performance-based Design Optimization (PDO) algorithm using the two meta-heuristic algorithms, introduced in the previous two sections, as shown in Figures 3.5 and 3.6. The PDO aims towards solving the design optimization problem expressed by Eq. 3.2. The proposed PDO is illustrated using Figures 3.5 and 3.6 for the PDO-GA (PDO using GA) and PD-BOA (PDO using BOA), respectively.

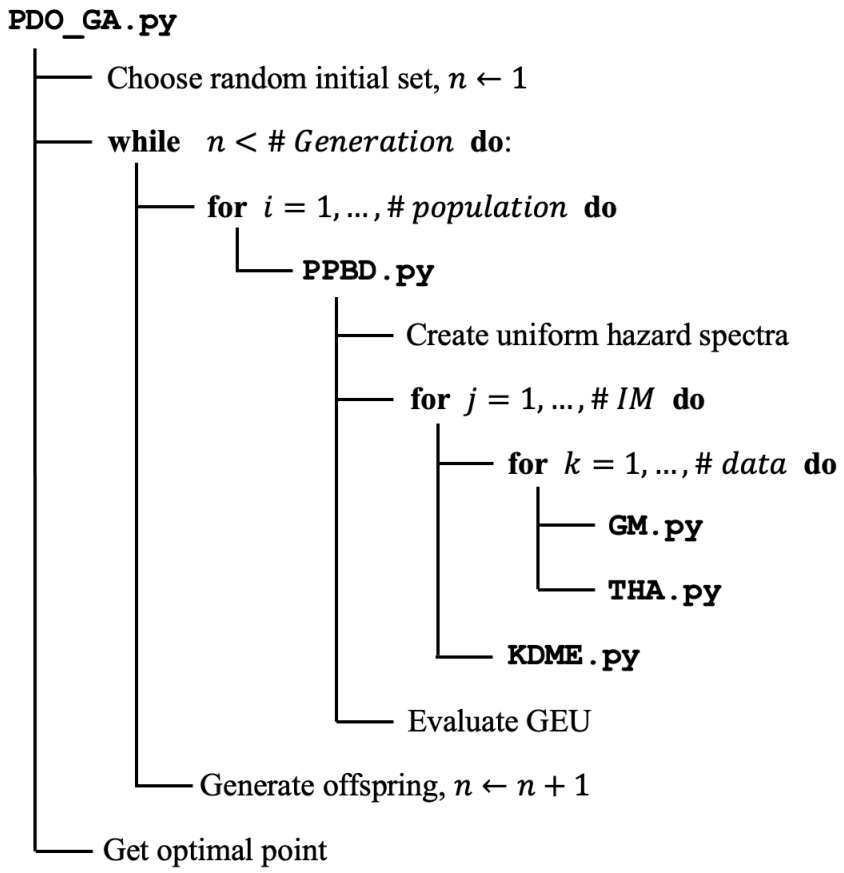


Figure 3.5: Sketch of the PDO-GA.

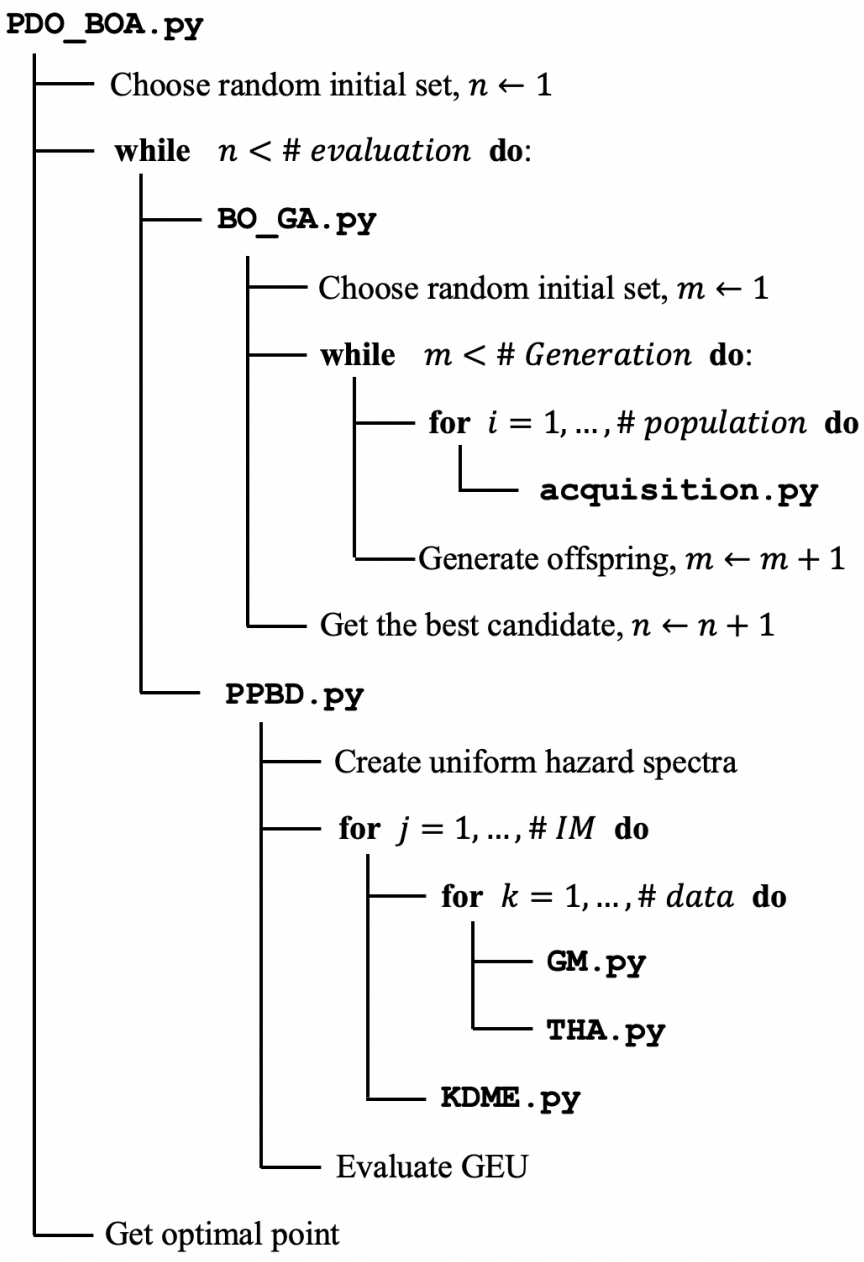


Figure 3.6: Sketch of the PD-BOA.

Chapter 4

Applications of the PBD Optimization Framework

4.1 Steel MRF Models

Two hypothetical steel Moment Resisting Frame (MRF) buildings (three-bay, five-story and three-bay, nine-story building frames) located in Berkeley, California are used as application examples. The design configurations and variables of these two frames are discussed herein.

The overall height of the three-bay, five-story building is 75 ft from the base, and each story has equal height (15 ft). Spacing of the columns (beam span) is 20 ft, refer to Figure 4.1. Likewise, the three-bay, nine-story building has equal story height (15 ft) with identical beam span 20 ft, refer to Figure 4.2.

All elements of the steel MRF buildings are wide flange steel beams meeting ASTM standard A36 [8] with a yield stress of 36 ksi. The section properties of the beams and columns are chosen from among 261 wide beam sections in the AISC wide flange beam library (Table A.1 in Appendix A) which creates $261^9 \approx 5.62 \times 10^{21}$ possible design alternatives, $\boldsymbol{d}$, for each of the MRF buildings (note that each building has 9 design variables, as indicated in Figure 4.1 and 4.2). The simulation models are developed in OpenSeesPy [47]. Columns are modeled as nonlinear beam-column elements with 5 integration points. Detailed information of the simulation models are described in `THA.py` (Section B.1 in Appendix B), which constructs the models and runs the nonlinear time-history analyses.

4.2 PPBD Evaluation

The location of the hypothetical steel MRF buildings are selected to be 2150 Shattuck Ave., Berkeley, California ($37.87°, -122.27°$). The Hayward Fault is located about 3 km east of the site, and the San Andreas Fault is located about 30 km west of this site. Soil condition is assumed as stiff soil corresponding to ASCE 7 Site Class D [7]. We can obtain the site-specific hazard curves and uniform hazard spectra (Figure 4.3) by using the USGS Unified

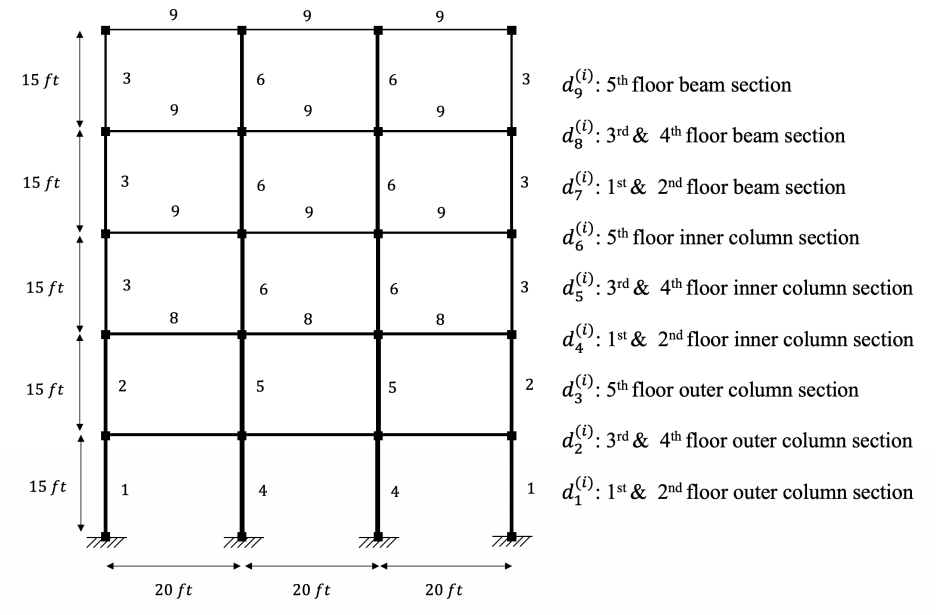


Figure 4.1: 3-bay, 5-story steel MRF model configuration and design variables.

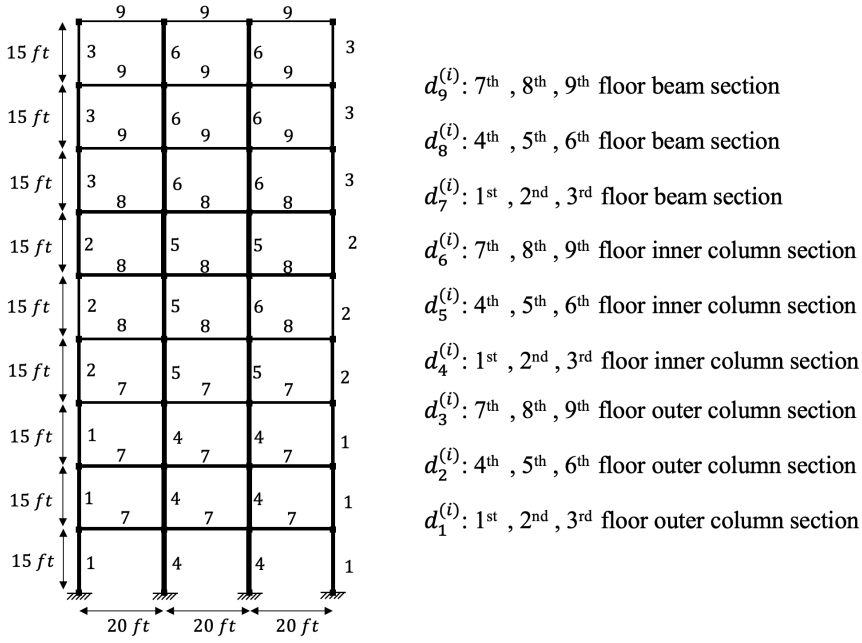


Figure 4.2: 3-bay, 9-story steel MRF model configuration and design variables.

Hazard Tool[4] or directly from the Seismic Performance Prediction Program (SP3)[5] [22]. The selected 11 hazard levels are defined in Table 4.1.

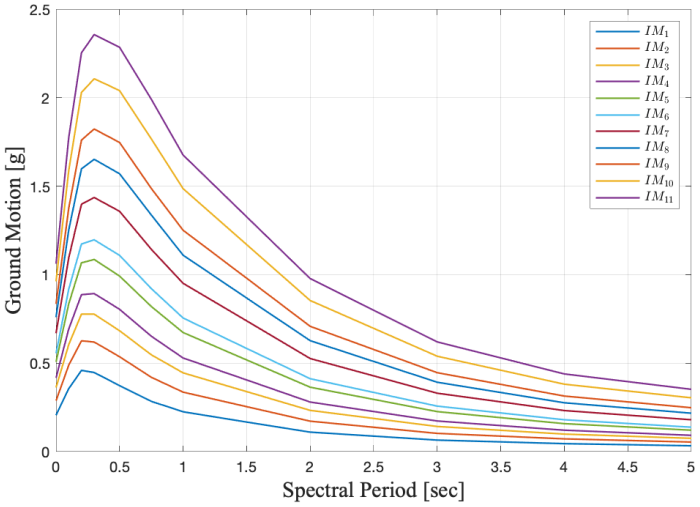


Figure 4.3: Selected 11 uniform hazard spectra for the considered Berkeley site.

Table 4.1: Return periods of the selected 11 IMs.

$IM\#$	Hazard level	Annual rate of exceedance	Return period
1	50% in 30 years	2.284×10^{-2}	44 years
2	50% in 50 years	1.377×10^{-2}	73 years
3	50% in 75 years	9.199×10^{-3}	109 years
4	50% in 100 years	6.908×10^{-3}	145 years
5	20% in 50 years	4.453×10^{-3}	225 years
6	10% in 30 years	3.506×10^{-3}	285 years
7	10% in 50 years	2.105×10^{-3}	475 years
8	10% in 75 years	1.404×10^{-3}	712 years
9	5% in 50 years	1.025×10^{-3}	975 years
10	3% in 50 years	6.090×10^{-4}	1,642 years
11	2% in 50 years	4.040×10^{-4}	2,475 years

In these two steel MRF application examples, the quasi-stationary model is used with the modulating function $\phi_J(t)$ of Jennings et al. [25]. Assume that the time observing window is $t_s = 30$ sec with $\Delta t = 0.02$ sec, and maximum frequency $\omega_{max} = 2\pi/(1/75) = 471.24$

[4]Source: USGS Unified Hazard Tool, `https://earthquake.usgs.gov/hazards/interactive/`.
[5]`https://hbrisk.com/sp3`.

rad/sec with $\Delta\omega = 2\pi/t_s = 0.209$ rad/sec. `GM.py` (Section B.2 in Appendix B) generates the artificial ground motions compatible with the 11 uniform hazard response spectra (Figure 4.3) as shown in Figure 4.4. The code takes 130 sec to generate 1,000 artificial ground motions. `THA.py` (Section B.1 in Appendix B) performs the nonlinear time-history analyses with the generated ground motions. The code takes 895 sec for the 1,000 nonlinear time-history analyses of the 3-bay, 5-story steel MRF model and 1,730 sec for the 1,000 nonlinear time-history analyses of the 3-bay, 9-story steel MRF model (performed by an Intel Core i7-8650U computer with 8 GB RAM).

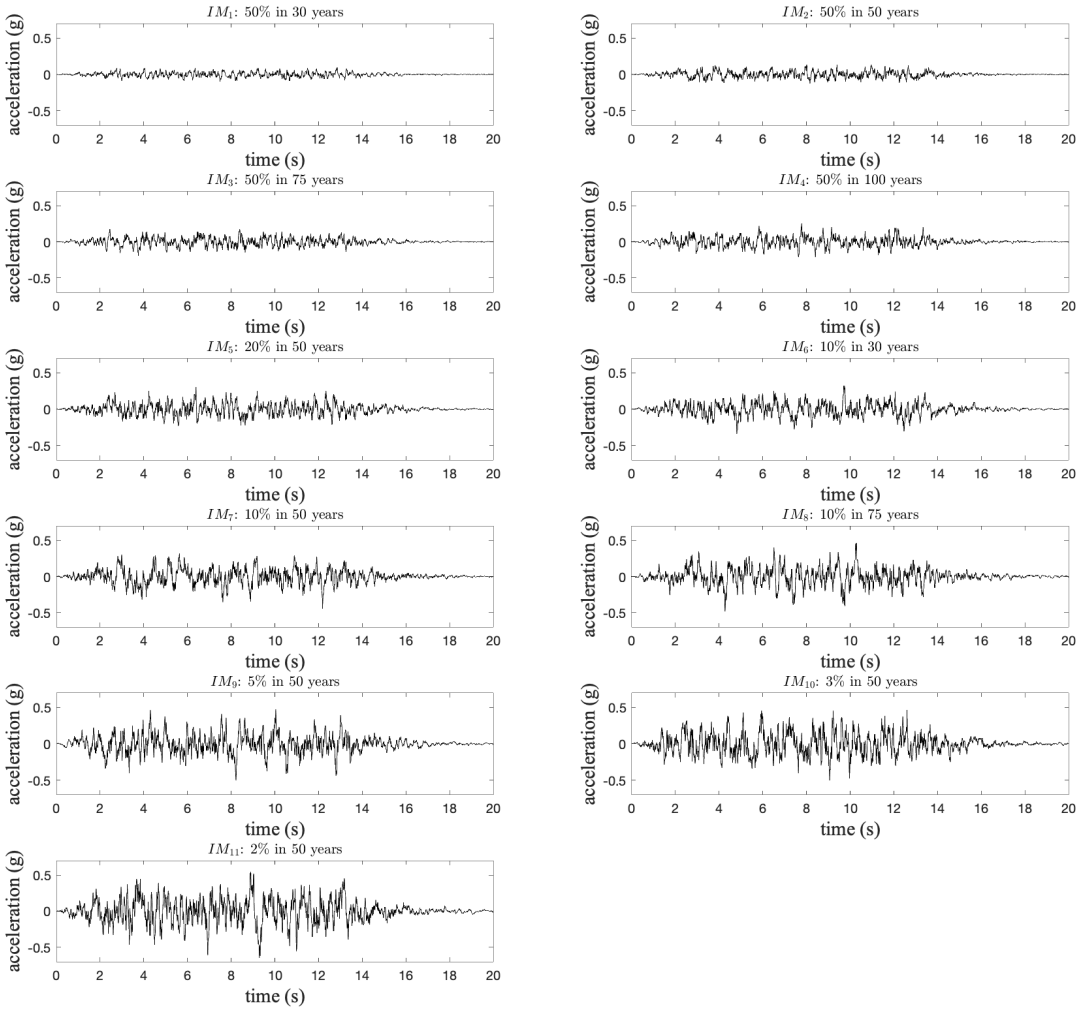


Figure 4.4: Artificial ground motions compatible with the selected 11 uniform hazard response spectra.

In the two application examples, we perform 30 structural analyses for each IM to construct the conditional probability distribution of the EDP given each IM (Total of 330 analyses are required in the PPBD evaluation framework for each of the application examples). Subsequently, we construct the least biased PDF from the 30 datasets by KDMEM.

The fragility curves in Figure 4.5 are used to define four Damage States, DS1, DS2, DS3, and DS4 corresponding to no, minor, moderate, and major damages, respectively, based on the inter-story drift ratios. Each fragility curve follows a lognormal CDF with the lognormal distribution parameters $\lambda_{DS2} = 0$, $\lambda_{DS3} = 0.6931$, $\lambda_{DS4} = 1.099$, and $\zeta = 0.30$.

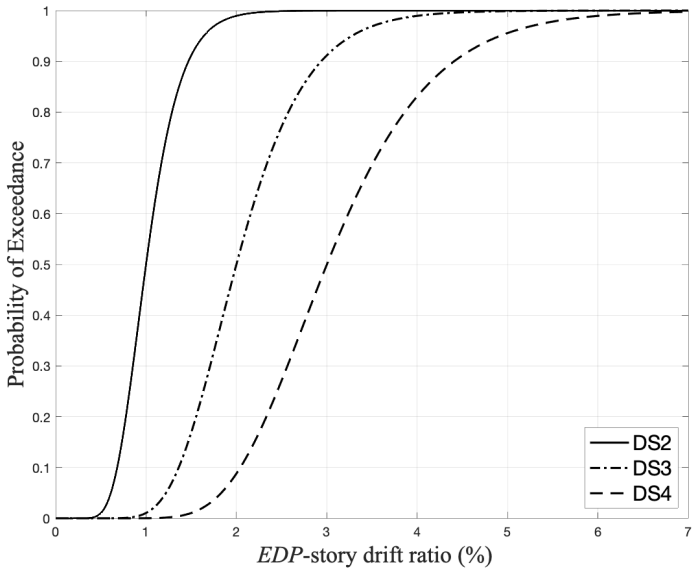


Figure 4.5: Fragility curves defining four damage states by the story drift ratio.

The loss curve for repair cost is generated by Tables 4.2 and 4.3, which are reproduced from [44]. First, we can define a unit cost function of the j-th item (total of 17 items in these MRF applications), $c_j(q)$ as a function of the quantity of this item q, as follows,

$$c_j(q) = \begin{cases} c_{j,max}, & \text{if } q < q_{j,min}, \\ c_{j,max} - \dfrac{c_{j,max} - c_{j,min}}{q_{j,max} - q_{j,min}}(q - q_{j,min}), & \text{if } q_{j,min} \le q < q_{j,max}, \\ c_{j,min}, & \text{if } q_{j,max} \le q. \end{cases} \tag{4.1}$$

The information about $q_{j,min}$, $c_{j,max}$, $q_{j,max}$ & $c_{j,min}$, $\forall\, j = 1, 2, \ldots, 17$ is listed in Table 4.3.

The deterministic total Repair Cost (RC) is as follows,

$$RC = \sum_{j=1}^{17} \left(\sum_{i=1}^{4} c_j(q_{ij})\; q_{ij}\; p(DS_i) \right), \tag{4.2}$$

Table 4.2: Damage-associated repair quantities for the performance groups (Note: MEP stands for Mechanical, Electrical, & Pluming.).

Repair items	Units	DS1	DS2	DS3	DS4
Demolition/Access					
1) Finish protection	ft^2	0	6,000	6,000	6,000
2) Ceiling system removal	ft^2	0	2,000	3,000	5,000
3) Drywall assembly removal	ft^2	0	800	800	6,000
4) Miscellaneous MEP	locations	0	2	4	6
5) Remove exterior skin (salvage)	ft^2	0	0	0	5,600
Repair					
6) Welding protection	ft^2	0	1,500	1,500	1,500
7) Shore beams below and remove	locations	0	0	0	12
8) Cut slab at damaged connection	ft^2	0	70	150	1,600
9) Carbon arc out weld	ft	0	40	50	50
10) Remove damaged element	ft^2	0	0	100	100
11) Replace weld-from above	ft	0	40	40	40
12) Remove/replace connection	lb	0	0	0	3,000
13) Replace slab	ft^2	0	70	70	1,600
Put-back					
14) Misc. MEP and clean-up	locations	0	2	4	6
15) Wall frame (studs and drywall)	ft^2	0	800	800	6,000
16) Replace exterior skin (salvage)	ft^2	0	0	0	5,600
17) Ceiling system	ft^2	0	2,000	3,000	5,000

where q_{ij} is the quantity of the j-th item when the damage state is DS_i. The information about q_{ij} for $i = 1, 2, \ldots, 4$ and $j = 1, 2, \ldots, 17$ is listed in Table 4.2.

The uncertainties in the unit repair costs as schematically illustrated in Figure 2.6 are considered by adopting cost multipliers for each item as random variables, X_j, $j = 1, 2, \ldots, 17$. Table 4.4 describes the information about X_j, $j = 1, 2, \ldots, 17$. Accordingly, RC is as follows,

$$RC = \sum_{j=1}^{17} \left(X_j \sum_{i=1}^{4} c_j(q_{ij})\ q_{ij}\ p(DS_i) \right) = \sum_{j=1}^{17} a_j X_j, \tag{4.3}$$

where $a_j = \sum_{i=1}^{4} c_j(q_{ij})\ q_{ij}\ p(DS_i)$.

Since the PMF of the DS, $p(DS_i)$, is determined by the design, $\boldsymbol{d} \in \mathcal{D}$, it is a function of $\boldsymbol{d}$, i.e., $p(DS_i|\boldsymbol{d})$. Thus, a_j is also a function of $\boldsymbol{d}$, i.e., $a_j(\boldsymbol{d}) = \sum_{i=1}^{4} c_j(q_{ij})\ q_{ij}\ p(DS_i|\boldsymbol{d})$. Moreover, $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{1}, \boldsymbol{\Sigma})$ where $\boldsymbol{1} \in \mathbb{R}^{17}$ is a vector whose elements are all ones and $\boldsymbol{\Sigma}$ is a

Table 4.3: Parameters for unit repair cost function (Note: MEP stands for Mechanical, Electrical, & Pluming.).

Repair items	Units	$q_{j,min}$	$c_{j,max}$	$q_{j,max}$	$c_{j,min}$
Demolition/Access					
1) Finish protection	ft^2	1,000	\$ 0.3	40,000	\$ 0.15
2) Ceiling system removal	ft^2	1,000	\$ 2	10,000	\$ 1.25
3) Drywall assembly removal	ft^2	1,000	\$ 2.5	20,000	\$ 1.5
4) Miscellaneous MEP	locations	6	\$ 200	24	\$ 150
5) Remove exterior skin (salvage)	ft^2	3,000	\$ 30	10,000	\$ 25
Repair					
6) Welding protection	ft^2	1,000	\$ 1.5	10,000	\$ 1
7) Shore beams below and remove	locations	6	\$ 2,100	24	\$ 1,600
8) Cut slab at damaged connection	ft^2	10	\$ 200	100	\$ 150
9) Carbon arc out weld	ft	100	\$ 15	1,000	\$ 10
10) Remove damaged element	ft^2	100	\$ 80	2,000	\$ 50
11) Replace weld-from above	ft	100	\$ 50	1,000	\$ 40
12) Remove/replace connection	lb	2,000	\$ 6	20,000	\$ 5
13) Replace slab	ft^2	100	\$ 20	1,000	\$ 16
Put-back					
14) Misc. MEP and clean-up	locations	6	\$ 300	24	\$ 200
15) Wall frame (studs and drywall)	ft^2	100	\$ 12	1,000	\$ 8
16) Replace exterior skin (salvage)	ft^2	1,000	\$ 35	10,000	\$ 30
17) Ceiling system	ft^2	100	\$ 8	60,000	\$ 5

Table 4.4: Random variables for cost items.

Random variable #	Random variable	Distribution Type	Mean	Coefficient of variation (δ)	Correlation coefficients
$1 \sim 5$	Multipliers for demolition/access items	Normal	1.0	0.05	$\rho_{ij} = 0.2$, $i \neq j$ $i, j = 1 \sim 5$
$6 \sim 13$	Multipliers for repair items	Normal	1.0	0.1	$\rho_{ij} = 0.3$, $i \neq j$ $i, j = 6 \sim 13$
$14 \sim 17$	Multipliers for put-back items	Normal	1.0	0.05	$\rho_{ij} = 0.2$, $i \neq j$ $i, j = 14 \sim 17$

covariance matrix of $\boldsymbol{X}$. Therefore,

$$RC(\boldsymbol{d}) = \sum_{j=1}^{17} a_j(\boldsymbol{d}) X_j \sim \mathcal{N}\left(\sum_{j=1}^{17} a_j(\boldsymbol{d}), \sum_{j=1}^{17} a_j(\boldsymbol{d})^2 \Sigma_{jj} + 2 \sum_{j=2}^{17} \sum_{k=1}^{j-1} a_j(\boldsymbol{d}) a_k(\boldsymbol{d}) \Sigma_{jk} \right). \quad (4.4)$$

Other performances, CO2 emission, CO_2, and Construction Cost, CC, are also considered in this optimization problem. They are modeled with random variables as in Table 4.5[6] [30].

$$CO_2(\boldsymbol{d}) \sim \mathcal{N}\left(W(\boldsymbol{d}) \mu_{X_{18}}, \left(W(\boldsymbol{d})\right)^2 \sigma_{X_{18}}^2 \right), \quad (4.5)$$

and

$$CC(\boldsymbol{d}) \sim \mathcal{N}\left(W(\boldsymbol{d}) \mu_{X_{19}}, \left(W(\boldsymbol{d})\right)^2 \sigma_{X_{19}}^2 \right), \quad (4.6)$$

where $W(\boldsymbol{d}) = \sum_{i=1}^{\#ofelements} L_i \lambda_i(\boldsymbol{d})$ is the total weight of the steel structure, L_i is the length of i-th steel structural element, $\mu_{X_{18}}$ and $\mu_{X_{19}}$ are the mean values of the unit CO_2 emission and of the unit cost of A36 steel, respectively, and $\sigma_{X_{18}} = \mu_{X_{18}} \delta_{X_{18}} = 0.185$ (ton/ton) and $\sigma_{X_{19}} = \mu_{X_{19}} \delta_{X_{19}} = 55$ ($/ton) are the standard deviation values of the unit CO_2 emission and of the unit cost of A36 steel.

Table 4.5: Random variables for CO2 emission and construction cost.

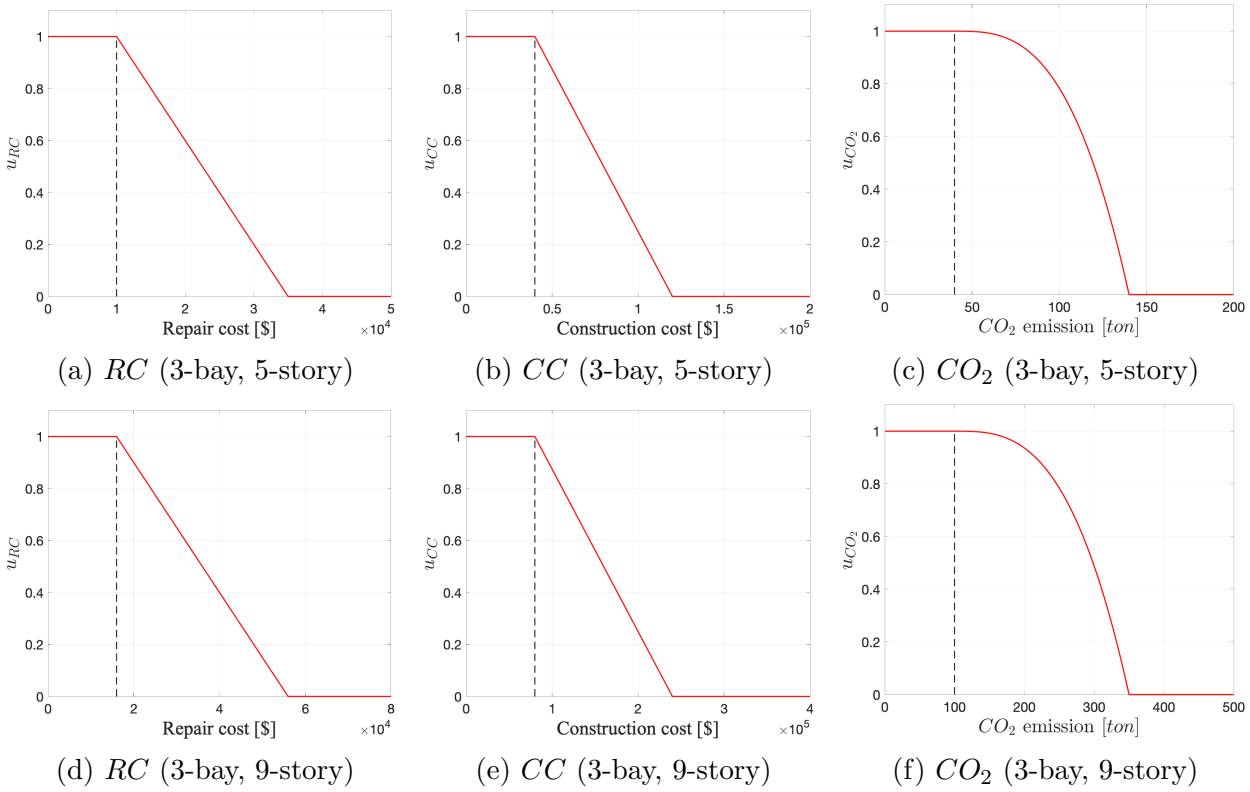
Random variable #	Random variable	Distribution Type	Mean	Coefficient of variation (δ)	Correlation coefficients
18	Unit CO2 emission (ton/ton)	Normal	1.85	0.1	Independent
19	Unit cost of A36 steel ($/ton)	Normal	1,100	0.05	Independent

Single attribute utility functions for repair cost, construction cost and CO2 emission, refer to Figure 4.6, are used. For the CO2 emission, a risk-averse utility function is adopted. On the other hand, for the repair and construction costs, risk-neutral utility functions are utilized. The weights $w_{RC} = 0.4$, $w_{CC} = 0.2$, and $w_{CO_2} = 0.4$ are selected for the total utility, $u = w_{RC} u_{RC} + w_{CO_2} u_{CO_2} + w_{CC} u_{CC}$ in this application example.

4.3 Optimization Algorithms

In the two steel MRF applications, the two structural models have the same design set, $\mathcal{D} = \{1, \ldots, 261\}$[9]. Since $\mathcal{D}$ is in a discrete space, it is difficult to apply the meta-heuristic

[6]Decarbonization challenge for steel: `https://www.mckinsey.com/industries/metals-and-mining/our-insights/decarbonization-challenge-for-steel`.



(a) RC (3-bay, 5-story) (b) CC (3-bay, 5-story) (c) CO_2 (3-bay, 5-story)

(d) RC (3-bay, 9-story) (e) CC (3-bay, 9-story) (f) CO_2 (3-bay, 9-story)

Figure 4.6: Single attribute utility functions for the steel MRF buildings.

algorithms, especially the BOA. Therefore, we need to convert $\mathcal{D}$ from discrete to continuous. Each section ID represents several section properties (cross-sectional area, moment of inertia, depth, and nominal weight), which are continuous properties having units of in^2, in^4, in, and lb/ft, respectively. Although the name of a section ID is based on its depth and nominal weight, the cross-sectional area and moment of inertia usually represent the section better, especially when its shape is defined, which is the case here using wide flange steel beams meeting ASTM standard A36. As shown in Figure 4.7, we can plot the entire design set into continuous (cross-sectional area, A, moment of inertia, I)-space, and all section IDs are subsequently located in the following triangular domain,

$$\mathcal{C} = \left\{ (A, I) \,\middle|\, \frac{I_{max}}{A_{max}} A \geq I, A \leq A_{max}, I \geq 0 \right\}, \tag{4.7}$$

where A_{max} and I_{max} are taken as 120 in^2 and 37,500 in^4, respectively. Moreover, all section IDs can be plotted on the dimensionless space and they are in the following domain,

$$\mathcal{C}_0 = \left\{ (x, y) | x \geq y, x \leq 1, y \geq 0 \right\}. \tag{4.8}$$

Thus, the discrete design set can be converted to a new continuous domain, as follows,

$$\mathcal{C}_0^9 = \left\{(x_1, y_1) \times \cdots \times (x_9, y_9) | x_i \geq y_i, x_i \leq 1, y_i \geq 0, i = 1, \ldots, 9\right\}. \tag{4.9}$$

This continuous domain is only used in searching for the next candidates, i.e., we are searching the best point based on the acquisition function (maximizing the acquisition function) with $\boldsymbol{x} \in \mathcal{C}_0^9$, but evaluating $\boldsymbol{d} \in \mathcal{D}$. If there does not exist $\boldsymbol{d} \in \mathcal{D}$ corresponding to $\boldsymbol{x_{next}} = \underset{\boldsymbol{x} \in \mathcal{C}_0}{\operatorname{argmax}}\, \alpha_{PI}(\boldsymbol{x})$ (in most cases, there does not), $\boldsymbol{d_{next}}$ is taken as the closest point (by the Euclidean norm) to $\boldsymbol{x_{next}}$ in the dimensionless domain shown in the right plot of Figure 4.7.

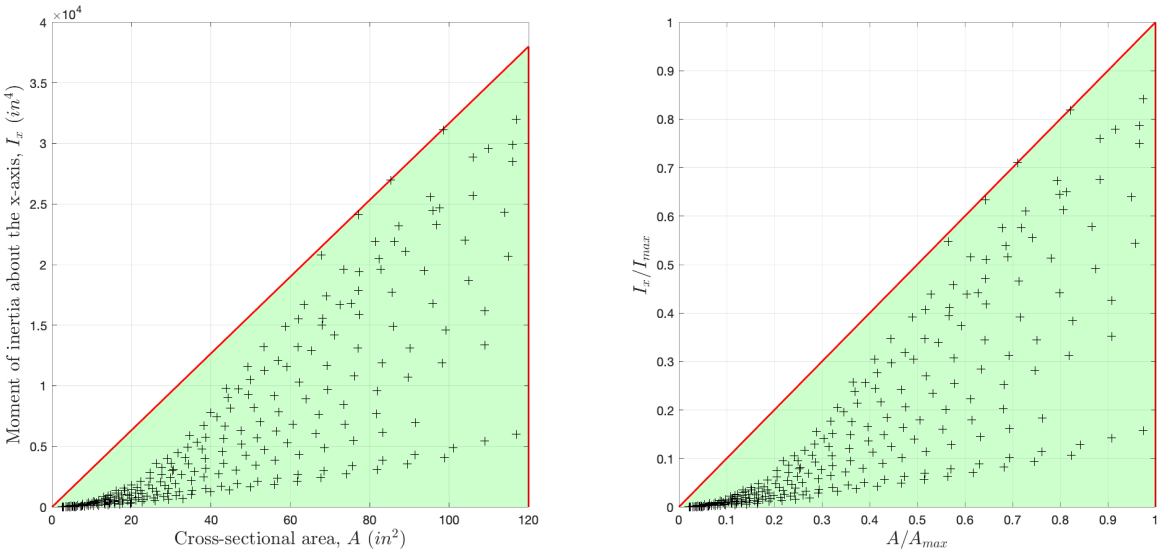


Figure 4.7: Design space converted to cross-sectional area, A, vs. moment of inertia about the x axis, I_x, (left) and in a dimensionless space (right).

Finally, we optimize the $GEU(\alpha)$ with $\alpha = 0.3$ until our optimal design alternative's GEU reaches to the target GEU, $GEU_{target} = 0.5$. Table 4.6 shows the parameters of the GA used in the two steel MRF applications for the PDO and Table 4.7 shows the parameters of the GA used for optimizing the PI acquisition function in the BOA, refer to the results discussion in the next section.

Table 4.6: Parameters of the Genetic Algorithm for PDO.

Population size	Crossover rate	Random Mutation rate
50	0.6	0.05

Table 4.7: Parameters of the Genetic Algorithm to optimize the PI acquisition function.

Population size	Crossover rate	Random Mutation rate
100	0.7	0.1

4.4 Results

For the 3-bay, 5-story steel MRF model, BOA evaluated 186 points and took 22.45 hours, and GA evaluated 236 points and took 20.33 hours to reach the target GEU, $GEU_{target} = 0.5$. BOA required fewer evaluations but more computational cost than GA. On the other hand, for the 3-bay, 9-story steel MRF model, BOA evaluated 192 points and took 36.68 hours, and GA evaluated 262 points and took 44.83 hours to reach the same target GEU. In this case, BOA required fewer evaluations and less computational cost than GA.

Since the acquisition functions are frequently non-trivial to optimize [42], maximizing the acquisition functions requires non-convex optimization procedures. In the present applications, GA is adopted to maximize the PI acquisition function to determine the next candidate point. PDO using BOA performed better in terms of the number of evaluations and searching heuristics than GA but it costs some computational time to search the next candidates, which could take a large portion of the total computational time.

The considered two application models have identical searching space in this study. Therefore, searching costs for both models are not very different. However, the PPBD evaluation process, which requires nonlinear time-history analyses, of the 3-bay, 9-story steel MRF model obviously costs more than that of the 3-bay, 5-story steel MRF model due to the larger size (almost double) of the former model than the latter. This led to about twice the computational time for the nonlinear time-history analyses of the 9-story frame compared to the 5-story frame.

The results are summarized in Table 4.8 and Figure 4.8 for the 3-bay, 5-story steel MRF building and in Table 4.9 and Figure 4.9 for the 3-bay, 9-story steel MRF building. These results demonstrate that the proposed PDO using BOA performs better (in terms of number of evaluations and computational time) for the 3-bay, 9-story steel MRF building with a relatively smaller number of variables (in this study, only 9 design variables d_1 to d_9 are considered, as shown in Figures 4.1 and 4.2) than using GA. On the other hand, comparable results are obtained of the two approaches of PDO-GA and PD-BOA for the 3-bay, 5-story steel MRF building. Detailed concluding remarks related to these applications are listed in Chapter 6 together with planned future extensions.

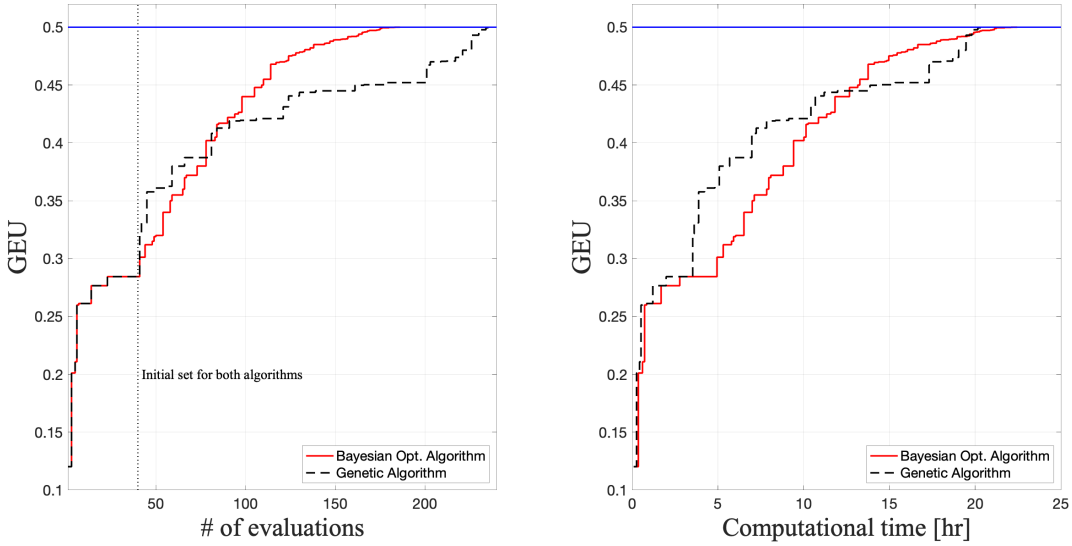


Figure 4.8: Optimization results of the 3-bay, 5-story steel MRF, the number of evaluations vs. GEU (left) and vs. computational time (right).

Table 4.8: Optimal design of the 3-bay, 5-story steel MRF via BOA and GA.

Design variable	Bayesian Optimization	Genetic Algorithm
d_1	W36×282	W36×330
d_2	W27×194	W27×217
d_3	W24×103	W21×83
d_4	W40×324	W36×487
d_5	W36×210	W33×201
d_6	W21×166	W24×162
d_7	W18×130	W18×158
d_8	W18×119	W14×120
d_9	W14×61	W14×90
GEU	0.5005	0.5017
# of evaluations	186	236
Computational time (hr)	22.45	20.33

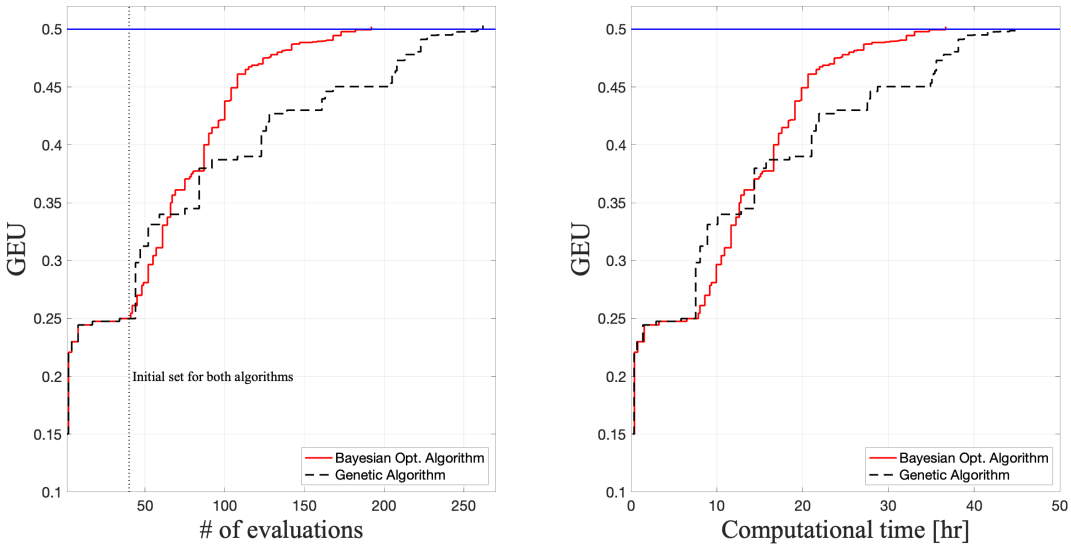


Figure 4.9: Optimization results of the 3-bay, 9-story steel MRF, the number of evaluations vs. GEU (left) and vs. computational time (right).

Table 4.9: Optimal design of the 3-bay, 9-story steel MRF via BOA and GA.

Design variable	Bayesian Optimization	Genetic Algorithm
d_1	W40×372	W40×362
d_2	W30×191	W33×201
d_3	W21×83	W24×84
d_4	W40×397	W40×372
d_5	W33×263	W33×291
d_6	W21×166	W24×162
d_7	W18×130	W18×143
d_8	W18×65	W18×60
d_9	W12×40	W14×53
GEU	0.5022	0.5032
# of evaluations	192	262
Computational time (hr)	36.68	44.83

Chapter 5

Structural Model Updating

5.1 Finite Element Model Updating Framework

Developing an accurate digital twin (numerical model) of an existing physical structure is very important for simulation, damage detection, and overall evaluation (e.g., reconnaissance efforts following major extreme events, e.g., earthquakes) of the structure [10] [34]. However, complete information about the existing structure is usually not available; thus, various assumptions on the numerical model, such as geometrical dimensions and material properties, are required. This causes discrepancies between the structural responses from the numerical model and the actually measured structural responses from an instrumented structural in laboratory of field settings. Structural Model Updating (SMU) is used to reduce or even close this gap between the prediction and the ground truth and its process can be expressed as the following optimization problem:

$$\boldsymbol{\theta}^{*} = \underset{\boldsymbol{\theta} \in \mathcal{X}}{\operatorname{argmin}} \, f(y(\boldsymbol{\theta}), y^{*}), \tag{5.1}$$

where $\boldsymbol{\theta}$ is a vector of the numerical model parameters, $\mathcal{X}$ is a feasible set of the numerical model parameters, and f is the objective function representing the gap between the structural responses from the numerical model, $y(\boldsymbol{\theta})$, and the measured structural responses of the existing structure, y^{*} (e.g., the mean squared error function).

Finite Element (FE) models are normally used to evaluate structural responses in the SMU framework and the general architecture of this framework is as shown in Figure 5.1 [26]. In this chapter, the SMU framework for complex and large structures is proposed, and sophisticated FE Analysis (FEA) software, such as ABAQUS or OpenSees, is adapted in this framework to predict the response of such structures. The meta-heuristic algorithms introduced in the Chapter 3 are used in this framework to solve the optimization problem in Eq. 5.1, which is generally a non-convex problem. The proposed SMU framework codes are developed in Python, and the codes (Appendix C) actively interact with the FEA software in the Python environment by modifying the input parameters of the FE numerical model and reading the FEA results (Figure 5.2).

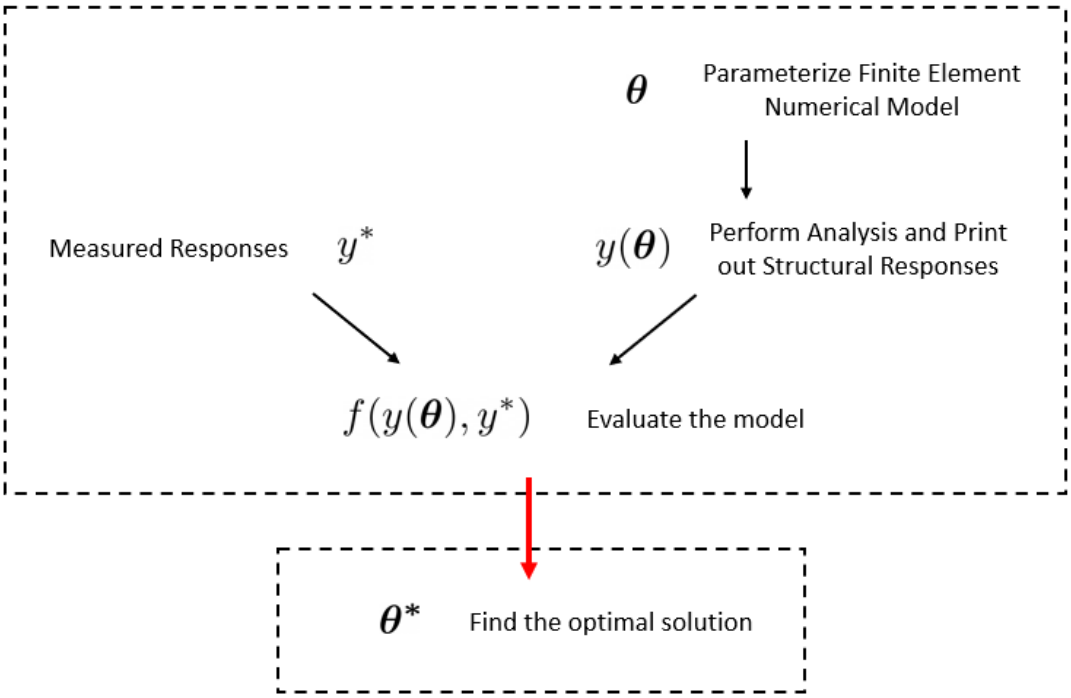


Figure 5.1: Sketch of the FE model updating framework.

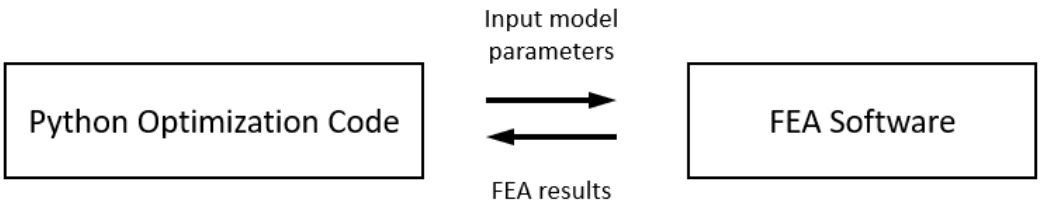


Figure 5.2: Interaction between a Python optimization code and FEA software.

5.2 Applications to Overhead Highway Box Beam Sign Structures

ABAQUS FEA Models

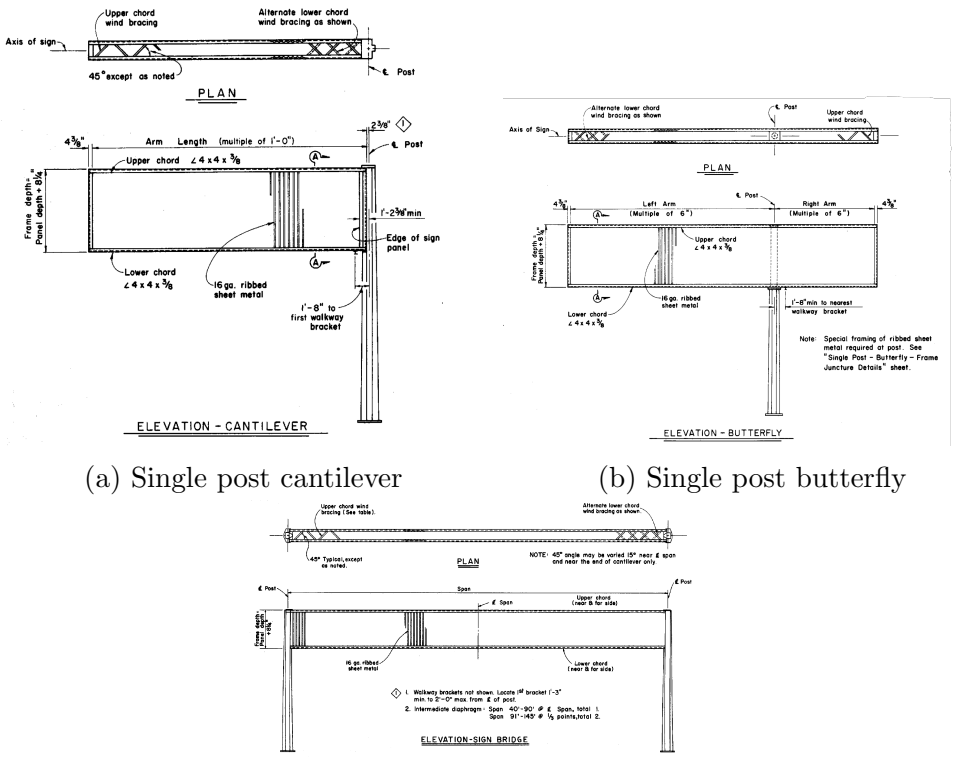
In the project entitled "In-Service Structural Evaluation of Box Beam Overhead Sign Structures"[7], three-dimensional ABAQUS models of the overhead highway sign structures are developed for pre-test simulations. Three different configurations of these structures are selected for this project (Figures 5.3 and 5.4). It is noted that the example structures shown in Figure 5.4 are identified by their ID from the Office of Structure Maintenance and Investigations (OSMI). All structural details including geometry and dimensions of the structures are determined based on the standard as-built drawings[8] and the inspection reports with "Box Beam Perspective Tool v3" data (Figures 5.5 and 5.6). ABAQUS FEA models for the three different types of sign structures are shown in Figure 5.7.

ABAQUS-Python Model Updating Framework

In this project, we propose the SMU framework combining ABAQUS FEA and Python Genetic Algorithm for general box beam overhead sign structures because it is difficult to create highly accurate digital twins of these complex structures only with the available information from drawings and inspections. The flowchart of the proposed framework is shown in Figure 5.8. We can create and run ABAQUS FEA model in the Python environment with the `abaqus` and `abaqusConstant` modules, and the Python code can import the analysis results for evaluating the accuracy of the numerical model from the `abaqus.rpt` where the ABAQUS software records the structural responses we are interested in. All processes in the ABAQUS software are defined as functions in the ABAQUS/Python script, which allows us not only to modify the model input parameters to perform the optimization process but also to easily create models with various geometry and dimensions. In this framework, we collect the experiment data (e.g., material properties and vibration responses) and the structural configuration information (e.g., using the as-built drawings or from terrestrial laser scanning data), and create an initial ABAQUS model through the function defined in the ABAQUS/Python script. Based on the initial model, we can define design parameters to be optimized by selecting design variables considering expected sources of uncertainties. We use the measured structural responses or the dynamic features of the structures estimated based on the responses to evaluate the accuracy of the developed and updated numerical

[7]This project is one of the on-going projects in the PEER Center and STructural Artificial Intelligence Research lab (STAIRlab), `http://stairlab.berkeley.edu/`, with support from California Department of Transportation (Caltrans) as Task Order 008 of the PEER-Bridge Research Program, `https://peer.berkeley.edu/research/peer-bridge`.

[8]Standard Plans (1969, 1971, and 1973) issued by the Division of Highways in the Department of Public Works, State of California.



(a) Single post cantilever

(b) Single post butterfly

(c) Two post frame

Figure 5.3: Three different types of highway sign structure configurations.

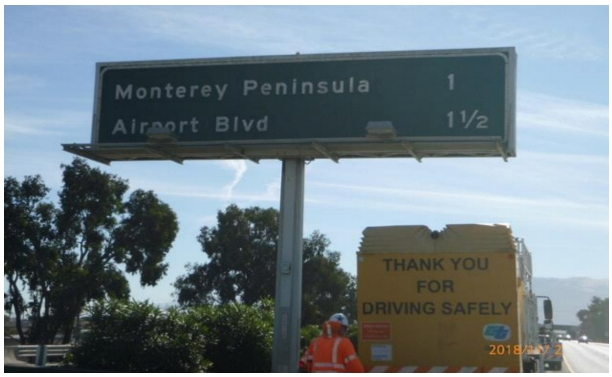
models. This evaluation is the objective function in the proposed SMU framework, and the Genetic Algorithm (refer to Chapter 3) is used in this framework.

In this project, vibration data is adopted to estimate the natural periods of the overhead sign structures, and these estimated natural periods are considered as the ground truth in the proposed SMU framework (Figure 5.9). This vibration data was obtained by Caltrans through Phyphox which is a mobile accelerometer application (Figure 5.10).

5.3 Single Post Butterfly SMU Example

A single post butterfly sign structure located on the Vic Fazio highway (south bound) near Davis, California ($38.552°, -121.768°$), has been chosen to collect material samples (Figure 5.11a). The ABAQUS FEA model for this sign structure is developed to study the responses of the overhead box beam sign structures and to demonstrate the performance of the proposed SMU framework. The panel size of the sign structure is 70 in × 22 ft, and the height of the post is 15.5 ft (Figure 5.11b).

(a) Single post cantilever structure, Location 24, OSMI ID: 44156005239EB1



(b) Single post butterfly structure, Location 20, OSMI ID: 44101087286SB1

(c) Two post frame structure, Location 39, OSMI ID: 36001003437SB1

Figure 5.4: Three example configurations of overhead sign structures (Source: Caltrans).

Vibration Data and Estimated Fundamental Periods

While collecting material samples from the sign structure, the research team[9] in STAIRlab also collected vibration data by using in-house 6-Degrees of Freedom (6-DoF) sensors called 6-DoF PEER-CENTS[10] (Figure 5.12). Three sensors were attached to the sign structure (Figure 5.13), and the team administered hammer hit tests in the longitudinal, transverse, and vertical directions of the sign structure. Accelerations recorded during the tests, refer to Figure 5.14, clearly show the vibrations of the structure identified during four hammer hitting and during the collection of the material samples (cutting process).

The natural periods of vibration are identified based on the free vibration part of the recorded responses by counting cycles (Figure 5.15) and also by using the response spectra

[9]G. Su, S. Günay, A. Kasalanati, J. Meriles, and Z. Wang set up the sensors, performed hammer hits, and collected and analyzed the data. Figures 5.12 to 5.16 are developed together with this group.

[10]CENTS stands for Cost Effective New Technology Sensor.

PANEL DESCRIPTION :

PANEL DIMENSION : (Left to Right) width(ft) x height(ft)

12.0 ft (Approx) X 10.0 ft (Approx)

18.0 ft (Approx) X 10.0 ft (Approx)

18.0 ft (Approx) X 10.0 ft (Approx)

STRUCTURE INFORMATION :

Sign Type : Directional (DIR)
Sign Frame : Box Beam (BOX)
Post Base : Full Penetration
Coating-Type : Painted

Sign Vert. Clr. : Height =19 ft 2 in
Over Live Traffic : Two lanes
Light Fixture : Five
Guardrail : Yes

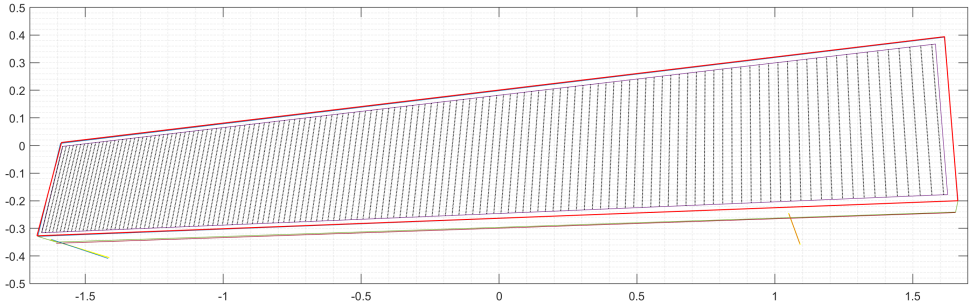
Element No.	Element Description	Tot Qty	Units
S.01	Foundation(s) :	2	EA
S.02	Anchor Bolts :	16	EA
S.03	Base Plate(s) :	2	EA
S.04	Column Support(s) :	2	EA
S.05	Column to Arm/Chord Connection :	4	EA
S.06	Arm/Chord Member :	0	EA
S.07	Truss/Arm/Chord Splice Connection :	0	EA
S.08	Span Truss Members :	68	LF
S.09	Bridge Mounted/Sign Frame :	0	EA
S.10	Sign Panel(s) :	480	SF
S.11	Walkway :	55	LF
S.12	Light Fixture(s) :	5	EA
S.13	Sign Attachment(s) :	0	EA
S.14	CMS Sign(s) :	0	SF

```
##### Box Beam Persspective Tool v3 #####

[Location = 39]
Total Length: 6.999983e+01 ft
Width: 24 in
Sign Panel Depth: 100 in
Chord Vert Leg: 4 in
Total Bert Depth: 108 in
WWB Depth: 4 in
WWB Front Stickout: 0 in
WWB Back Stickout: 57 in
WWB Loc1: 2.174998e+01 in
WWB Loc2: 7.475000e+02 in
```

Figure 5.5: Information about geometry and dimensions of the overhead sign structures available in the inspection reports.

(a) Photograph of the overhead sign structure.



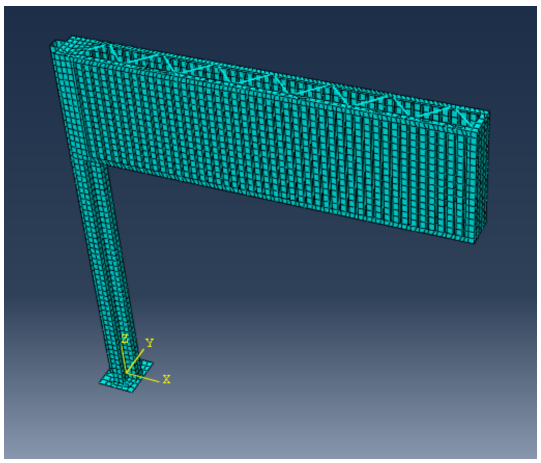
(b) MATLAB plot of the Excel coordinate data.

Figure 5.6: "Box Beam Perspective Tool v3" (Location 39, OSMI ID: 36001003437SB1).

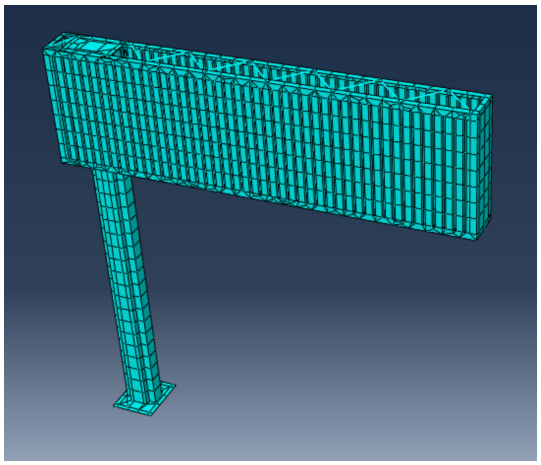
from the vibration data (Figure 5.16). We can observe that the fundamental periods in the X, Y, and Z directions are 0.50 sec, 0.38 sec, and 0.38 sec, respectively. Since the Y and Z directions are in-plane and the X direction is normal to the plane (out-of-plane), we conclude that the sign structure has out-of-plane and in-plane mode shapes with natural periods 0.50 sec and 0.38 sec, respectively. Therefore, the ground truth values of the natural frequencies of this sign structure are $\boldsymbol{y^*} = [y^*_{out}, y^*_{in}] = [(0.50)^{-1}, (0.38)^{-1}]$ sec^{-1} $= [2.00, 2.63]$ Hz.

ABAQUS Model and Parameters

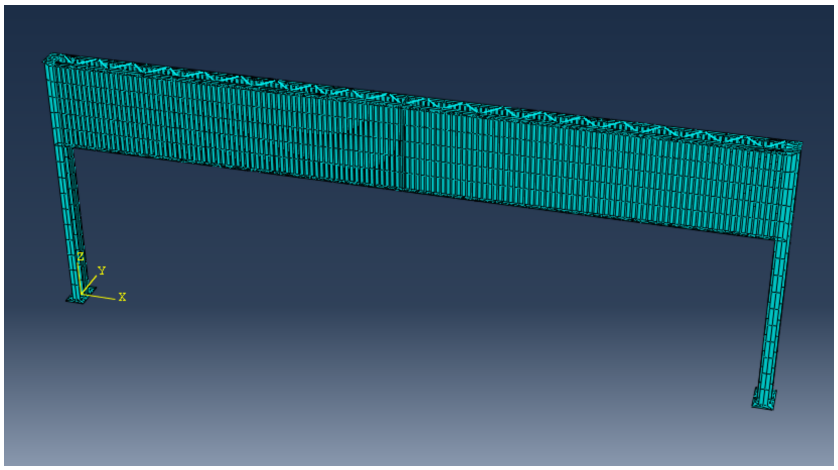
Detailed information about the sign structure could be obtained through the "as-built" drawings [17] for the generic overhead box beam sign structures, but there were still some parts requiring precise on-site measurements to be determined (Figure 5.17). Furthermore, it is expected that there may be changes in the "as-found" conditions of the sign structure due to various causes (e.g., steel corrosion damage affecting the connections between post, sign panel, box chords, and ribbed sheet metal) [45]. Based on these considerations, 11 "design" parameters (to be updated using the SMU framework) are selected for the ABAQUS model of the sign structure (Table 5.1, Figure 5.18). Initial values and feasible domains of the these parameters are determined as shown in Table 5.1. Mode shapes and their natural frequencies



(a) Single post cantilever



(b) Single post butterfly



(c) Two post frame

Figure 5.7: ABAQUS FEA models of overhead highway box beam sign structures.

of the ABAQUS model with the initial values are shown in Figure 5.19 (the obtained results from this ABAQUS model with the initial values of the parameters are: 2.56 Hz for the out-of-plane mode shape and 2.73 Hz for the in-plane mode shape). Comparing with the ground-truth, $\boldsymbol{y}^* = [2.00, 2.63]$ Hz, this model should be updated to improve its vibration response predictions by the proposed SMU framework.

The Mean Squared Error (MSE) function is adopted as the objective function for the SMU in this section. This function is expressed as follows,

$$MSE(\boldsymbol{y}, \boldsymbol{y}^*) = \sqrt{(y_{out} - y^*_{out})^2 + (y_{in} - y^*_{in})^2}, \tag{5.2}$$

where y_{out} and y_{in} are the natural frequencies of the out-of-plane mode and the in-plane modes, respectively, obtained from the ABAQUS sign structure model and its updated ver-

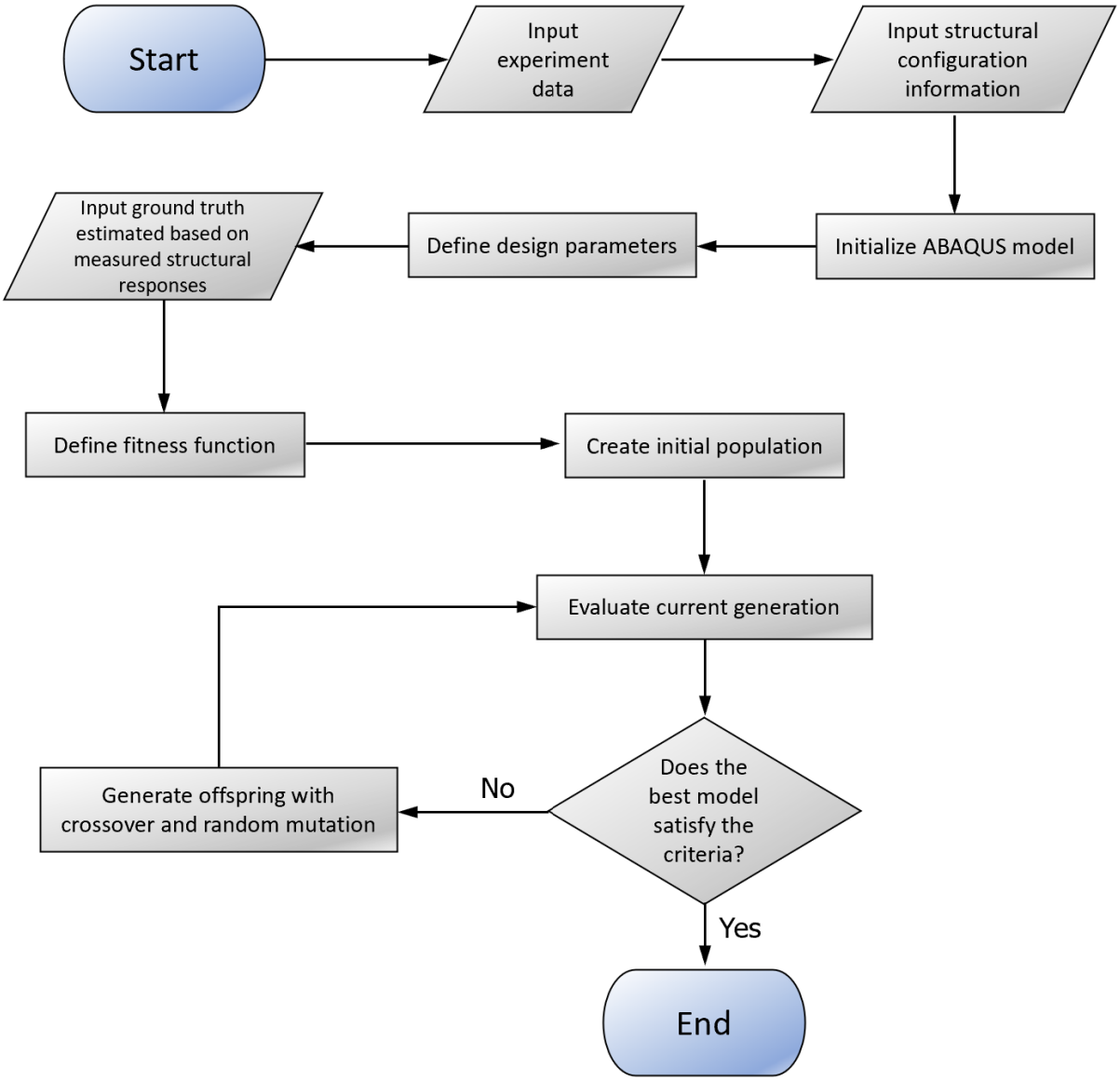


Figure 5.8: Flowchart of the proposed ABAQUS SMU framework for generic structures.

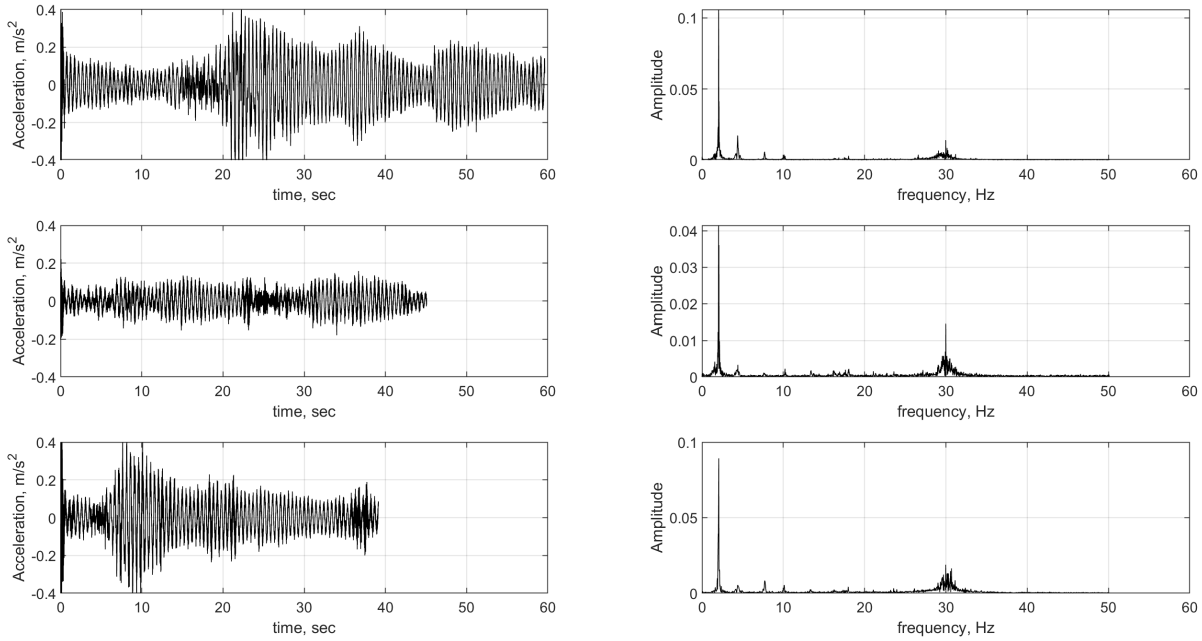


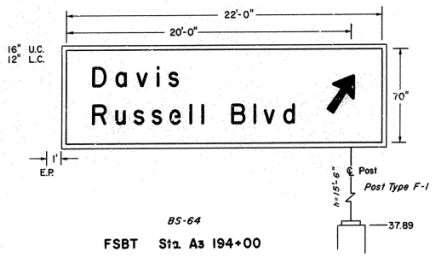
Figure 5.9: Phyphox vibration data and frequency response measured for 59.68 sec at 09:21:38 AM (Top), 45.16 sec at 09:22:58 AM (Middle) and 39.19 sec at 09:24:04 AM (Bottom), Location 39, OSMI ID: 36001003437SB1, Date: 10/31/2018 (Source: Caltrans).

(a) Location 39, Y-axis is vertical.

(b) Location 31, X-axis is vertical.

Figure 5.10: Phyphox-running device (iphone) attached to a box beam sign structure.

(a) Photograph of the sign structure



(b) As-built drawing of the sign structure

Figure 5.11: Single post butterfly sign structure near Davis, CA.

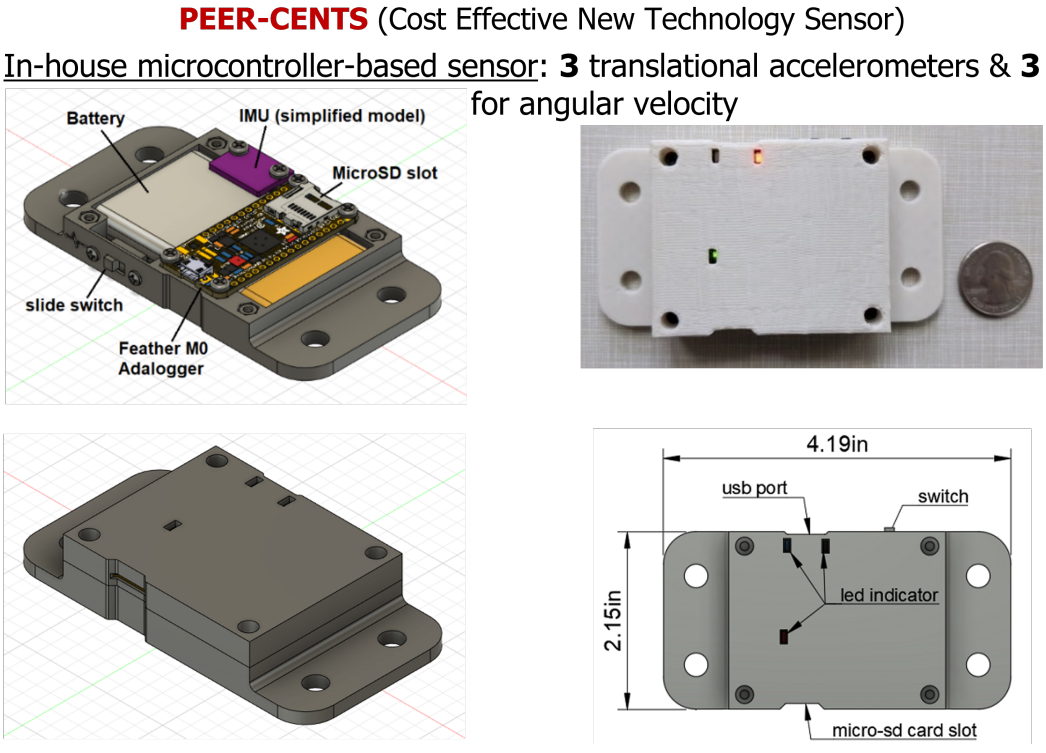


Figure 5.12: PEER-CENTS used for vibration data collection.

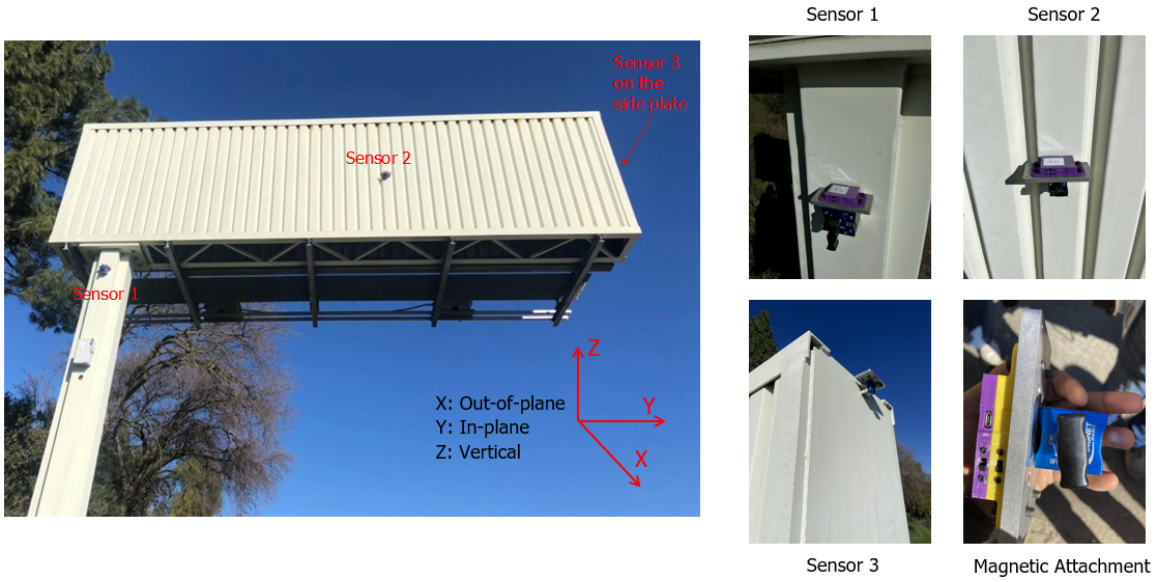


Figure 5.13: Sensor locations on the single post butterfly sign structure near Davis, CA.

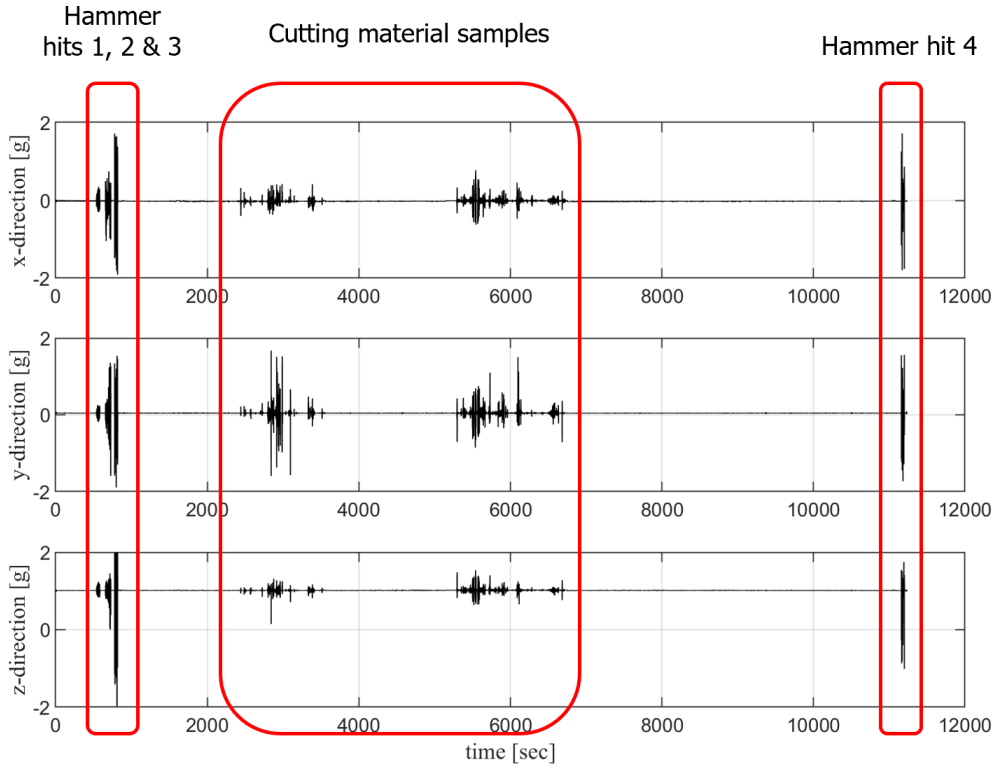


Figure 5.14: Recorded accelerations during hammer hit tests in three directions.

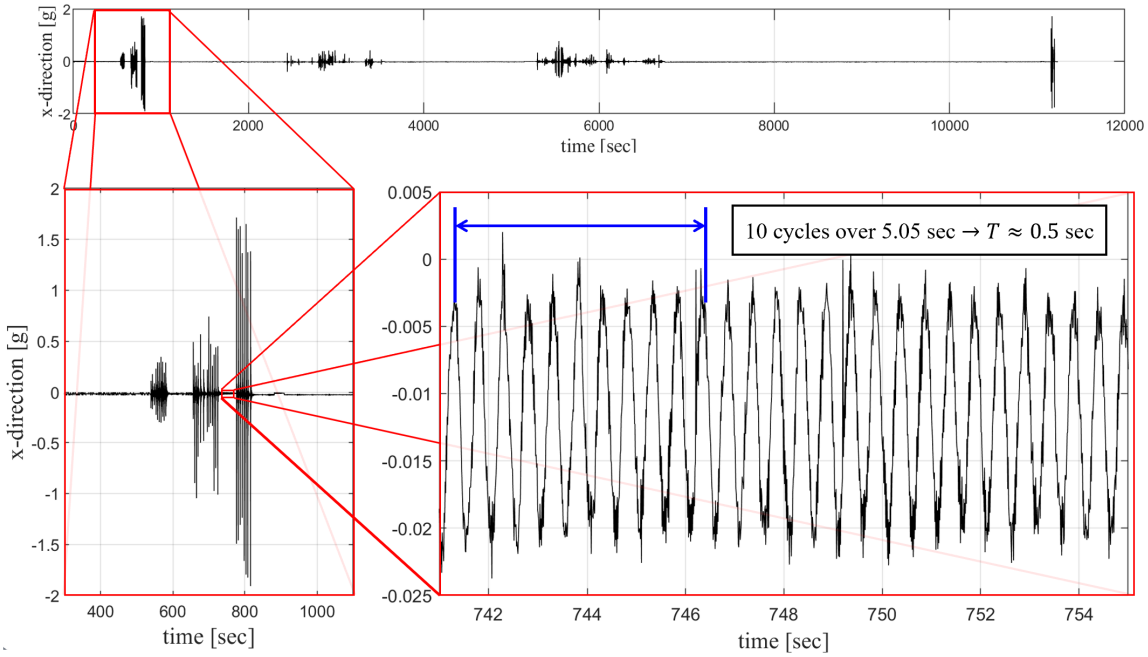
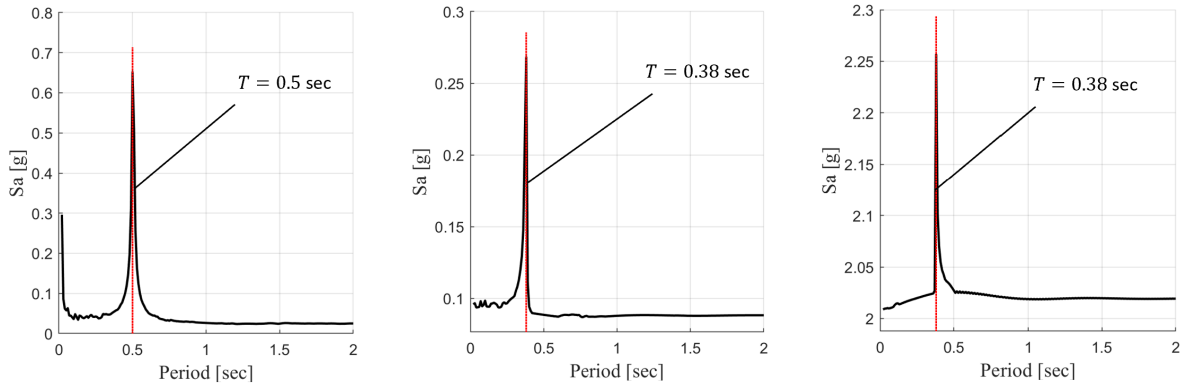
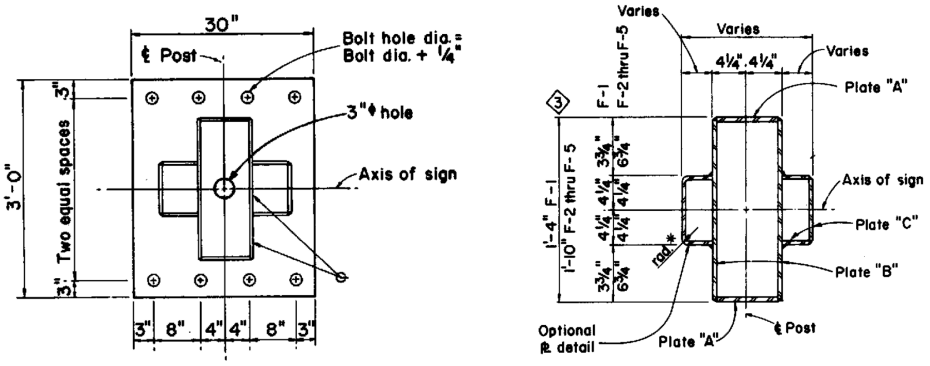


Figure 5.15: Free vibration acceleration after the 2nd hammer hit in X-direction.

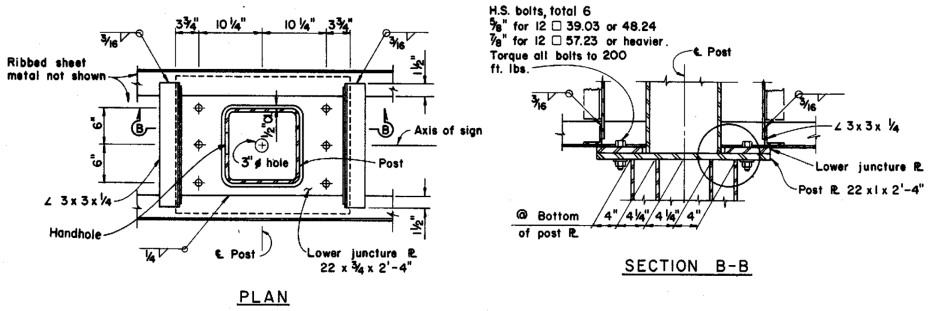


(a) X-direction (b) Y-direction (c) Z-direction

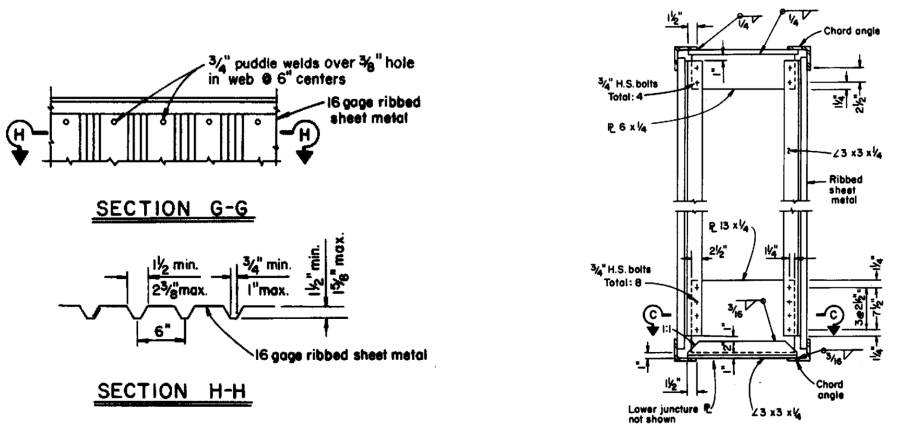
Figure 5.16: Response spectra of the free vibration data and estimated fundamental periods in the three directions, $T_X = 0.50$ sec, $T_Y = 0.38$ sec, and $T_Z = 0.38$ sec.



(a) Details of the base plate (left) and post section (right)

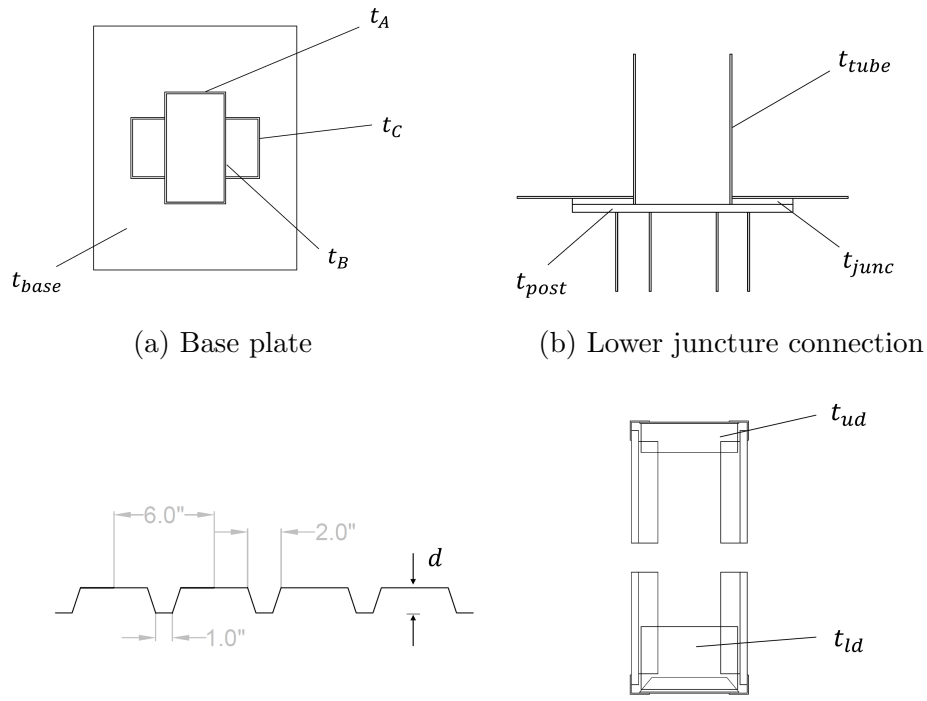


(b) Details of the lower juncture connection



(c) Details of the 16 gauge sheet metal (left) and diaphragm section (right).

Figure 5.17: As-built drawings of the generic single post butterfly sign structure.



(a) Base plate

(b) Lower juncture connection

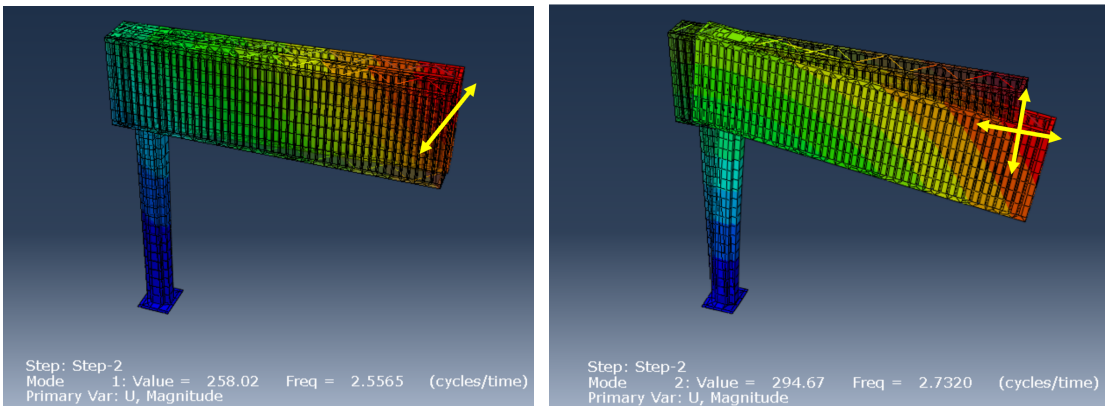
(c) Ribbed sheet metal

(d) Diaphragm

Figure 5.18: Parameterized sign structure model drawings.

Table 5.1: Selected 11 design parameters of the sign structure.

Parameter	Description	Initial Value	Feasible Domain
t_A	Plate A thickness	0.25 in	[0.225; 0.275] in
t_B	Plate B thickness	0.25 in	[0.225; 0.275] in
t_C	Plate C thickness	0.25 in	[0.225; 0.275] in
t_{base}	Base plate thickness	2.00 in	[1.600; 2.400] in
t_{post}	Post plate thickness	1.00 in	[0.800; 1.200] in
t_{junc}	Lower juncture plate thickness	0.75 in	[0.600; 0.900] in
t_{tube}	Post tube thickness	0.3125 in	[0.250; 0.375] in
t_{ud}	Upper diaphragm thickness	0.25 in	[0.225; 0.275] in
t_{ld}	Lower diaphragm thickness	0.25 in	[0.225; 0.275] in
d	Ribbed sheet metal depth	1.50 in	[1.500; 1.625] in
E_s	Steel tensile modulus of elasticity	29,000 ksi	[26,100; 31,900] ksi



(a) 1st mode shape, out-of plane (2.56 Hz) (b) 2nd mode shape, in-plane (2.73 Hz)

Figure 5.19: Natural frequencies and mode shapes of the sign structure ABAQUS model with initial parameters.

sions. Table 5.2 shows the parameters of the GA used in this model updating application.

Table 5.2: Parameters of the GA to update the sign structure model.

Population size	Crossover rate	Random Mutation rate	Number of generations
30	0.5	0.05	5

Results

In this SMU application, the optimization algorithm evaluated 150 ABAQUS models (i.e., ABAQUS software performed 150 different eigen analyses) and took 3,083 sec with a computer having an Intel Core i7-8650U with 8 GB RAM. The model updating results are shown in Figures 5.20 and 5.21. The natural frequencies of the final updated model are 2.014 Hz (0.497 sec; out-of-plane mode) and 2.584 Hz (0.387 sec; in-plane mode), which are very close to the ground truth, namely 2.00 Hz (0.50 sec; out-of-plane mode) and 2.63 Hz (0.38 sec; in-plane mode), refer to Figure 5.22. The results demonstrated that the proposed SMU framework can successfully determine the optimal ABAQUS sign structure model showing very similar dynamic features to those estimated from the measured structural responses of the actual overhead sign structure. The design parameters of the updated single post butterfly sign structure are summarized in Table 5.3. The results show that the model is sensitive to the design parameters since it has been effectively updated with small changes in the parameters.

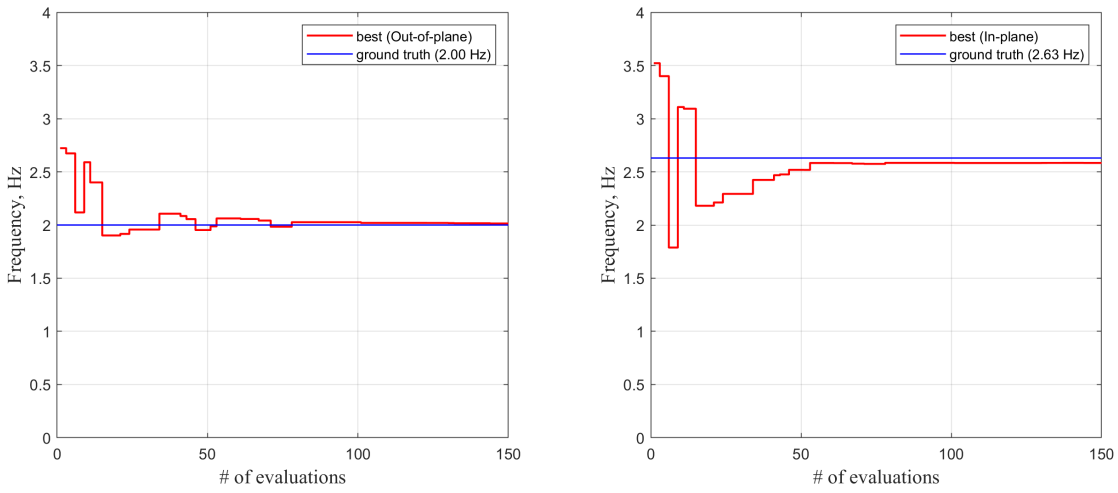


Figure 5.20: Model updating results of the single post butterfly sign structure, the number of evaluations vs. frequencies of the out-of-plane mode (left) and in-plane mode (right).

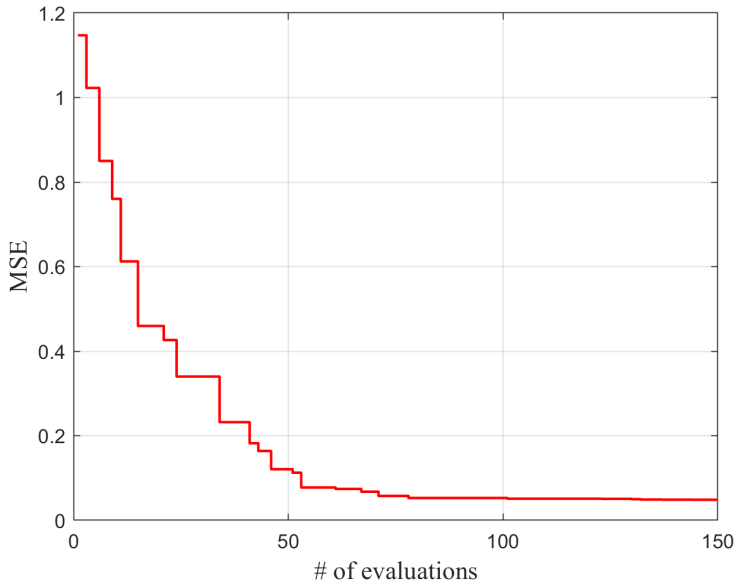
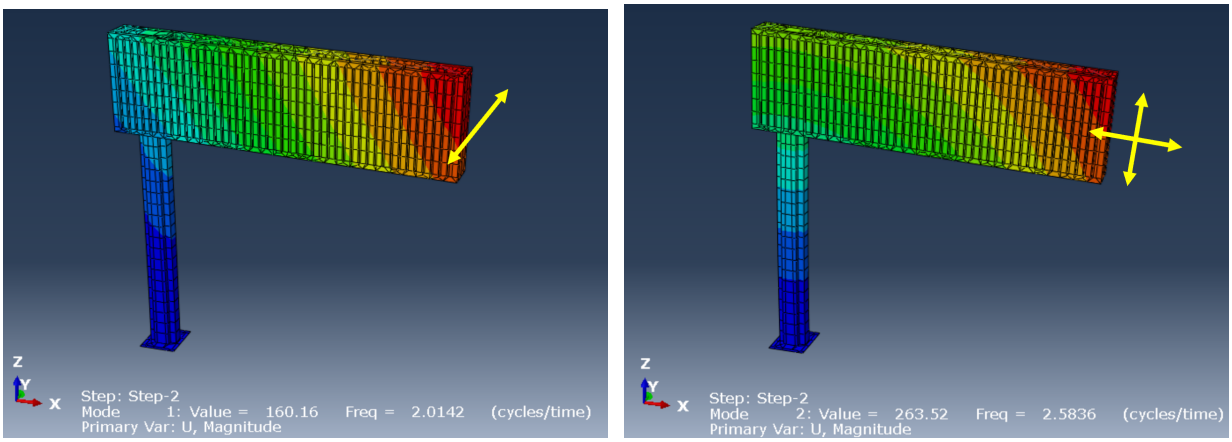


Figure 5.21: Model updating results of the single post butterfly sign structure, the number of evaluations vs. mean squared error.



(a) 1st mode shape, out-of plane (2.01 Hz) (b) 2nd mode shape, in-plane (2.58 Hz)

Figure 5.22: Natural frequencies and mode shapes of the updated final single post butterfly sign structure model.

Table 5.3: Selected 11 design parameters of the sign structure.

Parameter	Description	Initial Value	Updated Value
t_A	Plate A thickness	0.25 in	0.2453 in
t_B	Plate B thickness	0.25 in	0.2327 in
t_C	Plate C thickness	0.25 in	0.2718 in
t_{base}	Base plate thickness	2.00 in	2.3828 in
t_{post}	Post plate thickness	1.00 in	0.8477 in
t_{junc}	Lower juncture plate thickness	0.75 in	0.6389 in
t_{tube}	Post tube thickness	0.3125 in	0.3191 in
t_{ud}	Upper diaphragm thickness	0.25 in	0.2337 in
t_{ld}	Lower diaphragm thickness	0.25 in	0.2265 in
d	Ribbed sheet metal depth	1.50 in	1.55 in
E_s	Steel tensile modulus of elasticity	29,000 ksi	27,385 ksi

Chapter 6

Conclusions and Future Extensions

6.1 Conclusions

In this dissertation, a Performance-based Design Optimization (PDO) framework using the Probabilistic Performance-Based Design (PPBD) evaluation approach and two meta-heuristic algorithms, Genetic Algorithm (GA) and Bayesian Optimization Algorithm (BOA), namely PDO-GA and PD-BOA, is proposed. PDO-GA and PD-BOA are developed in Chapters 2 and 3, and their performances are demonstrated with the applications to hypothetical three-bay, five-story steel Moment Resisting Frame (MRF) and three-bay, nine-story steel MRF buildings in Chapter 4. The ABAQUS-Python Structural Model Updating (SMU) framework is proposed in Chapter 5 and its performance is demonstrated through the application to the single-post butterfly type overhead highway sign structure in California. The main conclusions of the study are summarized in the following list.

- In the application of the steel MRF buildings, both PDO-GA and PD-BOA produced optimal solutions within few evaluations, although we set a high value for the target Generalized Expected Utility (GEU) ($GEU(\alpha = 0.3)_{target} = 0.5$). The PD-BOA outperformed the PD-GA in the application of a more complex structure (3-bay, 9 story steel MRF building), which demonstrated that the searching algorithm in the PD-BOA shows a good performance in the PDO framework. However, it is expected that, if we are optimizing in a high-dimensional design space, the searching algorithm would take significant computational cost to determine the next candidates.

- The artificial ground motions generated by the quasi-stationary model are accurately compatible with the USGS uniform hazard response spectra, and the code demonstrated its performance in terms of computational time (0.13 sec per generating one ground motion, $\Delta t = 0.02$ sec and $t_{end} = 30$ sec). This approach is simpler and more efficient than selecting and scaling the ground motions from an existing database, e.g., the PEER-NGA (`https://peer.berkeley.edu/research/nga-west-2`).

- Kernel Density Maximum Entropy (KDME) algorithm is able to accuretly determine unbiased distributions. The developed Algorithm 1 in Chapter 2 shows a good computational performance by solving convex problems with CVXPY [18].
- Nonlinear time-history analysis was performed using OpenSeesPy[11]. The analysis using this platform is very efficient and accurate (analysis results were cross-checked by commercial software, MIDAS/Gen [24]).
- In the application of the SMU framework, the application demonstrated excellent results. The model's high sensitivity to small changes in the design parameters might be caused by the fact that this model is close to the single-post cantilever type (not the butterfly type) sign structure. Further validation will be conducted with other types of sign structures in the future.
- Comparing the updated model shown in Table 5.1 with the model based on the as-built drawing shown in Table 5.3, we observe that most design parameters decreased from the initial values. This implies that most structural elements of the updated model have experienced some reduction of their strength and stiffness. This may be attributed to structural deterioration over time, e.g., caused by fatigue, corrosion, or extreme loading.
- It has been verified that the Finite Element Analysis (FEA) software, such as ABAQUS, can be executed without notable computational losses in a Python code environment. A single eigen analysis took approximately 20 sec in the ABAQUS interface, while 150 evaluations took 3,083 sec (20 sec $\times 150 = 3,000$ sec). It is to be noted that, a similar OpenSees-Python SMU framework was developed in the course of this study, which was not discussed in this dissertation (in that regard, OpenSees is executed through a Tcl[12] code, not with the `OpenSeespy module`). In this framework, Python code run the OpenSees Tcl code with `subprocess` module. The computational loss is also negligible in this case.

6.2 Future Extensions

The future developments aim to make the PDO algorithms more efficient in terms of computational cost. In addition, the PDO algorithms can be extended to Reliability-Based Design Optimization (RBDO), i.e., considering the uncertainties in the material and cross-section properties. Future Extension of this dissertation could also include the improvements in FE modeling and the SMU algorithms. Future improvements are suggested in the following list.

[11]Developed by M. Zhu & M. Scott, Oregon State University, `https://openseespydoc.readthedocs.io/en/latest/`.

[12]Tool Command Language.

- The PPBD evaluation is a complex procedure requiring a large number of sophisticated structural analyses (e.g., nonlinear time-history analysis in the applications in Chapter 4). In addition, we need to explore high dimensional design spaces to perform the PDO for large and realistic designs. Accordingly, the current PDO framework demands significant computational resources. To reduce the design set effectively, multi-fidelity models with varying computational costs and accuracy can be adopted in the PDO framework [31]. Multi-fidelity surrogate models would enable us to leverage low-fidelity models to immediately remove poor design alternatives [19] and improve computational efficiency of the PDO framework.

- In Reliability-Based Design (RBD), inherent uncertainties in material and cross-section properties are considered by adding statistical dispersions to the design variables [9][39]. The RBD approach can be adopted in the proposed PDO framework by treating the design variables as random ones. In the PDO framework using the RBD approach, we need reliability method, such as First-Order Reliability Method (FORM) or Second-Order Reliability Method (SORM), to quantify the uncertainties in the material and cross-section properties and to construct the distributions of the Engineering Demand Parameters (EDPs). However, this step would require a large number of structural analyses to obtain a single Probability Density Function (PDF) of the considered EDP, an issue that justifies further development towards better computational efficiency.

- Advanced FEA software, such as ABAQUS, OpenSees, and SAP2000, enable highly sophisticated modeling, allowing for the development of more flexible models with various structural behaviors that can be considered in the SMU. For instance, detailed modeling of the connection between post, box frames, sign panel, and ribbed sheet metal for a highway sign structure would increase the model's degrees of freedom. Additionally, the inclusion of element conditions due to fatigue or corrosion would allow for the development of more accurate numerical models using information obtained from maintenance and inspections [1]. Trained Deep Learning (DL) segmentation models, such as visual-based corrosion detection, could provide a wealth of information about the corrosion of structures using only photographs [46] obtained through regular maintenance efforts. Such high model flexibility would allow for the exploration of additional "design" parameters that the objective function is sensitive to, which would greatly enhance the efficiency of the proposed SMU framework.

- The parameters of the metaheuristic algorithms should be optimized for better performance but those adopted for the GA in the presented applications (Chapters 4 and 5) are randomly chosen. Integration among the GA parameters is vital for successful GA search. Hassanat et al. [23] summarized and reviewed the topic of choosing random mutation and crossover rates for GAs. Developing the GA parameter settings for the PDO-GA and the ABAQUS-Python or OpenSees-Python model updating of different structural system types would be an important improvement for the proposed PDO and SMU frameworks.

Bibliography

[1] Khalid W Al Shboul, Hayder A Rasheed, and Husam A Alshareef. "Intelligent approach for accurately predicting fatigue damage in overhead highway sign structures". In: *Structures*. Vol. 34. Elsevier. 2021, pp. 3453–3463.

[2] U Alibrandi and G Ricciardi. "Efficient evaluation of the pdf of a random variable through the kernel density maximum entropy approach". In: *International journal for numerical methods in engineering* 75.13 (2008), pp. 1511–1548.

[3] Umberto Alibrandi. "Response Spectrum Code-Conforming PEER PBEE using Stochastic Dynamic Analysis and Information Theory". In: *KSCE Journal of Civil Engineering* 22.3 (2018), pp. 1002–1015.

[4] Umberto Alibrandi and Khalid M Mosalam. "A decision support tool for sustainable and resilient building design". In: *Risk and reliability analysis: Theory and applications*. Springer, 2017, pp. 509–536.

[5] Umberto Alibrandi and Khalid M Mosalam. "Kernel density maximum entropy method with generalized moments for evaluating probability distributions, including tails, from a small sample of data". In: *International Journal for Numerical Methods in Engineering* 113.13 (2018), pp. 1904–1928.

[6] Nizar Faisal Alkayem et al. "Structural damage detection using finite element model updating with evolutionary algorithms: a survey". In: *Neural Computing and Applications* 30 (2018), pp. 389–411.

[7] ASCE. "Minimum design loads for buildings and other structures". In: American Society of Civil Engineers. 2013.

[8] ASTM ASTM et al. *Standard specification for carbon structural steel*. 2012.

[9] Hassan Baji. "The effect of uncertainty in material properties and model error on the reliability of strength and ductility of reinforced concrete members". In: (2014).

[10] Daniel Thomas Bartilson. *Model updating in structural dynamics: advanced parametrization, optimal regularization, and symmetry considerations*. Columbia University, 2019.

[11] Pierfrancesco Cacciola. "A stochastic approach for generating spectrum compatible fully nonstationary earthquakes". In: *Computers & Structures* 88.15-16 (2010), pp. 889–901.

[12] Pierfrancesco Cacciola, P Colajanni, and G Muscolino. "Combination of modal responses consistent with seismic input representation". In: *Journal of Structural Engineering* 130.1 (2004), pp. 47–55.

[13] Giuseppe C Calafiore and Laurent El Ghaoui. *Optimization models*. Cambridge university press, 2014.

[14] Jenna Carr. "An introduction to genetic algorithms". In: *Senior Project* 1.40 (2014), p. 7.

[15] Jaskanwal PS Chhabra and Gordon P Warn. "A method for bounding imprecise probabilistic criteria when using a sequential decision process for the design of structural systems". In: *Structural Safety* 79 (2019), pp. 39–53.

[16] Jianye Ching, Matthew Muto, and James L Beck. "Structural model updating and health monitoring with incomplete modal data using Gibbs sampler". In: *Computer-Aided Civil and Infrastructure Engineering* 21.4 (2006), pp. 242–257.

[17] Division of Highways Department of Public Works. "Standard Plans". In: State of California, Business and Transportation Agency, 1973.

[18] Steven Diamond and Stephen Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* (2016). To appear. URL: `http://stanford.edu/~boyd/papers/pdf/cvxpy_paper.pdf`.

[19] Alexander IJ Forrester, András Sóbester, and Andy J Keane. "Multi-fidelity optimization via surrogate modelling". In: *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463.2088 (2007), pp. 3251–3269.

[20] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[21] Selim Günay and Khalid M Mosalam. "PEER performance-based earthquake engineering methodology, revisited". In: *Journal of Earthquake Engineering* 17.6 (2013), pp. 829–858.

[22] C Haselton and J Baker. "The FEMA P-58 methodology and the seismic performance prediction program (SP3)". In: *Available at: hbrisk. com/sp3 (last accessed 1 June 2019)* (2019).

[23] Ahmad Hassanat et al. "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach". In: *Information* 10.12 (2019), p. 390.

[24] MIDAS Institute Inc. *MIDAS/GEN 2021, v3.1*. New York, NY. URL: `https://www.midasoft.com/`.

[25] Paul C Jennings, George W Housner, and N Chien Tsai. "Simulated earthquake motions". In: (1968).

[26] Lambros S Katafygiotis, Costas Papadimitriou, and Heung-Fai Lam. "A probabilistic approach to structural model updating". In: *Soil Dynamics and Earthquake Engineering* 17.7-8 (1998), pp. 495–507.

[27] JooSeuk Kim and Clayton D Scott. "Robust kernel density estimation". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2529–2565.

[28] Khalid M Mosalam et al. "Performance-based engineering and multi-criteria decision analysis for sustainable and resilient building design". In: *Structural Safety* 74 (2018), pp. 1–13.

[29] John E Mottershead, Michael Link, and Michael I Friswell. "The sensitivity method in finite element model updating: A tutorial". In: *Mechanical systems and signal processing* 25.7 (2011), pp. 2275–2296.

[30] Hyo Seon Park, Jin Woo Hwang, and Byung Kwan Oh. "Integrated analysis model for assessing CO2 emissions, seismic performance, and costs of buildings through performance-based optimal seismic design with sustainability". In: *Energy and Buildings* 158 (2018), pp. 761–775.

[31] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591.

[32] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[33] Johannes O Royset, Selim Günay, and Khalid M Mosalam. "Risk-Adaptive Learning of Seismic Response using Multi-Fidelity Analysis". In: *Proceedings of the International Conference on Applied Statistics and Probability in Civil Engineering (ICASP), Seoul, Korea*. 2019.

[34] Masoud Sanayei et al. "Structural model updating using experimental static measurements". In: *Journal of structural engineering* 123.6 (1997), pp. 792–798.

[35] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[36] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[37] B Sudret, C Mai, and K Konakli. "Assessment of the lognormality assumption of seismic fragility curves using non-parametric representations". In: (2017).

[38] Aldo Tagliani. "Discrete probability distributions in the generalized moment problem". In: *Applied Mathematics and Computation* 112.2-3 (2000), pp. 333–343.

[39] Jian Tu, Kyung K Choi, and Young H Park. "A new study on reliability-based design optimization". In: (1999).

[40] Berwin A Turlach. "Bandwidth selection in kernel density estimation: A review". In: *CORE and Institut de Statistique*. Citeseer. 1993.

[41] Darrell Whitley. "A genetic algorithm tutorial". In: *Statistics and computing* 4.2 (1994), pp. 65–85.

[42] James T Wilson, Frank Hutter, and Marc Peter Deisenroth. "Maximizing acquisition functions for Bayesian optimization". In: *arXiv preprint arXiv:1805.10196* (2018).

[43] Steven R Winterstein and Cameron A MacKenzie. "Extremes of nonlinear vibration: Comparing models based on moments, L-moments, and maximum entropy". In: *Journal of offshore mechanics and Arctic engineering* 135.2 (2013).

[44] TY Yang et al. "Seismic performance evaluation of facilities: Methodology and implementation". In: *Journal of Structural Engineering* 135.10 (2009), pp. 1146–1154.

[45] Wael Zatar and Hai Nguyen. "Condition assessment of ground-mount cantilever weathering-steel overhead sign structures". In: *Journal of Infrastructure Systems* 23.4 (2017), p. 05017005.

[46] Guoqing Zhang and Khalid M. Mosalam. "Automated corrosion detection using deep learning". In: Tsinghua-Berkeley Shenzhen Institute Course Project, 2022.

[47] Minjie Zhu, Frank McKenna, and Michael H Scott. "OpenSeesPy: Python library for the OpenSees finite element framework". In: *SoftwareX* 7 (2018), pp. 6–11.

Appendix A

Properties of AISC Steel Sections

Table A.1: AISC wide flange beam properties [thickness & width dimensions are in (in)].

Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
1	44×335	98.5	31100	44.0	1.03	15.9	1.77
2	44×290	85.4	27000	43.6	0.865	15.8	1.58
3	44×262	77.2	24100	43.3	0.785	15.8	1.42
4	44×230	67.8	20800	42.9	0.710	15.8	1.22
5	40×397	117	32000	41.0	1.22	16.1	2.20
6	40×372	110	29600	40.6	1.16	16.1	2.05
7	40×362	106	28900	40.6	1.12	16.0	2.01
8	40×324	95.3	25600	40.2	1.00	15.9	1.81
9	40×297	87.3	23200	39.8	0.930	15.8	1.65
10	40×277	81.5	21900	39.7	0.830	15.8	1.58
11	40×249	73.5	19600	39.4	0.750	15.8	1.42
12	40×215	63.5	16700	39.0	0.650	15.8	1.22
13	40×199	58.8	14900	38.7	0.650	15.8	1.07
14	40×392	116	29900	41.6	1.42	12.4	2.52
15	40×331	97.7	24700	40.8	1.22	12.2	2.13
16	40×327	95.9	24500	40.8	1.18	12.1	2.13
17	40×294	86.2	21900	40.4	1.06	12.0	1.93
18	40×278	82.3	20500	40.2	1.03	12.0	1.81
19	40×264	77.4	19400	40.0	0.960	11.9	1.73
20	40×235	69.1	17400	39.7	0.830	11.9	1.58
21	40×211	62.1	15500	39.4	0.750	11.8	1.42
22	40×183	53.3	13200	39.0	0.650	11.8	1.20

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
23	40×167	49.3	11600	38.6	0.650	11.8	1.03
24	40×149	43.8	9800	38.2	0.630	11.8	0.830
25	36×395	116	28500	38.4	1.22	16.8	2.20
26	36×361	106	25700	38.0	1.12	16.7	2.01
27	36×330	96.9	23300	37.7	1.02	16.6	1.85
28	36×302	89.0	21100	37.3	0.945	16.7	1.68
29	36×282	82.9	19600	37.1	0.885	16.6	1.57
30	36×262	77.2	17900	36.9	0.840	16.6	1.44
31	36×247	72.5	16700	36.7	0.800	16.5	1.35
32	36×231	68.2	15600	36.5	0.760	16.5	1.26
33	36×256	75.3	16800	37.4	0.960	12.2	1.73
34	36×232	68.0	15000	37.1	0.870	12.1	1.57
35	36×210	61.9	13200	36.7	0.830	12.2	1.36
36	36×194	57.0	12100	36.5	0.765	12.1	1.26
37	36×182	53.6	11300	36.3	0.725	12.1	1.18
38	36×170	50.0	10500	36.2	0.680	12.0	1.10
39	36×160	47.0	9760	36.0	0.650	12.0	1.02
40	36×150	44.3	9040	35.9	0.625	12.0	0.940
41	36×135	39.9	7800	35.6	0.600	12.0	0.790
42	33×387	114	24300	36.0	1.26	16.2	2.28
43	33×354	104	22000	35.6	1.16	16.1	2.09
44	33×318	93.7	19500	35.2	1.04	16.0	1.89
45	33×291	85.6	17700	34.8	0.960	15.9	1.73
46	33×263	77.4	15900	34.5	0.870	15.8	1.57
47	33×241	71.1	14200	34.2	0.830	15.9	1.40
48	33×221	65.3	12900	33.9	0.775	15.8	1.28
49	33×201	59.1	11600	33.7	0.715	15.7	1.15
50	33×169	49.5	9290	33.8	0.670	11.5	1.22
51	33×152	44.9	8160	33.5	0.635	11.6	1.06
52	33×141	41.5	7450	33.3	0.605	11.5	0.960
53	33×130	38.3	6710	33.1	0.580	11.5	0.855
54	33×118	34.7	5900	32.9	0.550	11.5	0.740
55	30×391	115	20700	33.2	1.36	15.6	2.44
56	30×357	105	18700	32.8	1.24	15.5	2.24
57	30×326	95.9	16800	32.4	1.14	15.4	2.05
58	30×292	86.0	14900	32.0	1.02	15.3	1.85
59	30×261	77.0	13100	31.6	0.930	15.2	1.65

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
60	30×235	69.3	11700	31.3	0.830	15.1	1.50
61	30×211	62.3	10300	30.9	0.775	15.1	1.32
62	30×191	56.1	9200	30.7	0.710	15.0	1.19
63	30×173	50.9	8230	30.4	0.655	15.0	1.07
64	30×148	43.6	6680	30.7	0.650	10.5	1.18
65	30×132	38.8	5770	30.3	0.615	10.5	1.00
66	30×124	36.5	5360	30.2	0.585	10.5	0.930
67	30×116	34.2	4930	30.0	0.565	10.5	0.850
68	30×108	31.7	4470	29.8	0.545	10.5	0.760
69	30×99	29.0	3990	29.7	0.520	10.5	0.670
70	30×90	26.3	3610	29.5	0.470	10.4	0.610
71	27×368	109	16200	30.4	1.38	14.7	2.48
72	27×336	99.2	14600	30.0	1.26	14.6	2.28
73	27×307	90.2	13100	29.6	1.16	14.4	2.09
74	27×281	83.1	11900	29.3	1.06	14.4	1.93
75	27×258	76.1	10800	29.0	0.980	14.3	1.77
76	27×235	69.4	9700	28.7	0.910	14.2	1.61
77	27×217	63.9	8910	28.4	0.830	14.1	1.50
78	27×194	57.1	7860	28.1	0.750	14.0	1.34
79	27×178	52.5	7020	27.8	0.725	14.1	1.19
80	27×161	47.6	6310	27.6	0.660	14.0	1.08
81	27×146	43.2	5660	27.4	0.605	14.0	0.975
82	27×129	37.8	4760	27.6	0.610	10.0	1.10
83	27×114	33.6	4080	27.3	0.570	10.1	0.930
84	27×102	30.0	3620	27.1	0.515	10.0	0.830
85	27×94	27.6	3270	26.9	0.490	10.0	0.745
86	27×84	24.7	2850	26.7	0.460	10.0	0.640
87	24×370	109	13400	28.0	1.52	13.7	2.72
88	24×335	98.3	11900	27.5	1.38	13.5	2.48
89	24×306	89.7	10700	27.1	1.26	13.4	2.28
90	24×279	81.9	9600	26.7	1.16	13.3	2.09
91	24×250	73.5	8490	26.3	1.04	13.2	1.89
92	24×229	67.2	7650	26.0	0.960	13.1	1.73
93	24×207	60.7	6820	25.7	0.870	13.0	1.57
94	24×192	56.5	6260	25.5	0.810	13.0	1.46
95	24×176	51.7	5680	25.2	0.750	12.9	1.34
96	24×162	47.8	5170	25.0	0.705	13.0	1.22

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
97	24×146	43.0	4580	24.7	0.650	12.9	1.09
98	24×131	38.6	4020	24.5	0.605	12.9	0.960
99	24×117	34.4	3540	24.3	0.550	12.8	0.850
100	24×104	30.7	3100	24.1	0.500	12.8	0.750
101	24×103	30.3	3000	24.5	0.550	9.00	0.980
102	24×94	27.7	2700	24.3	0.515	9.07	0.875
103	24×84	24.7	2370	24.1	0.470	9.02	0.770
104	24×76	22.4	2100	23.9	0.440	8.99	0.680
105	24×68	20.1	1830	23.7	0.415	8.97	0.585
106	24×62	18.2	1550	23.7	0.430	7.04	0.590
107	24×55	16.2	1350	23.6	0.395	7.01	0.505
108	21×275	81.8	7690	24.1	1.22	12.9	2.19
109	21×248	73.8	6830	23.7	1.10	12.8	1.99
110	21×223	66.5	6080	23.4	1.00	12.7	1.79
111	21×201	59.3	5310	23.0	0.910	12.6	1.63
112	21×182	53.6	4730	22.7	0.830	12.5	1.48
113	21×166	48.8	4280	22.5	0.750	12.4	1.36
114	21×147	43.2	3630	22.1	0.720	12.5	1.15
115	21×132	38.8	3220	21.8	0.650	12.4	1.04
116	21×122	35.9	2960	21.7	0.600	12.4	0.960
117	21×111	32.6	2670	21.5	0.550	12.3	0.875
118	21×101	29.8	2420	21.4	0.500	12.3	0.800
119	21×93	27.3	2070	21.6	0.580	8.42	0.930
120	21×83	24.4	1830	21.4	0.515	8.36	0.835
121	21×73	21.5	1600	21.2	0.455	8.30	0.740
122	21×68	20.0	1480	21.1	0.430	8.27	0.685
123	21×62	18.3	1330	21.0	0.400	8.24	0.615
124	21×55	16.2	1140	20.8	0.375	8.22	0.522
125	21×48	14.1	959	20.6	0.350	8.14	0.430
126	21×57	16.7	1170	21.1	0.405	6.56	0.650
127	21×50	14.7	984	20.8	0.380	6.53	0.535
128	21×44	13.0	843	20.7	0.350	6.50	0.450
129	18×311	91.6	6970	22.3	1.52	12.0	2.74
130	18×283	83.3	6170	21.9	1.40	11.9	2.50
131	18×258	76.0	5510	21.5	1.28	11.8	2.30
132	18×234	68.6	4900	21.1	1.16	11.7	2.11
133	18×211	62.3	4330	20.7	1.06	11.6	1.91

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
134	18×192	56.2	3870	20.4	0.960	11.5	1.75
135	18×175	51.4	3450	20.0	0.890	11.4	1.59
136	18×158	46.3	3060	19.7	0.810	11.3	1.44
137	18×143	42.0	2750	19.5	0.730	11.2	1.32
138	18×130	38.3	2460	19.3	0.670	11.2	1.20
139	18×119	35.1	2190	19.0	0.655	11.3	1.06
140	18×106	31.1	1910	18.7	0.590	11.2	0.940
141	18×97	28.5	1750	18.6	0.535	11.1	0.870
142	18×86	25.3	1530	18.4	0.480	11.1	0.770
143	18×76	22.3	1330	18.2	0.425	11.0	0.680
144	18×71	20.9	1180	18.5	0.495	7.64	0.810
145	18×65	19.1	1070	18.4	0.450	7.59	0.750
146	18×60	17.6	984	18.2	0.415	7.56	0.695
147	18×55	16.2	890	18.1	0.390	7.53	0.630
148	18×50	14.7	800	18.0	0.355	7.50	0.570
149	18×46	13.5	712	18.1	0.360	6.06	0.605
150	18×40	11.8	612	17.9	0.315	6.02	0.525
151	18×35	10.3	510	17.7	0.300	6.00	0.425
152	16×100	29.4	1490	17.0	0.585	10.4	0.985
153	16×89	26.2	1300	16.8	0.525	10.4	0.875
154	16×77	22.6	1110	16.5	0.455	10.3	0.760
155	16×67	19.6	954	16.3	0.395	10.2	0.665
156	16×57	16.8	758	16.4	0.430	7.12	0.715
157	16×50	14.7	659	16.3	0.380	7.07	0.630
158	16×45	13.3	586	16.1	0.345	7.04	0.565
159	16×40	11.8	518	16.0	0.305	7.00	0.505
160	16×36	10.6	448	15.9	0.295	6.99	0.430
161	16×31	9.13	375	15.9	0.275	5.53	0.440
162	16×26	7.68	301	15.7	0.250	5.50	0.345
163	14×398	117	6000	18.3	1.77	16.6	2.85
164	14×370	109	5440	17.9	1.66	16.5	2.66
165	14×342	101	4900	17.5	1.54	16.4	2.47
166	14×311	91.4	4330	17.1	1.41	16.2	2.26
167	14×283	83.3	3840	16.7	1.29	16.1	2.07
168	14×257	75.6	3400	16.4	1.18	16.0	1.89
169	14×233	68.5	3010	16.0	1.07	15.9	1.72
170	14×211	62.0	2660	15.7	0.980	15.8	1.56

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
171	14×193	56.8	2400	15.5	0.890	15.7	1.44
172	14×176	51.8	2140	15.2	0.830	15.7	1.31
173	14×159	46.7	1900	15.0	0.745	15.6	1.19
174	14×145	42.7	1710	14.8	0.680	15.5	1.09
175	14×132	38.8	1530	14.7	0.645	14.7	1.03
176	14×120	35.3	1380	14.5	0.590	14.7	0.940
177	14×109	32.0	1240	14.3	0.525	14.6	0.860
178	14×99	29.1	1110	14.2	0.485	14.6	0.780
179	14×90	26.5	999	14.0	0.440	14.5	0.710
180	14×82	24.0	881	14.3	0.510	10.1	0.855
181	14×74	21.8	795	14.2	0.450	10.1	0.785
182	14×68	20.0	722	14.0	0.415	10.0	0.720
183	14×61	17.9	640	13.9	0.375	10.0	0.645
184	14×53	15.6	541	13.9	0.370	8.06	0.660
185	14×48	14.1	484	13.8	0.340	8.03	0.595
186	14×43	12.6	428	13.7	0.305	8.00	0.530
187	14×38	11.2	385	14.1	0.310	6.77	0.515
188	14×34	10.0	340	14.0	0.285	6.75	0.455
189	14×30	8.85	291	13.8	0.270	6.73	0.385
190	14×26	7.69	245	13.9	0.255	5.03	0.420
191	14×22	6.49	199	13.7	0.230	5.00	0.335
192	12×336	98.9	4060	16.8	1.78	13.4	2.96
193	12×305	89.5	3550	16.3	1.63	13.2	2.71
194	12×279	81.9	3110	15.9	1.53	13.1	2.47
195	12×252	74.1	2720	15.4	1.40	13.0	2.25
196	12×230	67.7	2420	15.1	1.29	12.9	2.07
197	12×210	61.8	2140	14.7	1.18	12.8	1.90
198	12×190	56.0	1890	14.4	1.06	12.7	1.74
199	12×170	50.0	1650	14.0	0.960	12.6	1.56
200	12×152	44.7	1430	13.7	0.870	12.5	1.40
201	12×136	39.9	1240	13.4	0.790	12.4	1.25
202	12×120	35.2	1070	13.1	0.710	12.3	1.11
203	12×106	31.2	933	12.9	0.610	12.2	0.990
204	12×96	28.2	833	12.7	0.550	12.2	0.900
205	12×87	25.6	740	12.5	0.515	12.1	0.810
206	12×79	23.2	662	12.4	0.470	12.1	0.735
207	12×72	21.1	597	12.3	0.430	12.0	0.670

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
208	12×65	19.1	533	12.1	0.390	12.0	0.605
209	12×58	17.0	475	12.2	0.360	10.0	0.640
210	12×53	15.6	425	12.1	0.345	10.0	0.575
211	12×50	14.6	391	12.2	0.370	8.08	0.640
212	12×45	13.1	348	12.1	0.335	8.05	0.575
213	12×40	11.7	307	11.9	0.295	8.01	0.515
214	12×35	10.3	285	12.5	0.300	6.56	0.520
215	12×30	8.79	238	12.3	0.260	6.52	0.440
216	12×26	7.65	204	12.2	0.230	6.49	0.380
217	12×22	6.48	156	12.3	0.260	4.03	0.425
218	12×19	5.57	130	12.2	0.235	4.01	0.350
219	12×16	4.71	103	12.0	0.220	3.99	0.265
220	12×14	4.16	88.6	11.9	0.200	3.97	0.225
221	10×112	32.9	716	11.4	0.755	10.4	1.25
222	10×100	29.3	623	11.1	0.680	10.3	1.12
223	10×88	26.0	534	10.8	0.605	10.3	0.990
224	10×77	22.7	455	10.6	0.530	10.2	0.870
225	10×68	19.9	394	10.4	0.470	10.1	0.770
226	10×60	17.7	341	10.2	0.420	10.1	0.680
227	10×54	15.8	303	10.1	0.370	10.0	0.615
228	10×49	14.4	272	10.0	0.340	10.0	0.560
229	10×45	13.3	248	10.1	0.350	8.02	0.620
230	10×39	11.5	209	9.92	0.315	7.99	0.530
231	10×33	9.71	171	9.73	0.290	7.96	0.435
232	10×30	8.84	170	10.5	0.300	5.81	0.510
233	10×26	7.61	144	10.3	0.260	5.77	0.440
234	10×22	6.49	118	10.2	0.240	5.75	0.360
235	10×19	5.62	96.3	10.2	0.250	4.02	0.395
236	10×17	4.99	81.9	10.1	0.240	4.01	0.330
237	10×15	4.41	68.9	9.99	0.230	4.00	0.270
238	10×12	3.54	53.8	9.87	0.190	3.96	0.210
239	8×67	19.7	272	9.00	0.570	8.28	0.935
240	8×58	17.1	228	8.75	0.510	8.22	0.810
241	8×48	14.1	184	8.50	0.400	8.11	0.685
242	8×40	11.7	146	8.25	0.360	8.07	0.560
243	8×35	10.3	127	8.12	0.310	8.02	0.495
244	8×31	9.13	110	8.00	0.285	8.00	0.435

Continuation of Table A.1							
Section ID	Section Size	Area (in^2)	Moment of Inertia (in^4)	Depth (in)	Web thickness	Flange width	Flange thickness
245	8×28	8.25	98.0	8.06	0.285	6.54	0.465
246	8×24	7.08	82.7	7.93	0.245	6.50	0.400
247	8×21	6.16	75.3	8.28	0.250	5.27	0.400
248	8×18	5.26	61.9	8.14	0.230	5.25	0.330
249	8×15	4.44	48.0	8.11	0.245	4.02	0.315
250	8×13	3.84	39.6	7.99	0.230	4.00	0.255
251	8×10	2.96	30.8	7.89	0.170	3.94	0.205
252	6×25	7.34	53.4	6.38	0.320	6.08	0.455
253	6×20	5.87	41.4	6.20	0.260	6.02	0.365
254	6×15	4.43	29.1	5.99	0.230	5.99	0.260
255	6×16	4.74	32.1	6.28	0.260	4.03	0.405
256	6×12	3.55	22.1	6.03	0.230	4.00	0.280
257	6×9	2.68	16.4	5.90	0.170	3.94	0.215
258	6×8.5	2.52	14.9	5.83	0.170	3.94	0.195
259	5×19	5.56	26.3	5.15	0.270	5.03	0.430
260	5×16	4.71	21.4	5.01	0.240	5.00	0.360
261	4×13	3.83	11.3	4.16	0.280	4.06	0.345

Appendix B

Source Codes for the PBD Optimization

The following are the codes used in the applications in Chapter 4.

B.1 THA.py

```
import openseespy.opensees as op
import math
import numpy as np

def linear(Design_Alt, ID, nby, nst, Lb, Lc, ground_motion):
    op.wipe()
    op.model('basic', '-ndm', 2, '-ndf', 3)
    g = 386.1 # in/sec^2
    inch = 1
    lb = 1
    kips = 1000*lb
    ft = 12*inch
    Failure = 0
    dt = 0.02
    tFinal = 20

    w = 9*kips/ft
    mass = Lb*w/2/g

    # /// outer column ID (nst), inner column ID (nst), beam ID (nst) ///
    outerID = []
    innerID = []
    beamID = []
```

```
for i in range(nst):
    outerID.append(Design_Alt[i])
    innerID.append(Design_Alt[i+nst])
    beamID.append(Design_Alt[i+nst*2])

# --------- Nodes and constraints ---------
for i in range(nst+1):
    for j in range(nby+1):
        op.node(i*(nby+1)+j+1, j*Lb, i*Lc)
        if i == 0:
            op.fix(j+1, 1, 1, 1)
        else:
            op.fix(i*(nby+1)+j+1, 0, 0, 0)
# --------- Mass ---------
for i in range(nst):
    for j in range(nby+1):
        if j == 0 or j == nby:
            op.mass((i+1)*(nby+1)+j+1, mass, 0, 0)
        else:
            op.mass((i+1)*(nby+1)+j+1, 2*mass, 0, 0)

# --------- Material properties ---------
steel = 1
Fy = 36e3 # psi
Es = 29000e3 # psi

# op.uniaxialMaterial('Steel01', steel, Fy, Es, 0.10, 0.05, 1.00, 0.05, 1.00)
op.uniaxialMaterial('Steel02', steel, Fy, Es, 0.10, 18.0, 0.925, 0.15, 0.05, 1.00, 0.05,
    1.00, 0)

# --------- Define elements ---------
coltranTag = 1
beamtranTag = 2
op.geomTransf('Linear', coltranTag)
op.geomTransf('Linear', beamtranTag)

eleTag = 0

for i in range(nst):
    for j in range(nby+1):
        eleTag += 1
        if j == 0 or j == nby:
            op.element('elasticBeamColumn', eleTag, i*(nby+1)+j+1, (i+1)*(nby+1)+j+1,
                ID[outerID[i]][2]*inch**2, Es, ID[outerID[i]][3]*inch**4, 1, '-mass', ID[
```

```
                    outerID[i ]][1]  * lb/ft/g) # outer column
            else:
                op.element('elasticBeamColumn', eleTag, i*(nby+1)+j+1, (i+1)*(nby+1)+j+1,
                    ID[innerID[i]][2]*inch**2, Es, ID[innerID[i]][3]*inch**4, 1, '-mass', ID[
                    innerID[i ]][1]  * lb/ft/g) # inner column
for i in range(nst):
    for j in range(nby):
        eleTag += 1
        op.element('elasticBeamColumn', eleTag, (i+1)*(nby+1)+j+1, (i+1)*(nby+1)+j
            +2, ID[beamID[i]][2]*inch**2, Es, ID[beamID[i]][3]*inch**4, 2, '-mass', ID[
            beamID[i]][1] * lb/ft/g) # beam

# --------- Define gravity loads ---------
op.timeSeries('Constant', 1)
op.pattern('Plain', 1, 1)
beamnumTag = 0
for i in range(nst):
    for j in range(nby):
        beamnumTag += 1
        op.eleLoad('-ele', (nby+1)*nst+beamnumTag, '-type', '-beamUniform', -w, 0,
            0)

Tol = 1e-8 # convergence tolerance for test
NstepGravity = 10
DGravity = 1/NstepGravity
op.integrator('LoadControl', DGravity) # determine the next time step for an analysis
op.numberer('Plain') # renumber dof's to minimize band-width (optimization), if you
    want to
op.system('BandGeneral') # how to store and solve the system of equations in the analysis
op.constraints('Plain') # how it handles boundary conditions
op.test('NormDispIncr', Tol, 6) # determine if convergence has been achieved at the end
    of an iteration step
op.algorithm('Newton') # use Newton's solution algorithm: updates tangent stiffness at
    every iteration
op.analysis('Static') # define type of analysis static or transient
op.analyze(NstepGravity) # apply gravity
op.loadConst('-time', 0.0) # maintain constant gravity loads and reset time to zero

Lambda = op.eigen('-fullGenLapack', 3) # eigenvalue mode 1
Omega = math.pow(Lambda[0], 0.5)

# --------- TO CHECK FUNDAMENTAL PERIODS! ---------
# print('Fundamental Period = ', 2*np.pi/Omega, 'sec')
# Omega1 = math.pow(Lambda[1], 0.5)
```

```
# print('Fundamental Period = ', 2 * np.pi / Omega1, 'sec')
# Omega2 = math.pow(Lambda[2], 0.5)
# print('Fundamental Period = ', 2 * np.pi / Omega2, 'sec')

betaKcomm = 2 * (0.02 / Omega)

op.rayleigh (0.0,  0.0,  0.0,  betaKcomm) # RAYLEIGH damping

GMdirection = 1
IDloadTag = 2 # load tag
op.timeSeries('Path',  2,  '-dt',  dt,  '-filePath',  ground_motion, '-factor',  g)
op.pattern('UniformExcitation', IDloadTag, GMdirection, '-accel', 2)

# --------- Linear analysis ---------
op.wipeAnalysis()
op.constraints ('Plain')
op.numberer('Plain')
op.system('BandGeneral')
op.test ('NormDispIncr', 1e-8, 10)
op.algorithm('Newton')
op.integrator ('Newmark', 0.5, 0.25)
op.analysis ('Transient')

tCurrent = op.getTime()
t = [tCurrent]
drift = []
for i in range(nst):
    drift.append([0.0])
large_disp_tol = 20 # stop inch
while tCurrent < tFinal:
    op.analyze(1, dt)
    tCurrent = op.getTime()
    t.append(tCurrent)
    for i in range(nst):
        drift[i].append(op.nodeDisp((nby + 1) * (i+1) + 1, 1) - op.nodeDisp((nby + 1)
            * i + 1, 1))
        if np.absolute(
                op.nodeDisp((nby + 1) * (i + 1) + 1, 1) - op.nodeDisp((nby + 1) * i + 1,
                    1)) > large_disp_tol:
            Failure = 1
            return 0, 0, Failure
print (' drift _max=', np.max(np.absolute(drift), axis=1))
return t, drift, Failure
```

```
def nonlinear(Design_Alt, ID, nby, nst, Lb, Lc, ground_motion):
    op.wipe()
    op.model('basic', '-ndm', 2, '-ndf', 3)
    g = 386.1 # in/sec^2
    inch = 1
    lb = 1
    kips = 1000*lb
    ft = 12*inch
    Failure = 0
    dt = 0.02
    tFinal = 20

    w = 9*kips/ft
    mass = Lb*w/2/g

    # /// outer column ID (nst), inner column ID (nst), beam ID (nst) ///
    outerID = []
    innerID = []
    beamID = []
    for i in range(nst):
        outerID.append(Design_Alt[i])
        innerID.append(Design_Alt[i+nst])
        beamID.append(Design_Alt[i+nst*2])

    # --------- Nodes and constraints ---------
    for i in range(nst+1):
        for j in range(nby+1):
            op.node(i*(nby+1)+j+1, j*Lb, i*Lc)
            if i == 0:
                op.fix(j+1, 1, 1, 1)
            else:
                op.fix(i*(nby+1)+j+1, 0, 0, 0)
    # --------- Mass ---------
    for i in range(nst):
        for j in range(nby+1):
            if j == 0 or j == nby:
                op.mass((i+1)*(nby+1)+j+1, mass, 0, 0)
            else:
                op.mass((i+1)*(nby+1)+j+1, 2*mass, 0, 0)

    # --------- Material properties ---------
    steel = 1
    Fy = 36e3 # psi
    Es = 29000e3 # psi
```

```
# op.uniaxialMaterial('Steel01', steel, Fy, Es, 0.10, 0.05, 1.00, 0.05, 1.00)
op.uniaxialMaterial('Steel02', steel, Fy, Es, 0.10, 18.0, 0.925, 0.15, 0.05, 1.00, 0.05,
    1.00, 0)

# ---------- Define sections (WFSection2d) ----------
outcolSecTag = 1
incolSecTag = 2
beamSecTag = 3
for i in range(nst):
    outcolSecTag += 10
    incolSecTag += 10
    beamSecTag += 10
    # /// section ('WFSection2d', SecTag, matTag, d, tw, bf, tf, Nfw, Nff) ///
    Nfw = 15
    Nff = 15
    op.section('WFSection2d', outcolSecTag, steel, ID[outerID[i]][4]*inch, ID[outerID[i
        ]][5]*inch, ID[outerID[i]][6]*inch, ID[outerID[i]][7]*inch, Nfw, Nff)  # outer
        Column
    op.section('WFSection2d', incolSecTag, steel, ID[innerID[i]][4]*inch, ID[innerID[i
        ]][5]*inch, ID[innerID[i]][6]*inch, ID[innerID[i]][7]*inch, Nfw, Nff)  # inner
        Column
    op.section('WFSection2d', beamSecTag, steel, ID[beamID[i]][4]*inch, ID[beamID[i]][5]*
        inch, ID[beamID[i]][6]*inch, ID[beamID[i]][7]*inch, Nfw, Nff)  # beam
    op.beamIntegration('Lobatto', outcolSecTag, outcolSecTag, 4)  # outer Column
    op.beamIntegration('Lobatto', incolSecTag, incolSecTag, 4)  # inner Column
    op.beamIntegration('Lobatto', beamSecTag, beamSecTag, 4) # beam

# ---------- Define coordinate-transformation ----------
coltranTag = 1
beamtranTag = 2
op.geomTransf('PDelta', coltranTag)
op.geomTransf('Linear', beamtranTag)

# ---------- Define nonlinear elements ----------
eleTag = 0
numIntPts = 5
for i in range(nst):
    for j in range(nby+1):
        eleTag += 1
        if j == 0 or j == nby:
            op.element('nonlinearBeamColumn', eleTag, i*(nby+1)+j+1, (i+1)*(nby+1)+j
                +1, numIntPts, (i+1)*10+1, coltranTag, '-mass', ID[outerID[i]][1] * lb/ft/
                g) # outer Column
```

```
            else:
                op.element('nonlinearBeamColumn', eleTag, i*(nby+1)+j+1, (i+1)*(nby+1)+j
                    +1, numIntPts, (i+1)*10+2, coltranTag, '-mass', ID[innerID[i]][1] * lb/ft/
                    g) # inner Column
    for i in range(nst):
        for j in range(nby):
            eleTag += 1
            op.element('nonlinearBeamColumn', eleTag, (i+1)*(nby+1)+j+1, (i+1)*(nby+1)+j
                +2, numIntPts, (i+1)*10+3, beamtranTag, '-mass', ID[beamID[i]][1] * lb/ft/g)
                 # beam

    # --------- Define gravity loads ---------
    op.timeSeries('Constant', 1)
    op.pattern('Plain', 1, 1)
    beamnumTag = 0
    for i in range(nst):
        for j in range(nby):
            beamnumTag += 1
            op.eleLoad('-ele', (nby+1)*nst+beamnumTag, '-type', '-beamUniform', -w, 0,
                0)

    Tol = 1e-8 # convergence tolerance for test
    NstepGravity = 10
    DGravity = 1/NstepGravity
    op.integrator('LoadControl', DGravity) # determine the next time step for an analysis
    op.numberer('Plain') # renumber dof's to minimize band-width (optimization), if you
        want to
    op.system('BandGeneral') # how to store and solve the system of equations in the analysis
    op.constraints('Plain') # how it handles boundary conditions
    op.test('NormDispIncr', Tol, 6) # determine if convergence has been achieved at the end
        of an iteration step
    op.algorithm('Newton') # use Newton's solution algorithm: updates tangent stiffness at
        every iteration
    op.analysis('Static') # define type of analysis static or transient
    op.analyze(NstepGravity) # apply gravity
    op.loadConst('-time', 0.0) # maintain constant gravity loads and reset time to zero

    Lambda = op.eigen('-fullGenLapack', 3) # eigenvalue mode 1
    Omega = math.pow(Lambda[0], 0.5)

    # --------- TO CHECK FUNDAMENTAL PERIODS! ---------
    # print('Fundamental Period = ', 2*np.pi/Omega, 'sec')
    # Omega1 = math.pow(Lambda[1], 0.5)
    # print('Fundamental Period = ', 2 * np.pi / Omega1, 'sec')
```

```
# Omega2 = math.pow(Lambda[2], 0.5)
# print('Fundamental Period = ', 2 * np.pi / Omega2, 'sec')

betaKcomm = 2 * (0.02 / Omega)

op.rayleigh (0.0,  0.0,  0.0,  betaKcomm) # RAYLEIGH damping

maxNumIter = 10
GMdirection = 1
IDloadTag = 2 # load tag
op.timeSeries('Path', 2, '-dt', dt, '-filePath', ground_motion, '-factor', g)
op.pattern('UniformExcitation', IDloadTag, GMdirection, '-accel', 2)

# --------- Nonlinear Analysis ---------
op.wipeAnalysis()
op.constraints ('Transformation')
op.numberer('RCM')
op.system('BandGeneral')

test = {1: 'NormDispIncr', 2: 'RelativeEnergyIncr', 3: 'EnergyIncr', 4: '
    RelativeNormUnbalance',
        5: 'RelativeNormDispIncr', 6: 'NormUnbalance'}
algorithm = {1: 'KrylovNewton', 2: 'SecantNewton', 3: 'ModifiedNewton', 4: '
    RaphsonNewton', 5: 'PeriodicNewton',
            6: 'BFGS', 7: 'Broyden', 8: 'NewtonLineSearch'}

op.test(test [1], Tol, maxNumIter)
op.algorithm(algorithm[1], '- initial ')
op.integrator ('Newmark', 0.5, 0.25)
op.analysis ('Transient')

tCurrent = op.getTime()
t = [tCurrent]
drift = []
for i in range(nst):
    drift .append([0.0])
large_disp_tol = 20 # stop inch
while tCurrent < tFinal:
    op.analyze(1, dt)
    tCurrent = op.getTime()
    t.append(tCurrent)
    for i in range(nst):
        drift [i].append(op.nodeDisp((nby + 1) * (i+1) + 1, 1) - op.nodeDisp((nby + 1)
            * i + 1, 1))
```

```
            if np.absolute(op.nodeDisp((nby + 1) * (i+1) + 1, 1) - op.nodeDisp((nby + 1) * i
                + 1, 1)) > large_disp_tol:
                Failure = 1
                return 0, 0, Failure
    print('drift_max=', np.max(np.absolute(drift), axis=1))
    return t, drift, Failure
```

B.2 GM.py

```
import numpy as np

def sigma_setting():
    ts = 30
    TL = 5
    T_end = 20
    dt = 0.02
    zeta = 0.02
    Sa_data = np.loadtxt('Sa_data.dat')
    T = Sa_data[:, 0]
    dw = 2 * np.pi / ts
    w = np.linspace(2 * np.pi / TL, 2 * np.pi / TL + dw * 2244, 2245)

    k = 4 * zeta / ((np.pi - 4 * zeta) * w + 4 * zeta * dw)
    deltax = np.sqrt(1 - (1 - 2 / np.pi * np.arctan(zeta / np.sqrt(1 - zeta ** 2))) ** 2 / (1
        - zeta ** 2))

    Nx = ts / 2 / np.pi * w / (-np.log(0.5))
    eta = np.sqrt(2 * np.log(2 * Nx * (1 - np.exp(-deltax ** (1.2) * np.sqrt(np.pi * np.log(2
        * Nx))))))

    K = np.zeros([len(k), len(k)])
    ks = np.zeros([1, len(k)])

    sigma = np.zeros((11, 2245, 1000))
    t = np.linspace(0, T_end - dt, int(T_end / dt))
    beta = 9 / 10
    t1 = 2.5 / beta
    t2 = 11.5 / beta
    phi = np.exp(-beta * (t - t2))
    i = 0
    while t[i] <= t1:
        phi[i] = (t[i] / t1) ** 2
        i = i + 1
```

```
    while t[i] <= t2:
        phi[i] = 1
        i = i + 1

    for IM in range(11):
        S = Sa_data[:, IM + 1]
        for i in range(len(w)):
            period = 2 * np.pi / w[i]
            findS = 0
            while period > T[findS]:
                findS = findS + 1
            Sa = (S[findS] - S[findS - 1]) / (T[findS] - T[findS - 1]) * (period - T[findS -
                1]) + S[findS - 1]
            ks[0][i] = k[i] * Sa ** 2 / eta[i] ** 2
            K[i][0:i] = np.ones([1, i]) * k[i]

        K = (K - np.diag(k)) * dw + np.eye(len(k))
        G = np.matmul(np.linalg.inv(K), ks.T)
        sigma[IM] = np.sqrt(G * phi * dw)
    return sigma, w, t

def generator(sigma, w, t):
    init_time = time.time()
    uc = np.random.normal(0, 1, len(w))
    us = np.random.normal(0, 1, len(w))
    usigmac = np.multiply(np.outer(np.ones([len(t), 1]), uc), sigma.T)
    usigmas = np.multiply(np.outer(np.ones([len(t), 1]), us), sigma.T)
    csin = np.cos(np.outer(t, w))
    sine = np.sin(np.outer(t, w))
    ground_motion = np.sum(np.multiply(csin, usigmac) + np.multiply(sine, usigmas), axis=1)
    np.savetxt('ground_motion.dat', ground_motion.reshape((200, 5)))
    return ground_motion
```

Appendix C

Source Codes for the Structural Model Updating

The following are the codes used in the applications in Chapter 5.

C.1 SMU.py

```
import numpy as np
import pandas as pd
import abaqus_run as abaq
import abaqus_create as abaq_model_create
import time
import xlsxwriter

# Genetic Algorithm parameters:
Gen_num = 5
popul = 30
survive_rate = 0.2
offspring_rate = 0.5
CO_rate = 0.5
RM_rate = 0.05
rand_bound = 0.005
Neval = Gen_num * (1 - survive_rate) * popul + survive_rate * popul

# Ground truth
f_out = 2.00
f_in = 2.63

GT = [f_out, f_in]
```

```
# Objective function
def mse(Periods, GT):
    return ((Periods[0]-GT[0])**2+(Periods[1]-GT[1])**2)**(0.5)

# parameter means
params_open = open("params_origin.txt", 'r')
params = []
for i in params_open.read().split():
    params.append(float(i))
params_open.close()

cov_open = open("cov.txt", 'r')
cov = []
for i in cov_open.read().split():
    cov.append(float(i))
cov_open.close()

Nsigma = 1
params_ub = params + Nsigma*np.multiply(cov, params)
params_lb = params - Nsigma*np.multiply(cov, params)

init = time.time()
Gen = []
for k in range(popul):
    Gen.append(params_lb + np.multiply(np.random.uniform(0,1,len(params_ub)), params_ub-
        params_lb))
evalnum = 0
best = [0, 0, 100]
recorder_params = []
recorder_periods = []
abaq_model_create.overhead_sing_but_create()

for k in range(popul):
    abaq.abaqus_run(Gen[k])
    df = pd.read_fwf('abaqus.rpt', skiprows=[1], nrows=150)
    dfi = df.tail(5).head(2)
    perioddf = dfi['EIGFREQ Whole']
    Modeperiods = [float(perioddf.values[0]), float(perioddf.values[1])]
    if Modeperiods == 0:
        error = 10
    else:
        Periods = Modeperiods[0:1]
        error = obj.mse(Periods, GT)
    Gen[k] = np.append(Gen[k], error)
```

```
    evalnum += 1
    current_time = time.time() - init
    print("time:",current_time, " estimated time the task ends:", current_time/evalnum*
        Neval)
    print("evaluation",evalnum, " => MSE:", Gen[k][-1], ", Out-of-plane:", Periods[0], "(sec
        ) In-plane:", Periods[1],"(sec)")
    if Gen[k][-1] < best[-1]:
        best = []
        best = [Periods[0], Periods[1], Gen[k][-1]]
        best_params = Gen[k]
        best_periods = Periods
    print("(current best) MSE:", best[-1], ", Out-of-plane:", best[0], "(sec) In-plane:",
        best[1],"( sec)")
    recorder_params.append(best_params)
    recorder_periods.append(best_periods)

def error_sort( list ):
    return list [len(params_ub)]
Gen.sort(key=error_sort)
Gen_best = [Gen[0]]
Nsurvive = round(popul * survive_rate)

def CrossOver(Gen, N, CO_rate, RM_rate, rand_bound, params_ub, params_lb):
    offsprings = []
    for i in range(N):
        parent1 = np.random.randint(0,len(Gen))
        parent2 = np.random.randint(0,len(Gen))
        p = np.random.uniform(0,1)
        # random searching near the best candidate:
        if p <= RM_rate:
            parent1 = 0
            candidate = Gen[parent1] + np.multiply(np.random.uniform(-rand_bound, rand_
                bound, len(Gen[parent1])), Gen[parent1])
            for i in range(len(params_ub)):
                if candidate[i] > params_ub[i]:
                    candidate[i] = params_ub[i]
                if candidate[i] < params_ub[i]:
                    candidate[i] = params_lb[i]
        else:
            while parent1 == parent2:
                parent2 = np.random.randint(0,len(Gen))
            multip = []
            for i in range(len(Gen[parent1])):
                if i < len(Gen[parent1])*CO_rate:
```

```
                    multip.append(random.uniform(0,1))
                else:
                    multip.append(random.randint(0,1))
            random.shuffle(multip)
            candidate = Gen[parent1] + np.multiply(multip, Gen[parent2]-Gen[parent1])
        offsprings.append(candidate)
    return offsprings

for N in range(Gen_num-1):
    del Gen[Nsurvive:popul]
    offsprings = CrossOver(Gen, round(popul * offspring_rate), CO_rate, RM_rate, rand_bound
        , params_ub, params_lb)
    eval_N = popul - Nsurvive
    for k in range(round(popul * offspring_rate)):
        offsprings[k] = np.delete(offsprings[k], -1)
        Gen.append(offsprings[k])
    for k in range(popul - Nsurvive - round(popul * offspring_rate)):
        Gen.append(params_lb + np.multiply(np.random.uniform(0, 1, len(params_ub)),
            params_ub - params_lb))
    for k in range(eval_N):
        abaq.abaqus_run(Gen[k + popul - eval_N])
        df = pd.read_fwf('abaqus.rpt', skiprows=[1], nrows=150)
        dfi = df.tail(5).head(2)
        perioddf = dfi['EIGFREQ Whole']
        Modeperiods = [float(perioddf.values[0]), float(perioddf.values[1])]
        if Modeperiods == 0:
            error = 10
        else:
            Periods = Modeperiods[0:3]
            mod_num = Modeperiods[3:6]
            error = obj.mse(Periods, GT)
        Gen[k + popul - eval_N] = np.append(Gen[k + popul - eval_N], error)
        evalnum += 1
        current_time = time.time() - init
        print("time:", current_time, " estimated time the task ends:", current_time /
            evalnum * Neval)
        print("evaluation", evalnum, " => MSE:", Gen[k + popul - eval_N][-1], ", Out-of-
            plane:", Periods[0], "(sec), In-plane:", Periods[1],"(sec)")
        if Gen[k + popul - eval_N][-1] < best[-1]:
            best = []
            best = [Periods[0], Periods[1], Gen[k + popul - eval_N][-1]]
            best_params = Gen[k + popul - eval_N]
            best_periods = Periods
```

```
            print("(current best) MSE:", best[-1], ", Out-of-plane:", best[0], "(sec), In-plane
                :", best[1],"(sec)")
            recorder_params.append(best_params)
            recorder_periods.append(best_periods)
        Gen.sort(key=error_sort)
        params_save_overwrite = open("params_save.txt", 'w')
        for p in Gen[0]:
            params_save_overwrite.writelines(str(p))
            params_save_overwrite.writelines("\n")
        params_save_overwrite.close()
        Gen_best.append(Gen[0])
print(Gen_best)
workbook = xlsxwriter.Workbook('periods_results.xlsx')
worksheet = workbook.add_worksheet()
workbook2 = xlsxwriter.Workbook('params_results.xlsx')
worksheet2 = workbook2.add_worksheet()
for i in range(evalnum):
    for j in range(len(Periods)):
        worksheet.write(i, j, recorder_periods[i][j])
    for j in range(len(Gen[0])):
        worksheet2.write(i, j, recorder_params[i][j])
workbook.close()
workbook2.close()

abaq.abaqus_run(Gen[0])
df = pd.read_fwf('abaqus.rpt', skiprows=[1], nrows=150)
dfi = df.tail(5).head(2)
perioddf = dfi['EIGFREQ Whole']
Periods = [float(perioddf.values[0]), float(perioddf.values[1])]
print("MSE:",Gen[0][-1], ", Out-of-plane:", Periods[0],"(sec), In-plane:", Periods[1],"(sec)
    ")
```

C.2 abaqus_create.py

```
from abaqus import *
from abaqusConstants import *
import __main__

def overhead_sing_but_create():
    import section
    import regionToolset
    import displayGroupMdbToolset as dgm
    import part
```

```
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=50.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.Line(point1=(4.25, 8.0), point2=(4.25, -8.0))
s.VerticalConstraint(entity=g[2], addUndoState=False)
s.Line(point1=(4.25, -8.0), point2=(-4.25, -8.0))
s.HorizontalConstraint(entity=g[3], addUndoState=False)
s.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)
s.Line(point1=(-4.25, -8.0), point2=(-4.25, 8.0))
s.VerticalConstraint(entity=g[4], addUndoState=False)
s.PerpendicularConstraint(entity1=g[3], entity2=g[4], addUndoState=False)
s.Line(point1=(-4.25, 8.0), point2=(4.25, 8.0))
s.HorizontalConstraint(entity=g[5], addUndoState=False)
s.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)
s.Line(point1=(4.25, 4.25), point2=(9.25, 4.25))
s.HorizontalConstraint(entity=g[6], addUndoState=False)
s.Line(point1=(9.25, 4.25), point2=(9.25, -4.25))
s.VerticalConstraint(entity=g[7], addUndoState=False)
s.PerpendicularConstraint(entity1=g[6], entity2=g[7], addUndoState=False)
s.Line(point1=(9.25, -4.25), point2=(4.25, -4.25))
s.HorizontalConstraint(entity=g[8], addUndoState=False)
s.PerpendicularConstraint(entity1=g[7], entity2=g[8], addUndoState=False)
s.Line(point1=(-4.25, 4.25), point2=(-9.25, 4.25))
s.HorizontalConstraint(entity=g[9], addUndoState=False)
s.Line(point1=(-9.25, 4.25), point2=(-9.25, -4.25))
s.VerticalConstraint(entity=g[10], addUndoState=False)
s.PerpendicularConstraint(entity1=g[9], entity2=g[10], addUndoState=False)
s.Line(point1=(-9.25, -4.25), point2=(-4.25, -4.25))
s.HorizontalConstraint(entity=g[11], addUndoState=False)
s.PerpendicularConstraint(entity1=g[10], entity2=g[11], addUndoState=False)
p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=THREE_D,
```

```
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-1']
p.BaseShellExtrude(sketch=s, depth=186.0)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#a0 ]', ), )
p.Set(faces=faces, name='plate_A')
session.viewports['Viewport: 1'].view.setValues(nearPlane=333.254,
    farPlane=488.424, width=175.117, height=68.6684, viewOffsetX=-13.3524,
    viewOffsetY=-4.66995)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#5f ]', ), )
p.Set(faces=faces, name='plate_B')
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#3f00 ]', ), )
p.Set(faces=faces, name='plate_C')
session.viewports['Viewport: 1'].view.setValues(nearPlane=326.135,
    farPlane=495.544, width=264.275, height=103.63, viewOffsetX=4.82306,
    viewOffsetY=-5.13316)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=186.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=-15.0)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[6], sketchUpEdge=e[29],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(-15.0, 0.0,
    93.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=413.78, gridSpacing=10.34, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
s1.Line(point1=(93.0, 18.0), point2=(93.0, -18.0))
s1.VerticalConstraint(entity=g[2], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
```

```
p.ShellExtrude(sketchPlane=d2[6], sketchUpEdge=e1[29], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1, depth=30.0,
    flipExtrudeDirection=OFF)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=302.408,
    farPlane=524.299, width=245.049, height=96.0906, cameraPosition=(
    233.917, 163.41, 392.273), cameraUpVector=(-0.57735, 0.70551,
    -0.411001), cameraTarget=(-3.28079, -5.44579, 102.421),
    viewOffsetX=4.47218, viewOffsetY=-4.75972)
session.viewports['Viewport: 1'].view.setValues(nearPlane=303.441,
    farPlane=524.554, width=245.886, height=96.4187, cameraPosition=(
    233.917, 163.41, 393.185), cameraTarget=(-3.28079, -5.44579, 103.333),
    viewOffsetX=4.48745, viewOffsetY=-4.77597)
session.viewports['Viewport: 1'].view.setValues(session.views['Front'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=310.788,
    farPlane=533.186, width=103.085, height=40.4225)
session.viewports['Viewport: 1'].view.setValues(nearPlane=310.978,
    farPlane=532.997, width=103.148, height=40.4472, cameraPosition=(
    4.9463, 0.333363, 514.987), cameraTarget=(4.9463, 0.333363, 93))
session.viewports['Viewport: 1'].view.setValues(cameraPosition=(0.666995,
    0.166684, 514.987), cameraTarget=(0.666995, 0.166684, 93))
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#f ]', ), )
p.Set(faces=faces, name='base_plate')
session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=-12.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=-28.0)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[10], sketchUpEdge=e[8],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(-28.0, 0.0,
    93.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=426.19, gridSpacing=10.65, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=382.012,
    farPlane=470.369, width=244.125, height=95.7282, cameraPosition=(
```

```
    420.44, 4.41718, 141.748), cameraTarget=(-5.75, 4.41718, 141.748))
s.Line(point1=(-93.0, -8.0), point2=(-93.0, -12.0))
s.VerticalConstraint(entity=g[2], addUndoState=False)
s.Line(point1=(-93.0, -12.0), point2=(-97.0, -12.0))
s.HorizontalConstraint(entity=g[3], addUndoState=False)
s.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)
s.Line(point1=(-93.0, 8.0), point2=(-93.0, 12.0))
s.VerticalConstraint(entity=g[4], addUndoState=False)
s.Line(point1=(-93.0, 12.0), point2=(-97.0, 12.0))
s.HorizontalConstraint(entity=g[5], addUndoState=False)
s.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)
s.Line(point1=(-167.0, -12.0), point2=(-171.0, -12.0))
s.HorizontalConstraint(entity=g[6], addUndoState=False)
s.Line(point1=(-171.0, -12.0), point2=(-171.0, -8.0))
s.VerticalConstraint(entity=g[7], addUndoState=False)
s.PerpendicularConstraint(entity1=g[6], entity2=g[7], addUndoState=False)
s.Line(point1=(-167.0, 12.0), point2=(-171.0, 12.0))
s.HorizontalConstraint(entity=g[8], addUndoState=False)
s.Line(point1=(-171.0, 12.0), point2=(-171.0, 8.0))
s.VerticalConstraint(entity=g[9], addUndoState=False)
s.PerpendicularConstraint(entity1=g[8], entity2=g[9], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=354.785,
    farPlane=661.689, width=361.612, height=141.798, viewOffsetX=27.1921,
    viewOffsetY=10.4438)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[10], sketchUpEdge=e1[8], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s, depth=272.0,
    flipExtrudeDirection=OFF)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=168.304,
    farPlane=589.229, width=323.918, height=127.017, viewOffsetX=17.5033,
    viewOffsetY=-11.3685)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[5], sketchUpEdge=e[66],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, 0.0,
    186.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
```

```
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=684.279,
    farPlane=991.052, width=91.1398, height=35.7385, cameraPosition=(
    -7.13129, -5.30098, 969.666), cameraTarget=(-7.13129, -5.30098, 132))
s1.Line(point1=(136.0, 8.0), point2=(136.0, 12.0))
s1.VerticalConstraint(entity=g[30], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[13], entity2=g[30], addUndoState=False)
s1.Line(point1=(136.0, 12.0), point2=(132.0, 12.0))
s1.HorizontalConstraint(entity=g[31], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[30], entity2=g[31], addUndoState=False)
s1.Line(point1=(136.0, -8.0), point2=(136.0, -12.0))
s1.VerticalConstraint(entity=g[32], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[6], entity2=g[32], addUndoState=False)
s1.Line(point1=(136.0, -12.0), point2=(132.0, -12.0))
s1.HorizontalConstraint(entity=g[33], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[32], entity2=g[33], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=673.548,
    farPlane=1001.78, width=210.611, height=82.5865, cameraPosition=(
    199.958, -19.4873, 969.666), cameraTarget=(199.958, -19.4873, 132))
s1.Line(point1=(-136.0, 8.0), point2=(-136.0, 12.0))
s1.VerticalConstraint(entity=g[34], addUndoState=False)
s1.ParallelConstraint(entity1=g[10], entity2=g[34], addUndoState=False)
s1.Line(point1=(-136.0, 12.0), point2=(-132.0, 12.0))
s1.HorizontalConstraint(entity=g[35], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[34], entity2=g[35], addUndoState=False)
s1.Line(point1=(-136.0, -8.0), point2=(-136.0, -12.0))
s1.VerticalConstraint(entity=g[36], addUndoState=False)
s1.ParallelConstraint(entity1=g[3], entity2=g[36], addUndoState=False)
s1.Line(point1=(-136.0, -12.0), point2=(-132.0, -12.0))
s1.HorizontalConstraint(entity=g[37], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[36], entity2=g[37], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[5], sketchUpEdge=e1[66], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1, depth=78.0,
    flipExtrudeDirection=OFF)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=186.578,
    farPlane=570.955, width=117.897, height=46.2308, viewOffsetX=-9.29489,
    viewOffsetY=-15.4269)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[9], sketchUpEdge=e[26],
```

```
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, -12.0,
    132.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=772.572,
    farPlane=902.76, width=504.259, height=197.734, cameraPosition=(
    64.2693, 837.666, 202.705), cameraTarget=(64.2693, 0, 202.705))
s.Line(point1=(-133.0, 132.0), point2=(-136.0, 132.0))
s.HorizontalConstraint(entity=g[26], addUndoState=False)
s.Line(point1=(-136.0, 132.0), point2=(-136.0, 129.0))
s.VerticalConstraint(entity=g[27], addUndoState=False)
s.PerpendicularConstraint(entity1=g[26], entity2=g[27], addUndoState=False)
s.Line(point1=(-136.0, 57.0), point2=(-136.0, 54.0))
s.VerticalConstraint(entity=g[28], addUndoState=False)
s.Line(point1=(-136.0, 54.0), point2=(-133.0, 54.0))
s.HorizontalConstraint(entity=g[29], addUndoState=False)
s.PerpendicularConstraint(entity1=g[28], entity2=g[29], addUndoState=False)
s.Line(point1=(136.0, 129.0), point2=(136.0, 132.0))
s.VerticalConstraint(entity=g[30], addUndoState=False)
s.Line(point1=(136.0, 132.0), point2=(133.0, 132.0))
s.HorizontalConstraint(entity=g[31], addUndoState=False)
s.PerpendicularConstraint(entity1=g[30], entity2=g[31], addUndoState=False)
s.Line(point1=(133.0, 54.0), point2=(136.0, 54.0))
s.HorizontalConstraint(entity=g[32], addUndoState=False)
s.Line(point1=(136.0, 54.0), point2=(136.0, 57.0))
s.VerticalConstraint(entity=g[33], addUndoState=False)
s.PerpendicularConstraint(entity1=g[32], entity2=g[33], addUndoState=False)
s.Line(point1=(122.0, 54.0), point2=(94.0, 54.0))
s.HorizontalConstraint(entity=g[34], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=112.457,
    farPlane=634.8, width=223.62, height=87.6876, viewOffsetX=35.6452,
    viewOffsetY=-42.2556)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[9], sketchUpEdge=e1[26], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s, depth=24.0,
    flipExtrudeDirection=OFF)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(width=111.179, height=43.5966,
```

```
    viewOffsetX=-9.60005, viewOffsetY=-15.335)
session.viewports['Viewport: 1'].view.setValues(nearPlane=165.351,
    farPlane=596.16, width=98.2147, height=38.5127, cameraPosition=(
    264.575, 88.3961, 479.946), cameraUpVector=(-0.57735, 0.811441,
    -0.0907162), cameraTarget=(-3.10599, 46.3367, 103.732),
    viewOffsetX=-8.48059, viewOffsetY=-13.5468)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=-8.0)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#8 ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[14], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
e, d2 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d2[14], sketchUpEdge=e[49],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, -8.0,
    132.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=782.503,
    farPlane=892.829, width=393.699, height=154.381, cameraPosition=(
    -9.33807, 837.666, 222.259), cameraTarget=(-9.33807, 0, 222.259))
s1.Line(point1=(94.0, 132.0), point2=(122.0, 132.0))
s1.HorizontalConstraint(entity=g[33], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d1 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d1[14], sketchUpEdge=e1[49], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1, depth=16.0,
    flipExtrudeDirection=OFF)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=164.181,
    farPlane=597.329, width=132.878, height=52.1053, viewOffsetX=-2.38148,
    viewOffsetY=-15.4698)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#80000000 ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[14], faces=pickedFaces)
```

```
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#7a #1 ]', ), )
p.Set(faces=faces, name='post_plate')
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[5], sketchUpEdge=e[33],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, 0.0,
    186.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=681.311,
    farPlane=994.02, width=124.185, height=48.6964, cameraPosition=(
    3.86393, -0.63455, 969.666), cameraTarget=(3.86393, -0.63455, 132))
s.Line(point1=(114.0, 6.0), point2=(102.0, 6.0))
s.HorizontalConstraint(entity=g[80], addUndoState=False)
s.Line(point1=(102.0, 6.0), point2=(102.0, -6.0))
s.VerticalConstraint(entity=g[81], addUndoState=False)
s.PerpendicularConstraint(entity1=g[80], entity2=g[81], addUndoState=False)
s.Line(point1=(102.0, -6.0), point2=(114.0, -6.0))
s.HorizontalConstraint(entity=g[82], addUndoState=False)
s.PerpendicularConstraint(entity1=g[81], entity2=g[82], addUndoState=False)
s.Line(point1=(114.0, -6.0), point2=(114.0, 6.0))
s.VerticalConstraint(entity=g[83], addUndoState=False)
s.PerpendicularConstraint(entity1=g[82], entity2=g[83], addUndoState=False)
s.Line(point1=(125.0, -10.5), point2=(122.0, -10.5))
s.HorizontalConstraint(entity=g[84], addUndoState=False)
s.Line(point1=(122.0, -10.5), point2=(122.0, -7.5))
s.VerticalConstraint(entity=g[85], addUndoState=False)
s.PerpendicularConstraint(entity1=g[84], entity2=g[85], addUndoState=False)
s.Line(point1=(125.0, 10.5), point2=(122.0, 10.5))
s.HorizontalConstraint(entity=g[86], addUndoState=False)
s.Line(point1=(122.0, 10.5), point2=(122.0, 7.5))
s.VerticalConstraint(entity=g[87], addUndoState=False)
s.PerpendicularConstraint(entity1=g[86], entity2=g[87], addUndoState=False)
s.Line(point1=(91.0, -10.5), point2=(94.0, -10.5))
s.HorizontalConstraint(entity=g[88], addUndoState=False)
s.Line(point1=(94.0, -10.5), point2=(94.0, -7.5))
s.VerticalConstraint(entity=g[89], addUndoState=False)
s.PerpendicularConstraint(entity1=g[88], entity2=g[89], addUndoState=False)
```

```
s.Line(point1=(91.0, 10.5), point2=(94.0, 10.5))
s.HorizontalConstraint(entity=g[90], addUndoState=False)
s.Line(point1=(94.0, 10.5), point2=(94.0, 7.5))
s.VerticalConstraint(entity=g[91], addUndoState=False)
s.PerpendicularConstraint(entity1=g[90], entity2=g[91], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[5], sketchUpEdge=e1[33], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s, depth=78.0,
    flipExtrudeDirection=OFF)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=163.914,
    farPlane=597.597, width=110.187, height=47.5922, viewOffsetX=-12.0729,
    viewOffsetY=-6.59232)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=-9.0)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[20], sketchUpEdge=e[23],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, -9.0,
    132.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=790.256,
    farPlane=885.076, width=279.06, height=120.532, cameraPosition=(
    4.08381, 837.666, 230.022), cameraTarget=(4.08381, 0, 230.022))
s1.Line(point1=(94.0, 127.0), point2=(94.0, 121.0))
s1.VerticalConstraint(entity=g[2], addUndoState=False)
s1.Line(point1=(94.0, 72.0), point2=(94.0, 59.0))
s1.VerticalConstraint(entity=g[3], addUndoState=False)
s1.Line(point1=(122.0, 127.0), point2=(122.0, 121.0))
s1.VerticalConstraint(entity=g[4], addUndoState=False)
s1.Line(point1=(122.0, 72.0), point2=(122.0, 59.0))
s1.VerticalConstraint(entity=g[5], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[20], sketchUpEdge=e1[23], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1, depth=18.0,
    flipExtrudeDirection=OFF)
```

```
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=169.394,
    farPlane=592.116, width=61.3328, height=26.491, viewOffsetX=-42.5976,
    viewOffsetY=-5.00181)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#3 ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[20], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=167.078,
    farPlane=594.432, width=87.6892, height=37.8749, viewOffsetX=-39.31,
    viewOffsetY=-0.768389)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=-7.5)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#c ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[23], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=171.462,
    farPlane=590.049, width=33.4379, height=14.4426, viewOffsetX=-60.1071,
    viewOffsetY=-0.137365)
p = mdb.models['Model-1'].parts['Part-1']
e, d1 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d1[14], sketchUpEdge=e[136],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, -8.0,
    132.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=788.494,
    farPlane=886.838, width=296.873, height=128.226, cameraPosition=(
    -2.83723, 837.666, 221.185), cameraTarget=(-2.83723, 0, 221.185))
s.Line(point1=(136.0, 128.0), point2=(136.0, 58.0))
s.VerticalConstraint(entity=g[36], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=804.564,
    farPlane=870.768, width=134.447, height=58.0707, cameraPosition=(
    199.84, 837.666, 262.397), cameraTarget=(199.84, 0, 262.397))
s.Line(point1=(-136.0, 128.0), point2=(-136.0, 58.0))
s.VerticalConstraint(entity=g[37], addUndoState=False)
```

```
p = mdb.models['Model-1'].parts['Part-1']
e1, d2 = p.edges, p.datums
p.ShellExtrude(sketchPlane=d2[14], sketchUpEdge=e1[136], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s, depth=16.0,
    flipExtrudeDirection=OFF)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=149.634,
    farPlane=611.876, width=284.98, height=123.089, viewOffsetX=97.7619,
    viewOffsetY=-51.543)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=8.0)
session.viewports['Viewport: 1'].view.setValues(nearPlane=165.701,
    farPlane=595.81, width=91.9261, height=39.705, viewOffsetX=-30.9256,
    viewOffsetY=-4.55428)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#0 #c ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[26], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=155.298,
    farPlane=606.213, width=198.165, height=85.592, viewOffsetX=88.0232,
    viewOffsetY=-71.0969)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
session.viewports['Viewport: 1'].view.setValues(nearPlane=698.238,
    farPlane=1091.33, width=150.907, height=65.1802, viewOffsetX=-93.9579,
    viewOffsetY=24.3922)
session.viewports['Viewport: 1'].view.setValues(nearPlane=693.771,
    farPlane=1057.92, width=149.942, height=64.7632, cameraPosition=(
    633.544, -512.488, 621.518), cameraUpVector=(-0.57735, 0.471441,
    0.666641), cameraTarget=(116.941, 84.0109, 199.681),
    viewOffsetX=-93.3568, viewOffsetY=24.2361)
session.viewports['Viewport: 1'].view.setValues(nearPlane=784.932,
    farPlane=1133.93, width=169.644, height=73.2732, cameraPosition=(
    -331.192, -702.508, 621.518), cameraUpVector=(0.331187, 0.667761,
    0.666641), cameraTarget=(135.432, -66.1501, 199.681),
    viewOffsetX=-105.624, viewOffsetY=27.4207)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#3c0000 ]', ), )
p.Set(faces=faces, name='tube_plate')
session.viewports['Viewport: 1'].view.setValues(nearPlane=766.049,
    farPlane=1152.81, width=374.646, height=161.818, viewOffsetX=-62.5578,
    viewOffsetY=25.6414)
```

```
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#c53 ]', ), )
p.Set(faces=faces, name='diap')
session.viewports['Viewport: 1'].view.setValues(nearPlane=779.355,
    farPlane=1139.5, width=223.019, height=96.3267, viewOffsetX=-112.647,
    viewOffsetY=57.7427)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=707.101,
    farPlane=1082.46, width=61.1541, height=26.4138, viewOffsetX=-104.038,
    viewOffsetY=9.8767)
session.viewports['Viewport: 1'].view.setValues(nearPlane=668.32,
    farPlane=1089.99, width=57.8001, height=24.9652, cameraPosition=(
    797.57, 159.922, 656.825), cameraUpVector=(-0.247105, 0.778207,
    -0.57735), cameraTarget=(101.245, -61.1836, 140.222),
    viewOffsetX=-98.332, viewOffsetY=9.335)
session.viewports['Viewport: 1'].view.setValues(nearPlane=663.303,
    farPlane=1095, width=108.323, height=46.7872, viewOffsetX=-103.897,
    viewOffsetY=7.47468)
session.viewports['Viewport: 1'].view.setValues(nearPlane=799.525,
    farPlane=1163.48, width=130.569, height=56.3959, cameraPosition=(
    -395.405, 661.248, 656.825), cameraUpVector=(-0.642706, -0.503584,
    -0.57735), cameraTarget=(55.1935, 86.1679, 140.222),
    viewOffsetX=-125.234, viewOffsetY=9.00974)
session.viewports['Viewport: 1'].view.setValues(nearPlane=789.661,
    farPlane=1173.34, width=227.834, height=98.4068, viewOffsetX=-100.576,
    viewOffsetY=27.5379)
session.viewports['Viewport: 1'].view.setValues(nearPlane=828.13,
    farPlane=1065.13, width=238.933, height=103.201, cameraPosition=(
    -395.405, 798.089, 9.49652), cameraUpVector=(-0.642706, -0.763511,
    -0.0630962), cameraTarget=(55.1935, 25.6398, 39.7919),
    viewOffsetX=-105.475, viewOffsetY=28.8794)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=697.308,
    farPlane=1092.26, width=160.326, height=69.2484, viewOffsetX=-90.0544,
    viewOffsetY=11.2774)
session.viewports['Viewport: 1'].view.setValues(nearPlane=789.311,
    farPlane=1182.05, width=181.479, height=78.385, cameraPosition=(
    -390.059, 499.44, 835.305), cameraUpVector=(0.556789, 0.57735,
    -0.597204), cameraTarget=(108.146, -17.1628, 300.937),
    viewOffsetX=-101.936, viewOffsetY=12.7653)
session.viewports['Viewport: 1'].view.setValues(nearPlane=790.166,
    farPlane=1181.2, width=142.424, height=61.5159, viewOffsetX=-114.455,
    viewOffsetY=3.87646)
```

```
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#8003f3ac #1fc2 #7 ]', ), )
p.Set(faces=faces, name='frames')
session.viewports['Viewport: 1'].view.setValues(nearPlane=756.602,
    farPlane=1214.76, width=542.98, height=234.525, viewOffsetX=-45.3543,
    viewOffsetY=50.2486)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=190.0)
session.viewports['Viewport: 1'].view.setValues(nearPlane=784.807,
    farPlane=1186.56, width=196.722, height=84.9688, viewOffsetX=-113.387,
    viewOffsetY=22.8059)
p = mdb.models['Model-1'].parts['Part-1']
e, d2 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d2[31], sketchUpEdge=e[178],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, 0.0,
    190.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=683.99,
    farPlane=991.342, width=85.6714, height=37.0034, cameraPosition=(
    -21.0822, -10.276, 969.666), cameraTarget=(-21.0822, -10.276, 132))
s1.Line(point1=(132.0, 38.0), point2=(130.0, 38.0))
s1.HorizontalConstraint(entity=g[10], addUndoState=False)
s1.Line(point1=(130.0, 38.0), point2=(129.5, 39.5))
session.viewports['Viewport: 1'].view.setValues(nearPlane=688.003,
    farPlane=987.329, width=45.1086, height=19.4834, cameraPosition=(
    -20.6911, -38.4856, 969.666), cameraTarget=(-20.6911, -38.4856, 132))
s1.Line(point1=(129.5, 39.5), point2=(128.5, 39.5))
s1.HorizontalConstraint(entity=g[12], addUndoState=False)
s1.Line(point1=(128.5, 39.5), point2=(128.0, 38.0))
s1.Line(point1=(128.0, 38.0), point2=(122.0, 38.0))
s1.HorizontalConstraint(entity=g[14], addUndoState=False)
s1.undo()
s1.Line(point1=(128.0, 38.0), point2=(124.0, 38.0))
s1.HorizontalConstraint(entity=g[14], addUndoState=False)
s1.Line(point1=(132.0, 62.0), point2=(130.0, 62.0))
s1.HorizontalConstraint(entity=g[15], addUndoState=False)
s1.Line(point1=(130.0, 62.0), point2=(129.5, 60.5))
s1.Line(point1=(129.5, 60.5), point2=(128.5, 60.5))
```

```
s1.HorizontalConstraint(entity=g[17], addUndoState=False)
s1.Line(point1=(128.5, 60.5), point2=(128.0, 62.0))
s1.Line(point1=(128.0, 62.0), point2=(124.0, 62.0))
s1.HorizontalConstraint(entity=g[19], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=681.766,
    farPlane=993.566, width=108.147, height=46.7113, cameraPosition=(
    -38.2696, -47.5601, 969.666), cameraTarget=(-38.2696, -47.5601, 132))
s1.linearPattern(geomList=(g[11], g[12], g[13], g[14], g[16], g[17], g[18],
    g[19]), vertexList=(), number1=43, spacing1=6.0, angle1=180.0,
    number2=1, spacing2=6.0, angle2=90.0)
session.viewports['Viewport: 1'].view.setValues(nearPlane=681.195,
    farPlane=994.137, width=113.921, height=49.2049, cameraPosition=(
    200.644, -48.1478, 969.666), cameraTarget=(200.644, -48.1478, 132))
s1.Line(point1=(-128.0, 62.0), point2=(-128.5, 60.5))
s1.Line(point1=(-128.5, 60.5), point2=(-129.5, 60.5))
s1.HorizontalConstraint(entity=g[357], addUndoState=False)
s1.Line(point1=(-129.5, 60.5), point2=(-130.0, 62.0))
s1.Line(point1=(-130.0, 62.0), point2=(-132.0, 62.0))
s1.HorizontalConstraint(entity=g[359], addUndoState=False)
s1.Line(point1=(-128.0, 38.0), point2=(-128.5, 38.0))
s1.HorizontalConstraint(entity=g[360], addUndoState=False)
s1.ParallelConstraint(entity1=g[351], entity2=g[360], addUndoState=False)
s1.undo()
s1.Line(point1=(-128.0, 38.0), point2=(-128.5, 39.5))
s1.Line(point1=(-128.5, 39.5), point2=(-129.5, 39.5))
s1.HorizontalConstraint(entity=g[361], addUndoState=False)
s1.Line(point1=(-129.5, 39.5), point2=(-130.0, 38.0))
s1.Line(point1=(-130.0, 38.0), point2=(-132.0, 38.0))
s1.HorizontalConstraint(entity=g[363], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=663.952,
    farPlane=1011.38, width=326.159, height=140.875, cameraPosition=(
    118.912, -53.9214, 969.666), cameraTarget=(118.912, -53.9214, 132))
s1.move(vector=(0.0, -50.0), objectList=(g[10], g[11], g[12], g[13], g[14],
    g[15], g[16], g[17], g[18], g[19], g[20], g[21], g[22], g[23], g[24],
    g[25], g[26], g[27], g[28], g[29], g[30], g[31], g[32], g[33], g[34],
    g[35], g[36], g[37], g[38], g[39], g[40], g[41], g[42], g[43], g[44],
    g[45], g[46], g[47], g[48], g[49], g[50], g[51], g[52], g[53], g[54],
    g[55], g[56], g[57], g[58], g[59], g[60], g[61], g[62], g[63], g[64],
    g[65], g[66], g[67], g[68], g[69], g[70], g[71], g[72], g[73], g[74],
    g[75], g[76], g[77], g[78], g[79], g[80], g[81], g[82], g[83], g[84],
    g[85], g[86], g[87], g[88], g[89], g[90], g[91], g[92], g[93], g[94],
    g[95], g[96], g[97], g[98], g[99], g[100], g[101], g[102], g[103],
    g[104], g[105], g[106], g[107], g[108], g[109], g[110], g[111], g[112],
    g[113], g[114], g[115], g[116], g[117], g[118], g[119], g[120], g[121],
```

```
        g[122], g[123], g[124], g[125], g[126], g[127], g[128], g[129], g[130],
        g[131], g[132], g[133], g[134], g[135], g[136], g[137], g[138], g[139],
        g[140], g[141], g[142], g[143], g[144], g[145], g[146], g[147], g[148],
        g[149], g[150], g[151], g[152], g[153], g[154], g[155], g[156], g[157],
        g[158], g[159], g[160], g[161], g[162], g[163], g[164], g[165], g[166],
        g[167], g[168], g[169], g[170], g[171], g[172], g[173], g[174], g[175],
        g[176], g[177], g[178], g[179], g[180], g[181], g[182], g[183], g[184],
        g[185], g[186], g[187], g[188], g[189], g[190], g[191], g[192], g[193],
        g[194], g[195], g[196], g[197], g[198], g[199], g[200], g[201], g[202],
        g[203], g[204], g[205], g[206], g[207], g[208], g[209], g[210], g[211],
        g[212], g[213], g[214], g[215], g[216], g[217], g[218], g[219], g[220],
        g[221], g[222], g[223], g[224], g[225], g[226], g[227], g[228], g[229],
        g[230], g[231], g[232], g[233], g[234], g[235], g[236], g[237], g[238],
        g[239], g[240], g[241], g[242], g[243], g[244], g[245], g[246], g[247],
        g[248], g[249], g[250], g[251], g[252], g[253], g[254], g[255], g[256],
        g[257], g[258], g[259], g[260], g[261], g[262], g[263], g[264], g[265],
        g[266], g[267], g[268], g[269], g[270], g[271], g[272], g[273], g[274],
        g[275], g[276], g[277], g[278], g[279], g[280], g[281], g[282], g[283],
        g[284], g[285], g[286], g[287], g[288], g[289], g[290], g[291], g[292],
        g[293], g[294], g[295], g[296], g[297], g[298], g[299], g[300], g[301],
        g[302], g[303], g[304], g[305], g[306], g[307], g[308], g[309], g[310],
        g[311], g[312], g[313], g[314], g[315], g[316], g[317], g[318], g[319],
        g[320], g[321], g[322], g[323], g[324], g[325], g[326], g[327], g[328],
        g[329], g[330], g[331], g[332], g[333], g[334], g[335], g[336], g[337],
        g[338], g[339], g[340], g[341], g[342], g[343], g[344], g[345], g[346],
        g[347], g[348], g[349], g[350], g[351], g[352], g[353], g[354], g[355],
        g[356], g[357], g[358], g[359], g[360], g[361], g[362], g[363]) )
    session.viewports['Viewport: 1'].view.setValues(nearPlane=653.614,
        farPlane=1021.72, width=392.686, height=169.61, cameraPosition=(86.101,
        -45.8097, 969.666), cameraTarget=(86.101, -45.8097, 132))
    p = mdb.models['Model-1'].parts['Part-1']
    e1, d1 = p.edges, p.datums
    p.ShellExtrude(sketchPlane=d1[31], sketchUpEdge=e1[178], sketchPlaneSide=SIDE1,
        sketchOrientation=RIGHT, sketch=s1, depth=70.0,
        flipExtrudeDirection=OFF)
    s1.unsetPrimaryObject()
    del mdb.models['Model-1'].sketches['__profile__']
    session.viewports['Viewport: 1'].view.setValues(nearPlane=789.881,
        farPlane=1181.48, width=145.309, height=62.7621, viewOffsetX=-90.3262,
        viewOffsetY=14.0645)
    session.viewports['Viewport: 1'].view.setValues(nearPlane=817.045,
        farPlane=1146.96, width=150.306, height=64.9205, cameraPosition=(
        35.7326, 499.44, 983.701), cameraUpVector=(0.15837, 0.57735, -0.80099),
        cameraTarget=(177.439, -17.1628, 266.989), viewOffsetX=-93.4326,
```

```
    viewOffsetY=14.5481)
session.viewports['Viewport: 1'].view.setValues(session.views['Bottom'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=781.855,
    farPlane=893.476, width=364.986, height=157.646, viewOffsetX=26.7605,
    viewOffsetY=75.2948)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=-24.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=240.0)
session.viewports['Viewport: 1'].view.setValues(nearPlane=701.45,
    farPlane=1088.12, width=118.363, height=51.1235, viewOffsetX=-107.7,
    viewOffsetY=15.2006)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#0:12 #14 ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[33], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=685.226,
    farPlane=1104.34, width=282.926, height=122.202, viewOffsetX=47.1134,
    viewOffsetY=-55.4963)
session.viewports['Viewport: 1'].view.setValues(nearPlane=623.601,
    farPlane=1015.01, width=257.482, height=111.212, cameraPosition=(
    -145.463, 499.44, 730.956), cameraUpVector=(0.33105, 0.57735,
    -0.746373), cameraTarget=(150.755, -17.1628, 63.1139),
    viewOffsetX=42.8763, viewOffsetY=-50.5053)
session.viewports['Viewport: 1'].view.setValues(nearPlane=625.697,
    farPlane=1012.91, width=277.091, height=119.682, viewOffsetX=52.0012,
    viewOffsetY=-52.2661)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#0:12 #30000000 ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[34], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=627.015,
    farPlane=1011.6, width=262.082, height=113.199, viewOffsetX=54.4949,
    viewOffsetY=-54.49)
session.viewports['Viewport: 1'].view.setValues(session.views['Top'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=784.154,
    farPlane=891.177, width=384.659, height=166.143, viewOffsetX=-135.596,
    viewOffsetY=-83.991)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
session.viewports['Viewport: 1'].view.setValues(session.views['Bottom'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=785.702,
```

```
    farPlane=889.629, width=367.075, height=158.548, viewOffsetX=151.362,
    viewOffsetY=82.3651)
session.viewports['Viewport: 1'].view.setValues(session.views['Bottom'])
session.viewports['Viewport: 1'].view.setValues(session.views['Top'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=778.179,
    farPlane=897.152, width=452.64, height=195.506, viewOffsetX=41.995,
    viewOffsetY=-85.7592)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
faces = f.getSequenceFromMask(mask=('[#fffff00 #fffffff:10 #3 ]', ), )
p.Set(faces=faces, name='sheet')
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
session.viewports['Viewport: 1'].view.setValues(nearPlane=678.741,
    farPlane=1110.82, width=348.836, height=150.67, viewOffsetX=-29.8873,
    viewOffsetY=22.2042)
p = mdb.models['Model-1'].parts['Part-1']
e, d2 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d2[5], sketchUpEdge=e[1419],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, 0.0,
    186.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
session.viewports['Viewport: 1'].view.setValues(nearPlane=685.221,
    farPlane=990.11, width=73.2202, height=31.6254, cameraPosition=(
    22.9731, -4.01591, 969.666), cameraTarget=(22.9731, -4.01591, 132))
s.Line(point1=(90.0, 10.0), point2=(70.0, -10.0))
s.Line(point1=(70.0, -10.0), point2=(50.0, 10.0))
s.PerpendicularConstraint(entity1=g[182], entity2=g[183], addUndoState=False)
s.Line(point1=(50.0, 10.0), point2=(30.0, -10.0))
s.PerpendicularConstraint(entity1=g[183], entity2=g[184], addUndoState=False)
s.Line(point1=(30.0, -10.0), point2=(10.0, 10.0))
s.PerpendicularConstraint(entity1=g[184], entity2=g[185], addUndoState=False)
s.Line(point1=(10.0, 10.0), point2=(-10.0, -10.0))
s.PerpendicularConstraint(entity1=g[185], entity2=g[186], addUndoState=False)
s.Line(point1=(-10.0, -10.0), point2=(-30.0, 10.0))
s.PerpendicularConstraint(entity1=g[186], entity2=g[187], addUndoState=False)
session.viewports['Viewport: 1'].view.setValues(nearPlane=649.162,
    farPlane=1026.17, width=437.684, height=189.045, cameraPosition=(
    19.0648, -29.09, 969.666), cameraTarget=(19.0648, -29.09, 132))
s.Line(point1=(-30.0, 10.0), point2=(-50.0, -10.0))
```

```
s.PerpendicularConstraint(entity1=g[187], entity2=g[188], addUndoState=False)
s.Line(point1=(-50.0, -10.0), point2=(-70.0, 10.0))
s.PerpendicularConstraint(entity1=g[188], entity2=g[189], addUndoState=False)
s.Line(point1=(-70.0, 10.0), point2=(-90.0, -10.0))
s.PerpendicularConstraint(entity1=g[189], entity2=g[190], addUndoState=False)
s.Line(point1=(-90.0, -10.0), point2=(-110.0, 10.0))
s.PerpendicularConstraint(entity1=g[190], entity2=g[191], addUndoState=False)
s.Line(point1=(-110.0, 10.0), point2=(-130.0, -10.0))
s.PerpendicularConstraint(entity1=g[191], entity2=g[192], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d1 = p.edges, p.datums
p.Wire(sketchPlane=d1[5], sketchUpEdge=e1[1419], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=264.0)
p = mdb.models['Model-1'].parts['Part-1']
e, d2 = p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=d2[39], sketchUpEdge=e[1383],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(108.0, 0.0,
    264.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=837.66, gridSpacing=20.94, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
s1.Line(point1=(90.0, -10.0), point2=(70.0, 10.0))
s1.Line(point1=(70.0, 10.0), point2=(50.0, -10.0))
s1.PerpendicularConstraint(entity1=g[88], entity2=g[89], addUndoState=False)
s1.Line(point1=(50.0, -10.0), point2=(30.0, 10.0))
s1.PerpendicularConstraint(entity1=g[89], entity2=g[90], addUndoState=False)
s1.Line(point1=(30.0, 10.0), point2=(10.0, -10.0))
s1.PerpendicularConstraint(entity1=g[90], entity2=g[91], addUndoState=False)
s1.Line(point1=(10.0, -10.0), point2=(-10.0, 10.0))
s1.PerpendicularConstraint(entity1=g[91], entity2=g[92], addUndoState=False)
s1.Line(point1=(-10.0, 10.0), point2=(-30.0, -10.0))
s1.PerpendicularConstraint(entity1=g[92], entity2=g[93], addUndoState=False)
s1.Line(point1=(-30.0, -10.0), point2=(-50.0, 10.0))
s1.PerpendicularConstraint(entity1=g[93], entity2=g[94], addUndoState=False)
s1.Line(point1=(-50.0, 10.0), point2=(-70.0, -10.0))
s1.PerpendicularConstraint(entity1=g[94], entity2=g[95], addUndoState=False)
s1.Line(point1=(-70.0, -10.0), point2=(-90.0, 10.0))
```

```
s1.PerpendicularConstraint(entity1=g[95], entity2=g[96], addUndoState=False)
s1.Line(point1=(-90.0, 10.0), point2=(-110.0, -10.0))
s1.PerpendicularConstraint(entity1=g[96], entity2=g[97], addUndoState=False)
s1.Line(point1=(-110.0, -10.0), point2=(-130.0, 10.0))
s1.PerpendicularConstraint(entity1=g[97], entity2=g[98], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
e1, d1 = p.edges, p.datums
p.Wire(sketchPlane=d1[39], sketchUpEdge=e1[1383], sketchPlaneSide=SIDE1,
    sketchOrientation=RIGHT, sketch=s1)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].view.setValues(nearPlane=686.957,
    farPlane=1102.61, width=298.956, height=129.126, viewOffsetX=-9.93916,
    viewOffsetY=-71.5455)
session.viewports['Viewport: 1'].view.setValues(nearPlane=627.716,
    farPlane=972.236, width=273.175, height=117.99, cameraPosition=(
    218.827, 499.44, -485.54), cameraUpVector=(-0.103557, 0.57735,
    0.809903), cameraTarget=(126.166, -17.1628, 239.148),
    viewOffsetX=-9.08203, viewOffsetY=-65.3757)
session.viewports['Viewport: 1'].view.setValues(nearPlane=627.594,
    farPlane=972.358, width=282.353, height=121.954, viewOffsetX=9.62196,
    viewOffsetY=-23.0336)
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
edges = e.getSequenceFromMask(mask=('[#3fffff ]', ), )
p.Set(edges=edges, name='braces')
session.viewports['Viewport: 1'].view.setValues(width=266.088, height=114.93,
    viewOffsetX=129.279, viewOffsetY=-80.7922)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=-12.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=-4.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=4.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=12.0)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#0:12 #20000 ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[45], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
```

```
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[44], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[43], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#2 ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[42], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=-15.0)
p = mdb.models['Model-1'].parts['Part-1']
p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=15.0)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#17 #0:11 #400000 ]', ), )
d2 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d2[50], faces=pickedFaces)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#30a #0:11 #8000000 ]', ), )
d1 = p.datums
p.PartitionFaceByDatumPlane(datumPlane=d1[51], faces=pickedFaces)
session.viewports['Viewport: 1'].view.setValues(nearPlane=615.589,
    farPlane=984.363, width=454.34, height=196.24, viewOffsetX=117.906,
    viewOffsetY=-57.8661)
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
    engineeringFeatures=ON)
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=OFF)
mdb.models['Model-1'].Material(name='Material-1')
mdb.models['Model-1'].materials['Material-1'].Density(table=((0.0007487, ), ))
mdb.models['Model-1'].materials['Material-1'].Elastic(table=((27500000.0,
    0.26), ))
mdb.models['Model-1'].materials['Material-1'].Plastic(table=((36300.0, 0.0), (
    75000.0, 0.2287)))
mdb.models['Model-1'].HomogeneousShellSection(name='t_base_plate',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=2.0, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
```

```
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_frames',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.25, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_plateA',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.25, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_plateB',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.25, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_plateC',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.3125, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_post_plate',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.3125, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_sheet', preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=0.0625,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].HomogeneousShellSection(name='t_tube_plate',
    preIntegrate=OFF, material='Material-1', thicknessType=UNIFORM,
    thickness=0.3125, thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].LProfile(name='Profile-1', a=1.5, b=1.5, t1=0.25,
```

```
        t2=0.25)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['base_plate']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='t_base_plate', offset=0.0,
        offsetType=MIDDLE_SURFACE, offsetField='',
        thicknessAssignment=FROM_SECTION)
    mdb.models['Model-1'].BeamSection(name='braces', integration=DURING_ANALYSIS,
        poissonRatio=0.0, profile='Profile-1', material='Material-1',
        temperatureVar=LINEAR, consistentMassMatrix=False)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['braces']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='braces', offset=0.0,
        offsetType=MIDDLE_SURFACE, offsetField='',
        thicknessAssignment=FROM_SECTION)
    mdb.models['Model-1'].HomogeneousShellSection(name='t_diap', preIntegrate=OFF,
        material='Material-1', thicknessType=UNIFORM, thickness=0.25,
        thicknessField='', nodalThicknessField='',
        idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
        thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
        integrationRule=SIMPSON, numIntPts=5)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['diap']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='t_diap', offset=0.0,
        offsetType=MIDDLE_SURFACE, offsetField='',
        thicknessAssignment=FROM_SECTION)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['frames']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='t_frames', offset=0.0,
        offsetType=MIDDLE_SURFACE, offsetField='',
        thicknessAssignment=FROM_SECTION)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['plate_A']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='t_plateA', offset=0.0,
        offsetType=MIDDLE_SURFACE, offsetField='',
        thicknessAssignment=FROM_SECTION)
    p = mdb.models['Model-1'].parts['Part-1']
    region = p.sets['plate_B']
    p = mdb.models['Model-1'].parts['Part-1']
    p.SectionAssignment(region=region, sectionName='t_plateB', offset=0.0,
```

```
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Part-1']
region = p.sets['plate_C']
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='t_plateC', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Part-1']
region = p.sets['post_plate']
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='t_post_plate', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Part-1']
region = p.sets['sheet']
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='t_sheet', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Part-1']
region = p.sets['tube_plate']
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='t_tube_plate', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Part-1']
region=p.sets['braces']
p = mdb.models['Model-1'].parts['Part-1']
p.assignBeamSectionOrientation(region=region, method=N1_COSINES, n1=(0.0, 0.0,
    -1.0))
p = mdb.models['Model-1'].parts['Part-1']
region = p.sets['braces']
orientation=None
mdb.models['Model-1'].parts['Part-1'].MaterialOrientation(region=region,
    orientationType=GLOBAL, axis=AXIS_1,
    additionalRotationType=ROTATION_NONE, localCsys=None, fieldName='',
    stackDirection=STACK_3)
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
```

```
p = mdb.models['Model-1'].parts['Part-1']
a.Instance(name='Part-1-1', part=p, dependent=ON)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].FrequencyStep(name='Step-1', previous='Initial',
    numEigen=5)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON, bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
session.viewports['Viewport: 1'].view.setValues(nearPlane=702.665,
    farPlane=1086.9, width=106.058, height=45.8087, viewOffsetX=19.9433,
    viewOffsetY=78.2243)
session.viewports['Viewport: 1'].view.setValues(nearPlane=617.706,
    farPlane=1046.16, width=93.2342, height=40.2699, cameraPosition=(
    633.544, -243.254, 734.159), cameraUpVector=(-0.57735, 0.77939,
    0.243347), cameraTarget=(116.941, -25.5113, 36.7747),
    viewOffsetX=17.532, viewOffsetY=68.7662)
session.viewports['Viewport: 1'].view.setValues(nearPlane=624.071,
    farPlane=1039.8, width=35.4522, height=15.3126, viewOffsetX=-38.2345,
    viewOffsetY=30.2915)
a = mdb.models['Model-1'].rootAssembly
v1 = a.instances['Part-1-1'].vertices
verts1 = v1.getSequenceFromMask(mask=('[#0 #76016000 ]', ), )
region = regionToolset.Region(vertices=verts1)
mdb.models['Model-1'].DisplacementBC(name='FIXED', createStepName='Step-1',
    region=region, u1=0.0, u2=0.0, u3=0.0, ur1=UNSET, ur2=0.0, ur3=0.0,
    amplitude=UNSET, fixed=OFF, distributionType=UNIFORM, fieldName='',
    localCsys=None)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=OFF,
    bcs=OFF, predefinedFields=OFF, connectors=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
    meshTechnique=ON)
session.viewports['Viewport: 1'].view.setValues(width=33.3358, height=14.4328,
    viewOffsetX=-38.8664, viewOffsetY=30.4392)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
    engineeringFeatures=OFF, mesh=ON)
session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
    meshTechnique=ON)
session.viewports['Viewport: 1'].view.setValues(session.views['Iso '])
p = mdb.models['Model-1'].parts['Part-1']
p.seedPart(size=5.0, deviationFactor=0.1, minSizeFactor=0.1)
```

```
    p = mdb.models['Model-1'].parts['Part-1']
    p.generateMesh()
    a1 = mdb.models['Model-1'].rootAssembly
    a1.regenerate()
    a = mdb.models['Model-1'].rootAssembly
    session.viewports['Viewport: 1'].setValues(displayedObject=a)
    session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)
    session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
        meshTechnique=OFF)
    mdb.Job(name='Job-1', model='Model-1', description='', type=ANALYSIS,
        atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
        memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
         explicitPrecision =SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
        modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
        scratch ='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1,
        numGPUs=0)
    mdb.jobs['Job-1'].submit(consistencyChecking=OFF)
    session.mdbData.summary()
    o3 = session.openOdb(name='C:/Users/Euihyun/Documents/ABAQUS/Job-1.odb')
    session.viewports['Viewport: 1'].setValues(displayedObject=o3)
    session.viewports['Viewport: 1'].makeCurrent()
    session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=2 )
    session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=1 )
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        CONTOURS_ON_DEF, ))
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        CONTOURS_ON_UNDEF, CONTOURS_ON_DEF, ))
    session.viewports['Viewport: 1'].odbDisplay.setFrame(step=0, frame=2 )
```

C.3 abaqus_run.py

```
def abaqus_run(params):
    import section
    import regionToolset
    import displayGroupMdbToolset as dgm
    import part
    import material
    import assembly
    import step
    import interaction
    import load
    import mesh
    import optimization
```

```
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
t_A = params[0]
t_B = params[1]
t_C = params[2]
t_base = params[3]
t_post = params[4]
t_junc = params[5]
t_tube = params[6]
t_ud = params[7]
t_ld = params[8]
d = params[9]
Es = params[10]
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
mdb.models['Model-1'].sections['t_base_plate '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_base,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_diap_up '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_ud,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_diap_lo '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_ld,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_frames '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=0.25,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_plateA '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_A,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_plateB '].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_B,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_plateC '].setValues(preIntegrate=OFF,
```

```
    material='Material-1', thicknessType=UNIFORM, thickness=t_C,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_post_plate'].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_post,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_sheet'].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=0.0625,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_tube_plate'].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_tube,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].sections['t_junc_plate'].setValues(preIntegrate=OFF,
    material='Material-1', thicknessType=UNIFORM, thickness=t_junc,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, integrationRule=SIMPSON, numIntPts=5)
mdb.models['Model-1'].materials['Material-1'].elastic.setValues(table=((
    Es.0, 0.26), ))
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
mdb.jobs['Job-1'].submit(consistencyChecking=OFF)
session.mdbData.summary()
session.viewports['Viewport: 1'].setValues(
    displayedObject=session.odbs['C:/Users/Euihyun/Documents/ABAQUS/Job-1.odb'])
o3 = session.openOdb(name='C:/Users/Euihyun/Documents/ABAQUS/Job-1.odb')
session.viewports['Viewport: 1'].setValues(displayedObject=o3)
session.viewports['Viewport: 1'].makeCurrent()
odb = session.odbs['C:/Users/Euihyun/Documents/ABAQUS/Job-1.odb']
xy_result = session.XYDataFromHistory(name='EIGFREQ Whole Model-1', odb=odb,
    outputVariableName='Eigenfrequency: EIGFREQ for Whole Model', steps=(
    'Step-1', ), __linkedVpName__='Viewport: 1')
c1 = session.Curve(xyData=xy_result)
xyp = session.XYPlot('XYPlot-1')
chartName = xyp.charts.keys()[0]
chart = xyp.charts[chartName]
chart.setValues(curvesToPlot=(c1, ), )
session.charts[chartName].autoColor(lines=True, symbols=True)
session.viewports['Viewport: 1'].setValues(displayedObject=xyp)
x0 = session.xyDataObjects['EIGFREQ Whole Model-1']
session.writeXYReport(fileName='abaqus.rpt', xyData=(x0, ))
```