The Effect of Foundations' Uplift on the Seismic Behavior RC Buildings with a Dual System Considering Soil-Structure-Interaction

Amir Amirbeygloo

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Civil Engineering

Eastern Mediterranean University
September 2024
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Civil Engineering.

Assoc. Prof. Dr. Eriş Uygar
Chair, Department of Civil Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Civil Engineering.

Prof. Dr. Mahmood Hosseini
Supervisor

Examining Committee

1. Prof. Dr. Mahmood Hosseini
2. Assoc. Prof. Dr. Mehmet Cemal Geneş
3. Assoc. Prof. Dr. Rifat Reşatoğlu

ABSTRACT

In conventional seismic design, it is often assumed that the building's foundation is firmly anchored in the ground. However, this assumption, particularly concerning vertical movement, is not always accurate, potentially leading to design errors, especially in soft soil conditions. This thesis investigates the impact of foundation movement on the seismic response of regular multi-story concrete RC buildings with shear walls and moment frames dual system, considering variables such as type of soil (C and D), foundation geometry, number of stories, foundation burial depth, and soil bearing capacity.

The study begins with the analysis of three buildings with 7, 10, and 15 floors using the non-linear time history analysis (NLTHA) method via Openseespy software. Results from the NLTHA, using selected earthquake accelerograms, indicate that foundation movement tendencies are influenced by foundation geometry and the number of stories, significantly affecting the seismic behavior of the buildings. Shallow foundations and soil can absorb earthquake energy through different displacement mechanisms such as, rocking, slipping, settling. This helps to lessen the forces that reach the superstructure. Without considering Soil-Structure Interaction (SSI), all earthquake energy is assumed to be dissipated solely by structural elements like beams, columns, and shear walls. This assumption often results in artificially high calculations for moments, stresses, and strains, especially when the structure is founded on clay soil. In contrast, for structures on sandy soil, the impact of ignoring SSI is typically less significant and may lead to only minor variations in these figures, which can often be disregarded.

Keywords: Soil-Structure Interaction, Dual System, Foundation Uplift, Multi-story RC Building, OpenSees, Nonlinear Time History Analysis, Shallow Foundations

ÖZ

Geleneksel sismik tasarımda, binanın temelinin zemine sıkıca sabitlendiği varsayılır. Ancak, özellikle dikey hareketle ilgili olarak bu varsayım her zaman doğru değildir ve bu da özellikle yumuşak zemin koşullarında tasarım hatalarına yol açabilir. Bu tez, temel hareketinin, zemin türü (C ve D), temel geometrisi, kat sayısı, temel gömülme derinliği ve zemin taşıma kapasitesi gibi değişkenleri dikkate alarak, perde duvarlar ve moment çerçeveleri çift sistemine sahip düzenli çok katlı betonarme binaların sismik tepkisi üzerindeki etkisini araştırmaktadır.

Çalışma, Openseespy yazılımı aracılığıyla doğrusal olmayan zaman tanım alanı analizi (NLTHA) yöntemi kullanılarak 7, 10 ve 15 katlı üç binanın analizi ile başlamaktadır. Seçilen deprem ivme kayıtları kullanılarak yapılan NLTHA sonuçları, temel hareketi eğilimlerinin temel geometrisi ve kat sayısından etkilendiğini ve bu durumun binaların sismik davranışını önemli ölçüde etkilediğini göstermektedir. Sığ temeller ve zemin, sallanma, kayma, oturma gibi farklı yer değiştirme mekanizmaları aracılığıyla deprem enerjisini emebilir. Bu, üst yapıya ulaşan kuvvetlerin azalmasına yardımcı olur. Zemin-Yapı Etkileşimi (SSI) dikkate alınmadan, tüm deprem enerjisinin yalnızca kirişler, kolonlar ve perde duvarlar gibi yapısal elemanlar tarafından dağıtıldığı varsayılır. Bu varsayım, özellikle yapı kil zemin üzerine oturduğunda, momentler, gerilmeler ve deformasyonlar için yapay olarak yüksek hesaplamalarla sonuçlanır. Buna karşılık, kumlu zemin üzerindeki yapılar için SSI'nin göz ardı edilmesinin etkisi genellikle daha az önemli olup, bu rakamlarda yalnızca küçük değişikliklere yol açabilir ve genellikle göz ardı edilebilir.

Anahtar Kelimeler: Zemin-Yapı Etkileşimi, Çift Sistem, Temel Kalkması, Çok Katlı Betonarme Bina, OpenSees, Doğrusal Olmayan Zaman Tanım Alanı Analizi, Sığ Temeller

DEDICATION

This thesis is dedicated to:

My wife, Samaneh Dehghan, and my parents, Mohammad Ali and Azra Ansari, along with my brothers, Mohammad Reza and Alireza. Your unwavering love, support, and faith in me have been my greatest sources of strength and inspiration. Your sacrifices and belief in my abilities have made this achievement possible. Thank you for always being there for me.

ACKNOWLEDGMENT

I would like to express my profound gratitude to Prof. Dr. Mahmood Hosseini for his exceptional supervision, advice, and guidance from the very early stages of this thesis. His invaluable insights and extraordinary experiences have greatly contributed to the success of this work. Most importantly, his unwavering encouragement and support in numerous ways have been indispensable. His innovative ideas, extensive experience, and passionate dedication have truly inspired and enriched my growth as a student. I am indebted to him more than words can convey.

I am also grateful to the members of my thesis committee, Assoc. Prof. Dr. Mehmet Cemal GeneŞ and Assoc. Prof. Dr. Rifat ReŞatoğlu, for their valuable comments and suggestions that significantly contributed to the quality of my work.

My thanks go to my friend, Shamsuddeen Yusef Umer who helped and encouraged me during the period of my studies and this thesis.

I specifically appreciate the administrative and technical staff at institute of graduate studies EMU, for their help and support in many administrative and technical aspects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBRIVATIONS

β	Load inclination angle
δ	Angle of friction between the foundation and soil
γ	Soil unit weight
ϕ′	Angle of internal friction
ρ	Ratio of total area of longitudinal reinforcement
ρsh	Area ratio of transverse reinforcement in the region of close spacing at the column end
θcap,pl	Plastic chord rotation from yield to cap point under monotonic loading, rad
θpc	Post-capping plastic rotation capacity under monotonic loading, rad
θu	Plastic rotation from yield to "ultimate" (at 20% stress loss), rad
θy	Yield rotation capacity, rad
Ω0	Overstrength Factor
Ac	Cross-sectional area of the column
Af	Area of the base of footing in contact with the soil
Ag	Gross cross-sectional area of the column
a_{sl}	Indicator variable (0 or 1) to signify the possibility of longitudinal reinforcing bar slip past the column end
B	Width of foundation
Cd	Deflection Amplification Factor
Cr	Parameter controlling the range of the elastic portion
Df	Embedment depth of foundation

d	Depth of column cross section, measured from extreme compressive fiber to centerline of tensile reinforcement
d′	Effective depth of the column
E	Elastic modulus
Es	Modulus of elasticity of steel
fc	Compressive strength of concrete
fy	Yield strength of reinforcing steel
Fcd,Fqd,Fγd	Depth factors
Fci,Fqi,Fγi	Inclination factors
Fcs,Fqs,Fγs	Shape factors
G	Shear modulus of the soil
H	Height of story
Kend	Stiffness at the end of foundation
Kin	Initial elastic stiffness of foundation
Kmid	Stiffness at the midway of foundation
Kp	Passive earth pressure coefficient
Ksh	Strain-hardening stiffness in model
Kss	Post-capping strain-softening stiffness in model
Kx	Lateral stiffness of foundation
Kxx	Rotational stiffness of foundation
Ky	Lateral stiffness of foundation
Kyy	Rotational stiffness of foundation
Kz	Vertical stiffness of foundation
K	Stiffness of soil
L	Length of foundation

Mc	Maximum moment capacity in the model
Mw	Moment magnitude
My	Yield moment capacity in the model
N	Axial load
Nc,Nq,Nγ	Bearing capacity factors
n	Modular ratio
P	Axial load in the column
Pult	Passive earth pressure per unit length of footing
P	Contact pressure
q	Instantaneous load
q0	Load at the yield point
qult	Ultimate bearing capacity per unit area of footing
Qult	Ultimate bearing capacity
R	Response Modification Coefficient
Re	End length ratio of foundation
Rk	Stiffness intensity ratio of foundation
Tult	Frictional resistance of foundation
tult	Frictional resistance per unit area of foundation
Vs30	Average shear wave velocity in the top 30 meters of soil
Wg	Vertical force acting at the base of the foundation
w	Soil deformation

Chapter 1

INTRODUCTION

1.1 Background

In structural design, certain simplifications are typically made, including the assumption that the structure is rigidly connected to the ground, disregarding foundation movements and soil deformation. Soil-structure interaction (SSI) is a crucial aspect of earthquake engineering that has garnered substantial global interest in recent years. SSI addresses the wave propagation in systems where structures are built on the soil. The concept originated in the late 19th century and has significantly advanced over the decades, particularly during the early 20th century. The latter half of the 20th century saw major progress, spurred by the demands of the nuclear power and offshore industries, the advent of high-performance computing, and the use of simulation tools such as finite element methods. Additionally, there has been a growing emphasis on enhancing seismic safety. The primary goal of seismic design remains the prevention of structural collapse.[1] During the dynamic analysis of how structures react to earthquake ground motion, it is generally assumed that the ground motion felt in an open region is transmitted directly to the foundation of the structure without any modifications. The underlying assumptions of this technique suggest that the soil beneath the building remains unchanged. This technique is based on the assumptions that the ground beneath the structure is entirely inflexible and that the structural foundation is firmly connected to the soil. Soils have a limited level of rigidity, and their structural foundations depend entirely on the force of gravity for support.[1]

It is widely recognized that ground deformation beneath foundations can effectively dissipate energy. However, this advantage often comes with the potential downsides of significant temporary and permanent distortions. To accurately assess the performance of these systems, it is crucial to develop robust calculations and consider the effects of permanent settlement and rotation.

1.2 Statement of Problem

Reinforced concrete (RC) structures with dual systems, comprising walls and frames, are highly effective at resisting lateral forces from wind and earthquakes thanks to their remarkable strength and stiffness. It is important to have analytical models that can accurately capture the observed nonlinear hysteretic behavior of RC walls, considering their critical significance in the seismic performance of buildings. [1] Numerical technologies, such as finite element analysis, are necessary to verify the accuracy of existing seismic design methods for walls and frames, as well as to study the elastic and plastic behavior of soil. Specifically, it is necessary to model the degradation in the ability to bend and the resulting decrease in sideways strength. This thesis utilizes the open system for earthquake engineering simulation (OpenSees) within a Python environment. An illustrative instance, wherein the spread of the consequences of a swaying foundation-superstructure becomes obvious, is seen in Figure 1[1]. The shear wall is generally more rigid than the frame, which means it will bear greater weight if the ground is firm. On the other hand, the frame, being more flexible, may suffer damage from significant movements caused by a less rigid ground.[2] or functioned as a dampening mechanism for the entire construction, dispersing the energy. Within this investigation, this particular subject is examined.

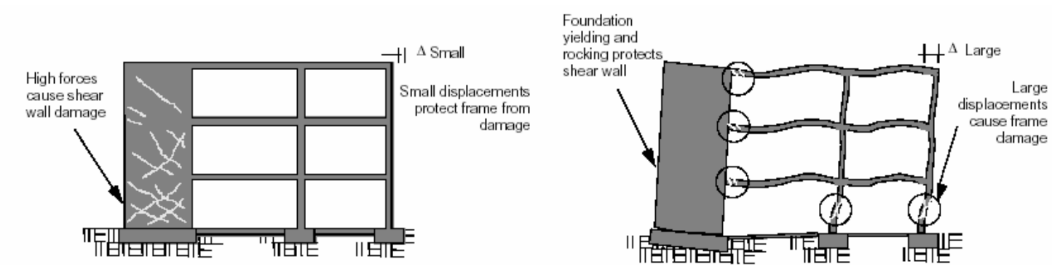


Figure 1: Comparison between response of a structural system, considering firm and uplifting foundation[1]

1.3 Research Question

This research focuses on examining the effects of soil-structure interaction on a dual system that includes shear walls and a 3D concrete frame. The elevation and movement of shallow foundations, along with soil characteristics, can introduce additional nonlinearity into the overall force-deformation behavior of the system. While such nonlinearity may help reduce seismic demands, it can also lead to excessive stress on lateral and/or gravity-resisting components due to displacement compatibility. This study explores the influence of this interaction on both the shear wall and the frame. The thesis aims to compare the impact of soil-structure interaction on the frame and shear walls to that of structures with assumed fixed conditions.

1.4 Scope and Novelty and Limitation

This research study focuses on the impact of a dual resistance system on clay and sandy soil in buildings of different heights, specifically 7, 10, and 15 stories. Each soil type has eleven different ground motion effects on these buildings. The models were analyzed using three-dimensional (3D) analysis. Regarding the novelty of this thesis, previous research primarily focused on 2D analysis under SSI conditions, with foundations and soil modeled in just two dimensions. These studies did not account for the hysteretic behavior of structural elements such as beams and columns, nor did they fully capture the material behavior of components like concrete and steel reinforcement in shear walls. With respect to the limitation, there are several constraints associated with this study. The selection of material attributes and behavior was based on previous studies, which may introduce some inaccuracies. Another limitation is made by limitation in OpenSeespy, which just supports two types of soil, including sand and clay soil.

1.5 Structure of Thesis

The literature review presented in Chapter 2 provides a comprehensive review of the relevant scholarly works. This chapter explains the studies conducted on the modeling of numerical soils, beams, columns, and shear walls. Chapter 3 presents the methodology employed specifically for this investigation. The process of designing structures, defining plastic hinges, modeling soils and shear walls, and selecting and scaling ground motion. In Chapter four, the results for two conditions are provided: one where the structure adopts a fixed position in the soil, and another where the structure is studied in interaction with the soil. This chapter examines the degradation, displacement, and acceleration of structures, as well as the behavior of soils. Chapter five contains the conclusions, recommendations, and summaries. In Appendix A, the scripts that simulate the soil and structure in a 3D model using OpenSeespy, including shear walls, are added. In Appendix B, scripts for calculating M_y are included. Appendix C contains the Python scripts that are provided to calculate the properties of hinges for reinforced concrete frames.

Chapter 2

LITERATURE REVIEW

2.1 Soil-Structure Interaction

When a structure is built on soil, it experiences two types of interaction effects: kinematic effects and inertial effects. Kinematic interaction effects occur when there is a change in the density and elasticity of the wave propagation medium, leading to changes in the way the waves propagate. It alters the speed at which waves travel and causes incoming seismic waves to bounce back or change direction. The kinematic effects of SSI refer to the alteration in the behavior of a structure when its reaction is analyzed using free-field motions compared to when the influence of the structure itself is taken into account. The phenomenon is independent of the mass of the structure and is influenced by the structure's geometry and configuration, the depth at which the foundation is embedded, the composition of the incident waves in the surrounding area, and the angle at which these waves strike the structure. For structures without any attachment, the effect of kinematic interaction can be disregarded when they are subjected to vertically propagating shear waves. Inertial effects arise from the collective dynamic response of the structure, foundation, and supporting soil medium. The elastic and inertial properties of soil media enhance the structural degrees of freedom, enabling the dissipation of seismic wave energy through the propagation of waves away from the structure and the hysteresis deformation of the supporting soil media. The influence of inertia is determined by the relative flexibility of the soil supporting the building. This means that the effect is not substantial for structures built on rigid soils or rock, but it can be considerable for structures that are stiff and huge.[2]

Soil-structure interaction, a significant topic in earthquake engineering, has received significant global interest in recent decades. Soil-structure interaction phenomena refer to the transmission of waves in a system where buildings are constructed on the surface of the soil.[3]

Typically, structures are considered to be immobile at their foundations during the analysis and design phase when subjected to dynamic forces. However, taking into account the actual support flexibility decreases the overall rigidity of the structure and lengthens the time of the system. Significant variations in spectral acceleration are found across different natural periods on the response spectrum curve. Therefore, the alteration in the natural period can significantly impact the seismic response of any building. Furthermore, the soil medium exhibits damping properties as a result of its inherent qualities. Some studies also examine the problems of expanding the natural period and the inclusion of excessive damping in soil caused by soil-structure interaction in building constructions. Furthermore, the correlation between the vibration periods of a structure and the supporting soil is crucial in determining the seismic behavior of the structure. The seismic events of the 1970 earthquake in Gediz, Turkey, and the 1967 earthquake in Caracas resulted in the demolition of a section of a factory and the destruction of structures, respectively, highlighting the significance of this matter. These findings demonstrate the necessity of considering soil-structure interaction when analyzing the dynamic behavior of structures in real-world scenarios. Therefore, it is crucial to include the interaction between soil and structure when analyzing the response of a structure to dynamic stresses.[2]

Research on soil-structure interaction has shown that the dynamic response of a structure situated on soft soil can differ significantly from that of an identical structure placed on firm soil.[4] A key factor contributing to this difference is the dissipation of vibrational energy in the flexibly mounted structure, which occurs through the emission of stress waves in the supporting medium and the hysteresis effect within the medium itself. There are well-established analytical methods for determining the effects of dynamic soil-structure interaction.[5] The study of structure-soil-structure interaction (SSSI) encompasses various methodologies, including analytical approaches, analytical-numerical techniques, numerical methods, experimental tests, and prototype inspections.[3] The numerical technique has seen substantial progress due to the rapid development of computers. This computational approach is widely recognized as one of the most effective methods for investigating SSSI. As a result, many seismologists have employed it, leading to a significant number of articles published on the topic from 1980 to the present.[3] Forexample, Cemal Genes (2005, 2012) emphasized on an arbitrary-shaped body subjected to dynamic loads requires the use of discrete numerical methods such as the finite element method (FEM) or the boundary element method (BEM). These two methods can be formulated in time or frequency spaces, and each has relative benefits and limitations. The decision of which methods should be used strongly depends on the properties of the system to be analysed, such as material properties, geometry, type of loading and boundary conditions. The FEM is well-suited for non-homogeneous and anisotropic materials of arbitrary-shaped structure with non-linear behaviour. For systems with infinite or semi-infinite extension, FE methods require a large-scale mesh to represent the surrounding soil medium (near-field), which is bounded by the far-field that is represented by artificial boundaries.[6, 7]

Due to its widespread use in building, there has been extensive research conducted in the field of shallow foundations. Shallow foundations that experience inelastic behavior offer a highly effective means of dissipating energy. The waste of energy leads to a decrease in the required resisting force by the structure. A big part of seismic input energy can be dissipated by various modes of movement the building's foundations including rocking, settling as well as sliding. However, the mode of rocking is particularly intriguing because it tends to naturally re-center and lead to a more stable reaction.[8] This movements demonstrates in the Figure 2.[8] As it is clear I assume three movements including rocking movement, sliding and vertical mode.

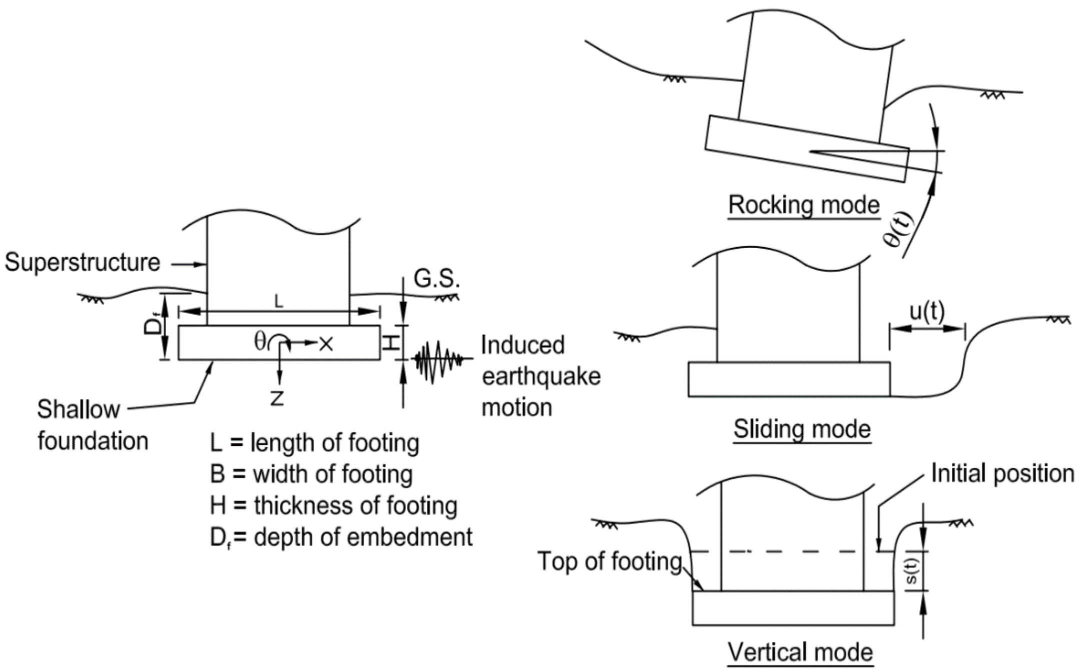


Figure 2: Different modes of foundation deformation[8]

Housner (1963) was the first to explore this concept by incorporating a rigid block model that could rock (Figure 3). He later proposed closed-form formulation for calculating of the kinetic energy loss.[8]

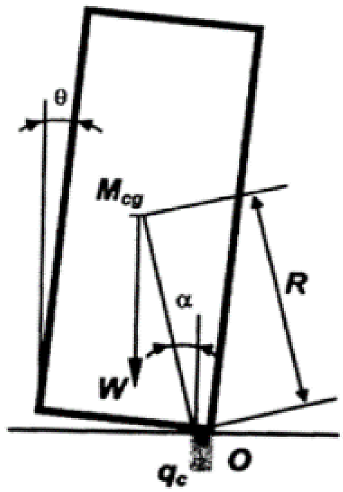


Figure 3: Rigid rocking block model[9]

Recently, scientists have been studying how shallow foundations behave during earthquakes. They've conducted experiments and used math to understand how these foundations might bend or flex under seismic stress. Typical methods used to predict the reaction of shallow foundations, taking into account the interaction between soil, foundation, and structure. Winkler-based methods can employ 1D spring elements to model the overall response of the soil structure interface. The Winkler spring technique is attractive in design because of its straightforwardness and little computing requirements. Moreover, the deterministic characteristics of spring-based models might facilitate the calibration process, provided that response data is available.[8]

The methodology for addressing the soil-foundation interaction (SFI) problem can be roughly classified into two main approaches: the Winklerian Approach and the Elastic Continuum Approach. One approach is to begin with the Winkler Model and incorporate an interaction element, or alternatively, start with the Continuum model and make a few simplifying assumptions regarding stresses and/or displacements. The Winkler Model The initial and most basic mechanical model, proposed by Winkler, asserts that the contact pressure, p, at each site varies in a linear manner with the soil deformation at that specific place. The mathematical representation is given by the equation

$$p = kw \tag{2.1}$$

where w represents the soil deformation and k is the modulus of subgrade reaction.[10] An analytical investigation was conducted by Chopra and Yim (1985) for assessment of the rocking behavior of a single-degree-of-freedom (SDOF) system, taking into account the uplift of the base. The study focused on analyzing individual spring elements that were assumed to have linear elastic properties (Figure 4). The authors have devised simplified formulas for determining the base shear resistance of flexible structures that are permitted to rise in the case of single-degree-of-freedom (SDOF) systems.[11] In their analysis, the researchers make three assumptions. Figure 4-a represents a rigid foundation, figure b illustrates a foundation with a two-element (spring-dashpot) system, and figure c depicts a dispersed winkler (spring-dashpot) system. In the application detailed in this report, Nakaki and Hart (1987) utilized vertically arranged elastic springs with viscous dampers strategically placed at the base of a shear wall structure. Their study highlighted the benefits of allowing the foundation supporting the shear wall system to uplift when subjected to earthquake-induced forces.[12]

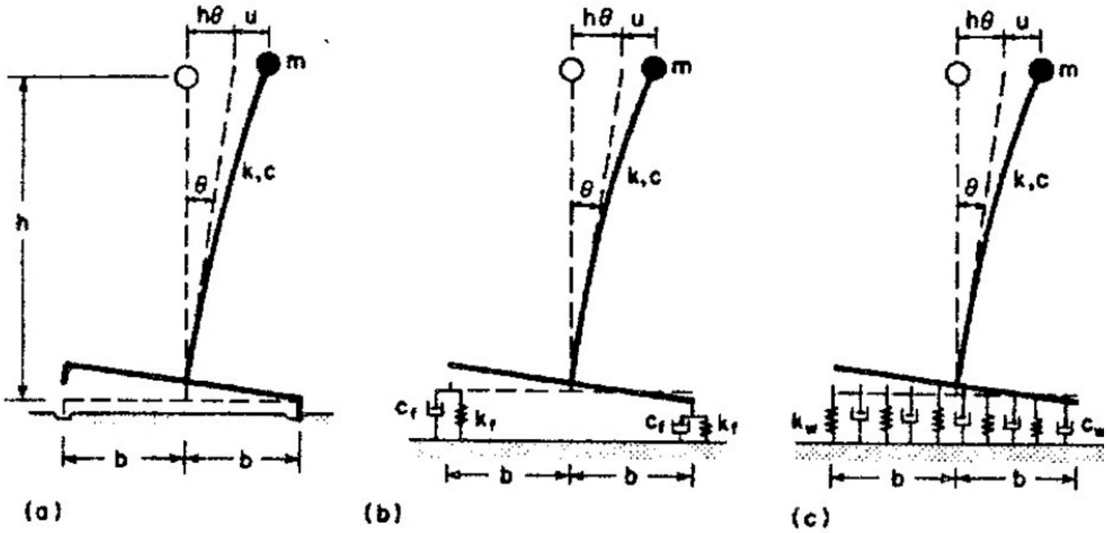


Figure 4: The foundation system idealized by Chopra and Yim (1985) includes three models: (a) a rigid foundation, (b) a two-element system comprising a spring and dashpot, and (c) a distributed Winkler system, which also incorporates spring and dashpot elements [11]

The Beam on Nonlinear Winkler Foundation (BNWF) model offers a spring-based alternative for analyzing Soil-Structure Interaction (SSI), incorporating the non-linear characteristics of the soil. This approach models the soil by using an array of one-dimensional non-linear springs, which are uniformly placed along the interface between the soil and the foundation.[8] During strong seismic events, the soil beneath the foundation is susceptible to plastic deformation due to the increased pressure caused by the foundation's rocking and sliding actions. This results in a gradual reduction in the system's stiffness as the contact between the foundation and the ground diminishes. The BNWF model incorporates a series of vertical and horizontal non-linear springs to assess plastic deformations and energy dissipation within the soil. As illustrated in Figure 5, the vertical springs along the footing's length (q-z elements) effectively capture the rocking, uplift, and settlement. In contrast, the horizontal springs (t-x and p-x elements) are strategically positioned to capture the soil's sliding and passive resistance.

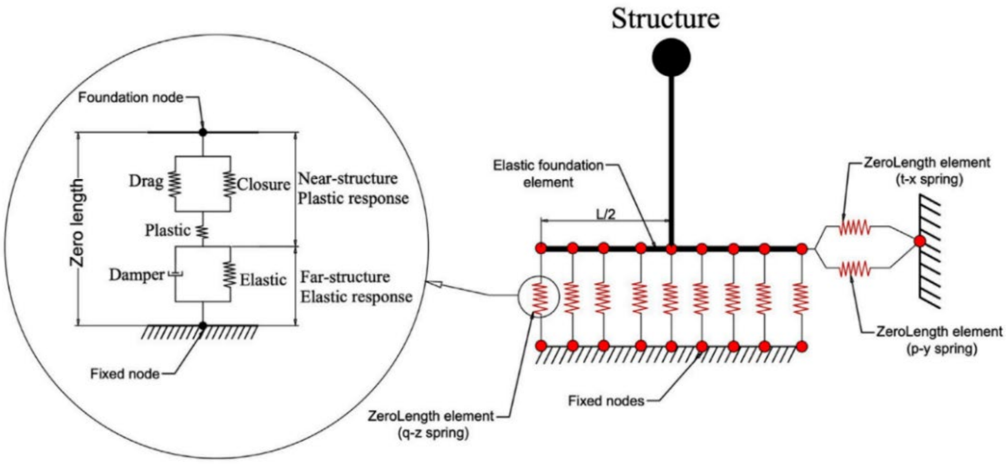


Figure 5: Schematic illustration of a beam-on-nonlinear Winkler foundation (BNWF) model[8]

The integration of the BNWF model into OpenSees was demonstrated by Raychowdhury and Hutchinson (2009) for a two-dimensional structural model. This model is applicable to all types of surface and embedded foundations and can be used for clay and sand soil deposits, irrespective of the structural model involved. As a result, the BNWF model is well-suited for a wide range of SSI problems in structural engineering. However, its primary limitation lies in its one-dimensional nature, where the spring's response is confined to the direction of the spring itself, and any load applied perpendicular to the spring axis does not influence its behavior. Despite this limitation, the BNWF approach has gained significant popularity and has been extensively adopted by researchers over the past two decades, thanks to its simplicity and versatility in addressing various challenges.[8] For instance, Shirzadi conducted a study examining the effect of torsion on the inelastic responses of steel structures supported by a nonlinear flexible medium.[11]

2.2 Performance-Based Assessment

The goal of Performance-Based Earthquake Engineering (PBEE) is to evaluate the seismic risk of structures by focusing on specific performance metrics or objectives. A performance objective for a building is defined as the acceptable probability of experiencing or exceeding a particular level of damage (ranging from elastic behavior to complete collapse) under a certain hazard level. The first iteration of PBEE assessment in the United States is reviewed in ATC (1996). [1]and FEMA (1997). [13] The guidelines encompass various performance levels, including Immediate Occupancy (IO), Life Safety (LS), and Collapse Prevention (CP). The issue was framed by applying these methodologies, as illustrated in Figure 6. In this context, the building is shown to experience lateral stresses caused by an earthquake, resulting in a nonlinear response and subsequent damage. Following this, relationships are established between structural response indicators (such as inter-story drifts, inelastic deformations of members, and member forces) and performance-based criteria like Immediate Occupancy, Life Safety, and Collapse Prevention.[14]

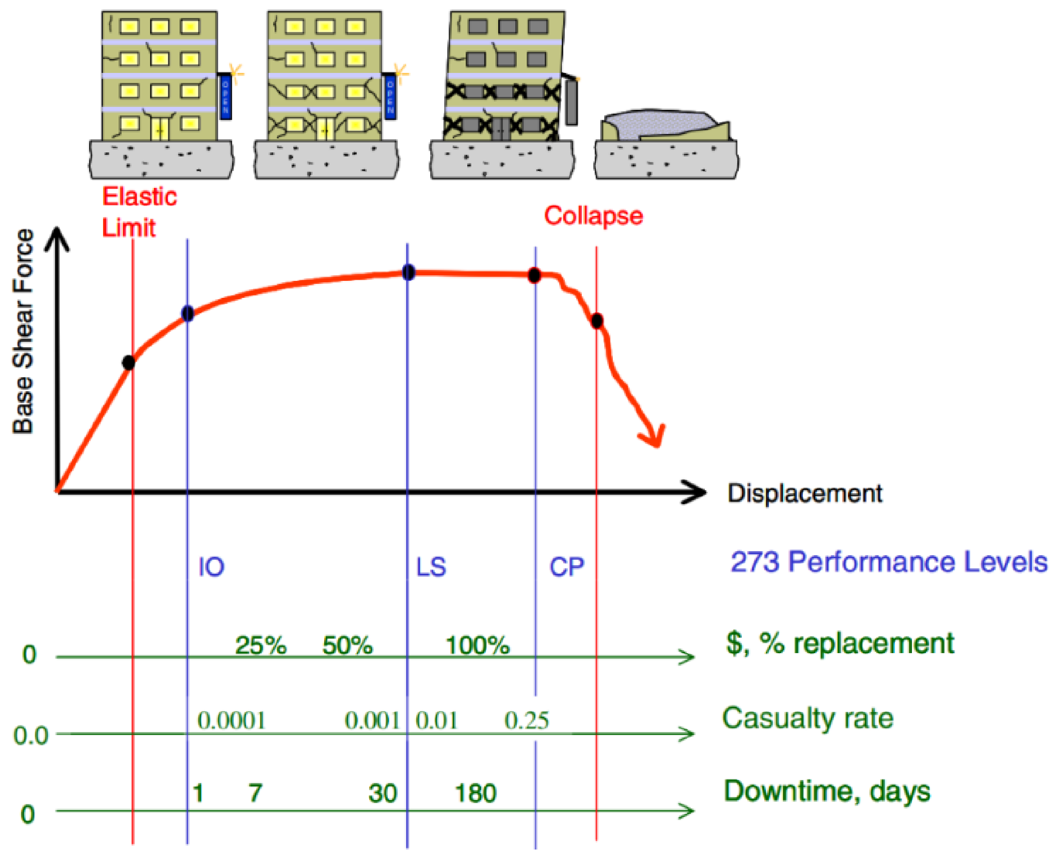


Figure 6: A visualization of performance-based earthquake engineering[14]

This approach begins by defining a ground motion intensity measure (IM), which describes the characteristics of the ground motion hazard that influence a structure's seismic behavior. The models generate engineering demand parameters (EDPs) for a building, including story drift ratios, shear forces, overturning moments, floor absolute accelerations, and inelastic deformation demands of components. These EDPs are then linked to damage metrics that quantify the degree of damage to the building and its structural and nonstructural elements. Finally, these damage metrics are used to assess the impacts on human casualties and financial losses, both of which can be considered as decision variables.[15]

There are differing perspectives on the definition of performance-based design. It is an approach that defines structural design criteria based on achieving specific performance objectives. Recent studies have explored the current state and emerging trends in performance-based design and its evaluations. Performance-based design is often used interchangeably with displacement-based design. This concept is based on the idea that performance targets can be linked to the level of damage sustained by the structure, which can, in turn, be related to displacements and drift. However, this assumption is overly simplistic, as the extent of damage is influenced by several additional factors, including the accumulation and distribution of structural damage, the failure modes of elements and components, the number of cycles and duration of the earthquake, and the acceleration levels.[16]

In performance-based design, a key question is to assess the appropriateness of the selected performance levels, the specific parameters used to define their minimum standards, and the definitions of seismic hazard. Figure 7 shows that for the three performance levels—serviceability, damage control, and life safety or collapse prevention—the primary structural characteristics are stiffness, strength, and deformation capacity.[16]

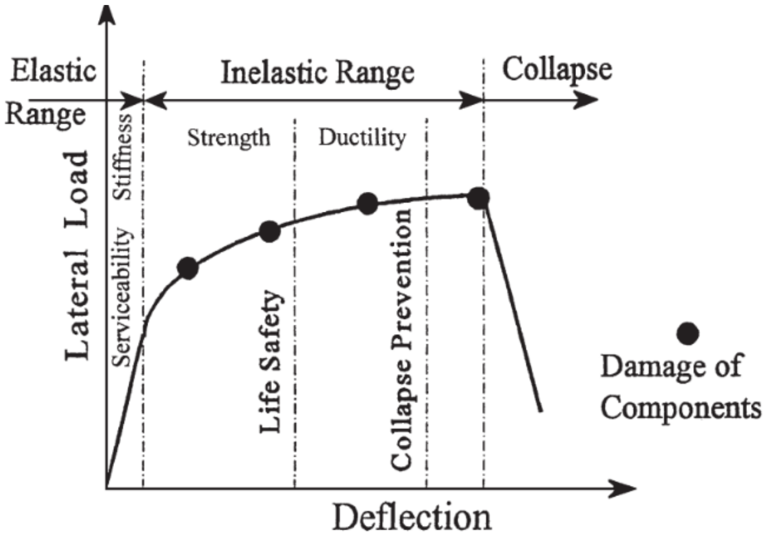


Figure 7: Typical performance curve for the structure[16]

A fundamental aspect of these methods involves defining nonlinear structural component models using monotonic backbone curves. These curves represent the characteristic force-deformation behavior of the components, derived from detailed seismic parameters. FEMA 356 specifies the parameters for these backbone curves, which govern the nonlinear moment-rotation behavior of reinforced concrete beam-columns. These parameters are determined by factors such as the longitudinal and transverse reinforcement, along with the axial and shear demands.[17] While these models are somewhat limited due to their highly idealized nature and generally conservative approach in deterministic response evaluations, they are recognized for their extensive coverage and their capacity to simulate the full range of behavior for various structural components across all major types of building construction. Equally important is the integration of the recommendations for element modeling into formal nonlinear assessment procedures.[18]

Significant progress has been made in the field of collapse assessment methods across several areas. Scientists have focused on two main areas: the P-Delta effect and nonlinear models. They've also tried to combine all these factors into a single, comprehensive approach to understanding building collapse. Here's a summary of some important research in this field.[19]

2.2.1 P-Delta Effects

Several methods have been suggested or applied in research studies to assess the collapse resistance of structures, by modeling them as single-degree-of-freedom systems. Takizawa and Jennings (1980) employed a single-degree-of-freedom model to evaluate the maximum strength of ductile reinforced concrete frame structures under the combined effects of severe ground shaking and gravity loads.[20] The study of global building collapse started by considering how the p-Delta of the building can amplify its shaking during an earthquake. Even when models assume that buildings can resist some shaking after they've been damaged, their overall stiffness can still become negative if the shaking is severe enough, leading to collapse because of p-Delta. Jennings and Husid, in 1968, examined a simple, one-story building with flexible supports at the end of columns. They used a model that assumed the building would behave in a predictable way when damaged.[21]. Researchers asserted that the duration of seismic shaking significantly affects the likelihood of collapse. This conclusion was drawn without considering cyclic deterioration behavior, but rather based on the observation that the probability of collapse increases when the loading path remains on a backbone curve with a negative slope for a prolonged period.[19] Sun conducted a study examining the effect of gravity on the dynamic behavior of a single degree of freedom (SDOF) system and its impact on changes in the system's period. The study demonstrated that the maximum displacement a system can endure without collapsing is strongly associated with the stability coefficient and the system's yield displacement.[22] In 1986, Bernal conducted a detailed study on how much earthquakes can amplify the shaking of buildings. He suggested factors based on the difference in shaking when you do and don't consider the building's P-Delta.[23] The study focused on simple buildings and assumed they behaved in a predictable way when damaged. It found that the amount of shaking amplification wasn't related to how quickly the building swayed. McRae, in 1994, continued this research by studying more complex buildings and considering how their p-Delta could affect shaking.[24] Bernal (1992, 1998), in his research, examined multi-story buildings with flexible frames. He found that the amount of shaking a building can withstand before collapsing depends a lot on how it's designed.

2.2.2 Degrading Hysteretic Models

The need for improved analytical models in seismic demand assessment studies is evident from Figure 8, which shows a consistent load-displacement response along with an overlaid quasi-static cycle response of a reinforced concrete column. The tests showed that the strength of the material first increases, then decreases. When the material is subjected to repeated stress, it becomes weaker over time, even if it hasn't been damaged. This weakening can happen even after the material has reached its maximum strength. The stiffness of the material also decreases with repeated stress.[25]

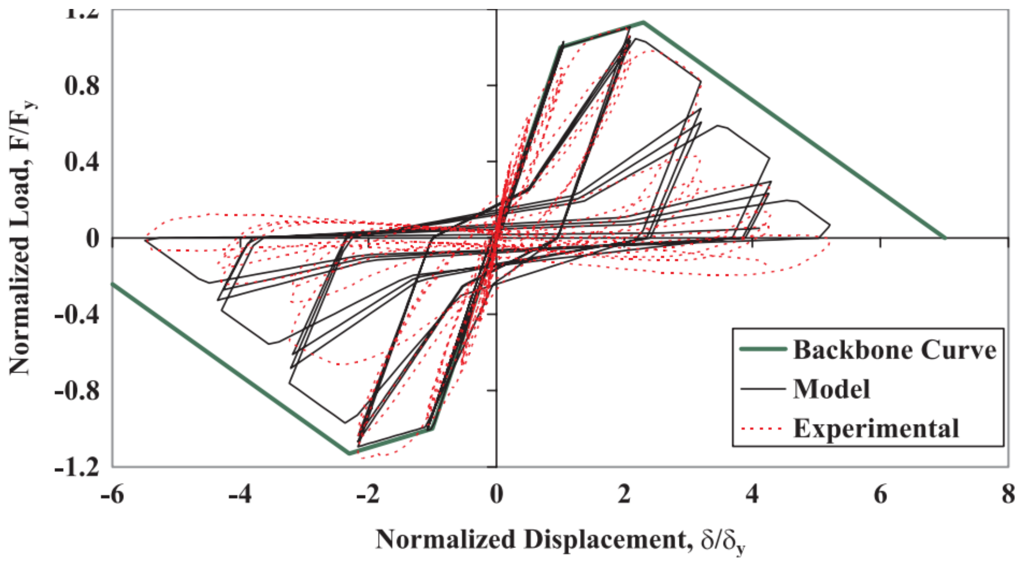


Figure 8: Monotonic and cyclic experimental response of a reinforcement concrete column[25]

Bilinear elastic-plastic models were originally preferred for hysteresis modeling due to their straightforwardness. The first model to account for the softening of reloading stiffness was introduced by Clough and Johnston in 1965, as shown in Figure 9.[26]

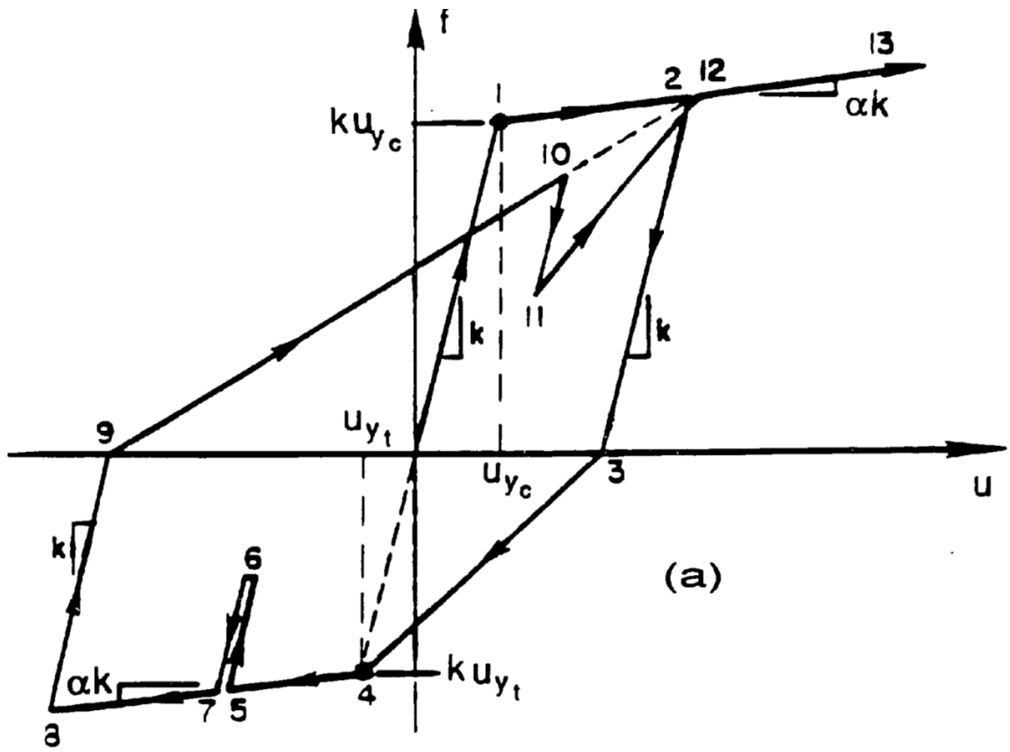


Figure 9: The Clough's hysteretic model for reinforced concrete structures[26]

This model includes the deterioration of reloading stiffness, which is influenced by the maximum displacement along the loading path. Due to this specific characteristic, the model is often referred to as the peak-oriented model. Mahin and Bertero made slight modifications to the original version in 1975.[27] In 1970, Takeda developed a model featuring a trilinear backbone that reduces stiffness during unloading, depending on the system's maximum displacement. This model is specifically designed for reinforced concrete (RC) components, with a trilinear envelope that includes a segment dedicated to uncracked concrete. Additionally, smooth hysteretic models, such as the Wen-Bouc model, have been developed to account for continuous changes in stiffness due to yielding and sudden changes caused by unloading, alongside models with piecewise linear behavior.[28] Empirical studies have shown that the hysteresis behavior depends on several structural characteristics that significantly influence deformation and energy dissipation. As a result, researchers have developed more adaptable models, such as the smooth hysteretic deteriorating mechanism introduced by Sivaselvan. This model provides guidelines for the reduction of both rigidity and strength and accounts for pinching, though it lacks a negative stiffness element. The mechanism developed by Song and Pincheira effectively represents the cyclic degradation of both strength and stiffness, using dissipated hysteretic energy as a basis. This model primarily focuses on peak responses and incorporates pinching based on degradation parameters. The backbone curve includes a segment with negative stiffness after the post-capping phase, followed by a residual strength branch. Since the initial backbone curve remains unchanged, only unloading and accelerated cyclic deterioration are considered as modes of degradation.[19] In the evaluation and strengthening of existing buildings based on their earthquake performance, as well as in the design of new buildings to resist seismic forces, components primarily impacted by bending are assessed by comparing the expected deformations from earthquakes to the maximum allowable deformations. To do this effectively, it is crucial to understand how flexure-controlled members deform at critical points in their force-deformation relationship, specifically during yielding and ultimate conditions. This understanding relies on the member's geometry, material properties, reinforcement, and axial load. Additionally, it is essential to compare the seismic deformation of each member with the deformation limits, using accurate effective stiffness values for all members of the lateral load-resisting system. The default stiffness values prescribed in current seismic design codes for new buildings often overestimate the stiffness of structural members. This overestimation is intentional to ensure safety in force-based design methods. However, these default values tend to underestimate the actual seismic deformation demands and are not appropriate for displacement-based evaluation or design, which could potentially compromise safety.[29]

The beam-column element model developed by Ibarra, Medina, and Krawinkler in 2005 and 2003 features a trilinear monotonic backbone. This backbone, along with its associated hysteresis rules, allows for versatile modeling of cyclic behavior, as illustrated in Figure 10. A key aspect of this model is the inclusion of a negative stiffness branch in the post-peak response, which facilitates the modeling of strain-softening behavior seen in physical phenomena such as concrete crushing, rebar buckling and fracture, and bond failure. The model also incorporates four fundamental mechanisms of cyclic deterioration: degradation of strength in the inelastic strain-hardening branch, degradation of strength in the post-peak strain-softening branch, accelerated deterioration of reloading stiffness, and deterioration of unloading stiffness.

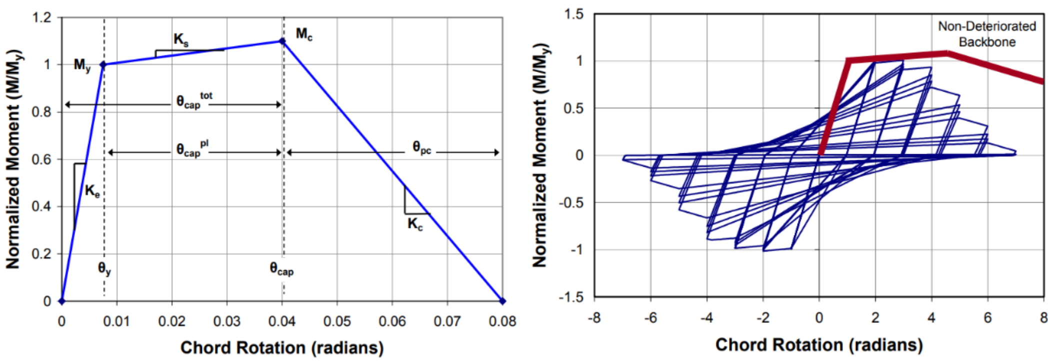


Figure 10: Monotonic and cyclic behavior of component model used in calibration study- model developed by Ibarra, Medina, and Krawinkler[18]

The peak-oriented cyclic response rules inherently account for the deterioration of additional reloading stiffness. Cyclic degradation is governed by an energy index that includes two key parameters: the normalized energy dissipation capacity and an exponent term that defines the variation in the rate of cyclic deterioration as damage accumulates. This element model was integrated into OpenSees by Altoontash in 2004.[18]

2.3 Material Behavior

2.3.1 Hysteretic and Damage Modeling of Steel Reinforcing Bars

The hysteresis behavior of reinforcing and prestressing steel bars affects the hysteresis of structural concrete elements. A failure in a reinforcing bar can be seen as a failure of the entire structural member. It is crucial to accurately model both the hysteresis and fatigue characteristics of these bars. Kent and Park performed an analysis of the stress-strain response of mild steel under cyclic loading. Their study involved testing reinforcing bars with different loadings to explore various initial strains and loading/unloading sequences in both tension and compression. [30] Cheng and Mander conducted a study focusing on the deteriorating properties of steels with yield stresses between 50 ksi and 120 ksi. Their model was improved by integrating damage relationships. In their study, they used the Menegotto-Pinto equation (1973), as applied by Mander (1984), to represent the stress-strain relationships during both loading and unloading. Figure 11 shows a comparison between experimental results and those simulated using their equations, illustrating the concept of high-strength bars.[31]

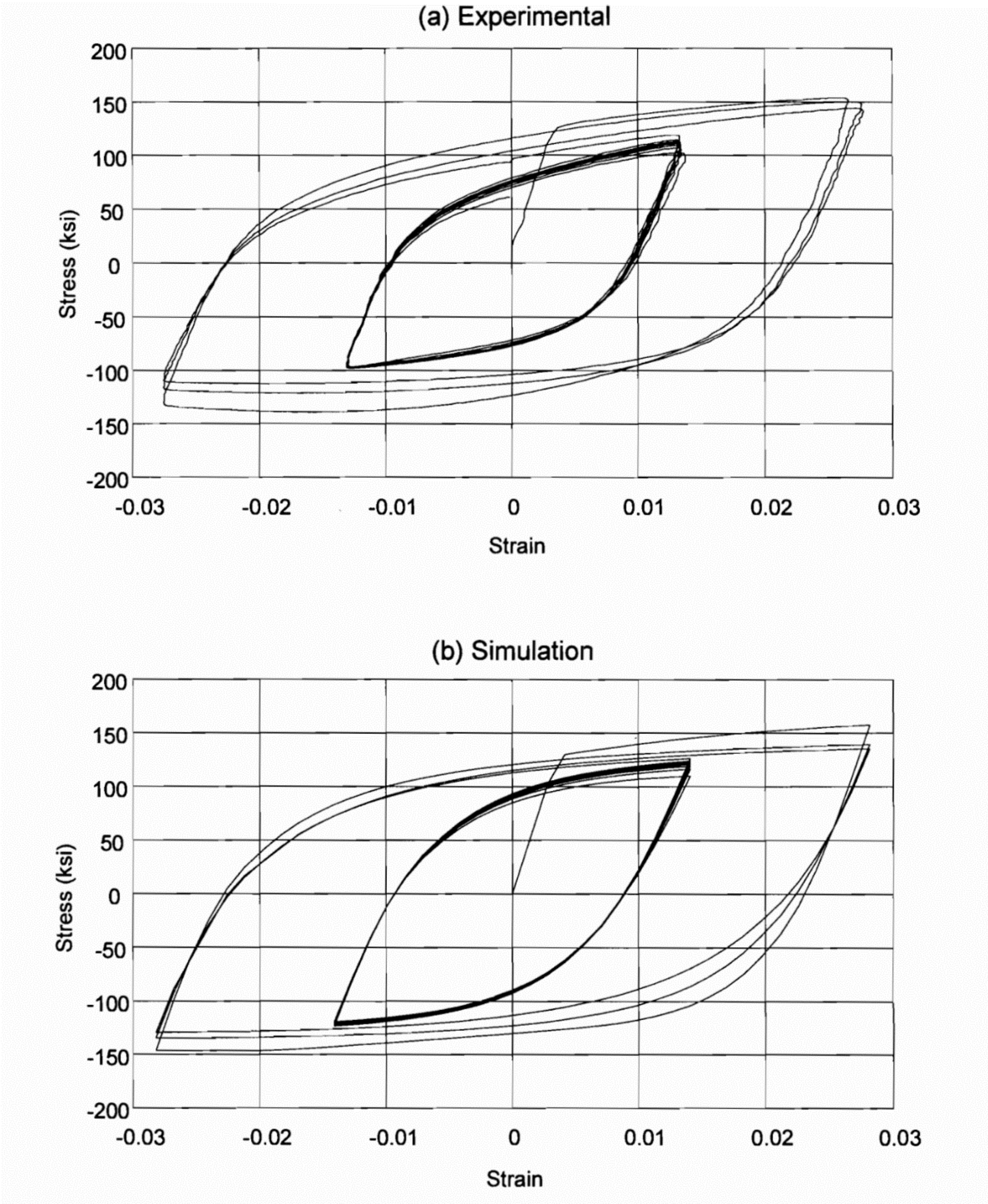


Figure 11: a) Experimental b) Simulated[31]

The hysteresis behavior of reinforcing and prestressing steel bars significantly influences the performance of structural concrete elements. Accurate modeling of these bars' hysteresis and fatigue properties is vital, as evidenced by research from Kent and Park, and Cheng and Mander. Their studies examined the stress-strain response and deterioration of steel under cyclic loading, underscoring the need for precise simulations, particularly for high-strength bars.

2.3.2 Modeling the Stress-Strain Cyclic Behavior of Concrete

When developing a computer program to simulate the cyclic behavior of concrete elements, it is crucial to incorporate all the hysteric features of both confined and unconfined concrete. A multitude of researchers have dedicated their time to empirically and analytically elucidate the characteristics of concrete.[31]

Under the influence of operational loads, a reinforced concrete structure may exhibit cracks in some components. Empirical experiments have demonstrated that concrete subjected to strain exhibits a cyclic pattern of activity that is akin to its response under compression. Therefore, it was deemed essential to provide a detailed analysis of the hysteric properties of concrete when subjected to both compression and tension. Special attention has also been given to the transition between the initiation and cessation of fractured areas. This problem has not been sufficiently addressed in prior models. Many current models presuppose an abrupt cessation of cracks accompanied by a fast alteration in the section modulus. Experiments conducted on weakly loaded columns do not provide evidence to support such a quick transformation.[31]

Multiple researchers have demonstrated and it is widely acknowledged that the use of transverse reinforcement enhances both the structural integrity and the ability of concrete to deform without fracturing. Various models have been proposed to explain the impact of confinement on the characteristics of confined concrete. The mechanics of passive confinement, achieved through reinforcing steel, has been effectively elucidated by Sheikh and Uzumeri (1980) for square sections with straight hoops. Mander (1988a) have successfully explained this phenomenon for all scenarios, including rectangular sections with hoops and ties, as well as circular sections with either spirals or hoops.[31] Figure 12 illustrates the outcomes of confining concrete and not confining concrete, as demonstrated by Mander. An equation is employed to compute the characteristics of concrete.[32]

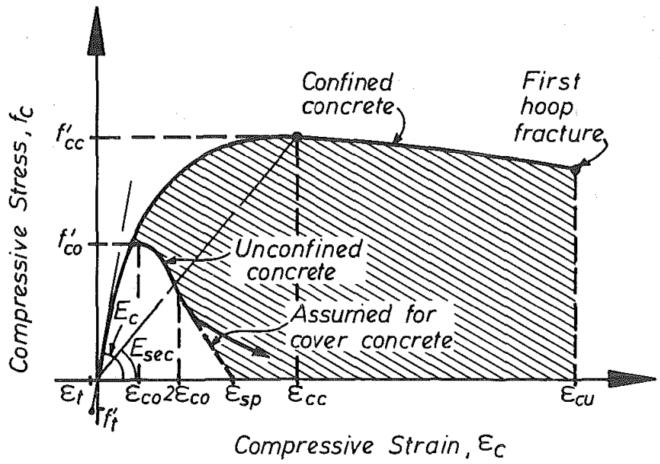


Figure 12: Stress-Strain model proposed for monotonic loading of confined and unconfined concrete[32]

In developing a computer program to simulate the cyclic behavior of concrete elements, it is essential to incorporate all hysteretic features of both confined and unconfined concrete. Numerous studies have empirically and analytically examined the characteristics of concrete under operational loads, highlighting the importance of accurately modeling the transition between crack initiation and cessation. This is particularly critical as transverse reinforcement has been shown to enhance the structural integrity and deformation capacity of concrete. Figure 12 compares the outcomes for confined and unconfined concrete, demonstrating the impact of confinement.

Chapter 3

METHODOLOGY

3.1 Introduction

In this chapter, the considered structures and their corresponding attributes will be clarified initially. Regularity is exhibited by the structures, which incorporate dual systems, including unique moment frames capable of withstanding a minimum of 25% of the mandated seismic force, as well as special reinforced concrete shear walls. The constructions are comprised of 7, 10, and 15 stories. The process by which the design is conceptualized and developed will be explained. Analysis is performed using the Minimum Design Loads and Associated Criteria for Buildings and Other Structures (ASCE7-16), and design is carried out according to the Building Code Requirements for Structural Concrete (ACI318-19). Additionally, Etabs software is utilized for the design work. Guidance is offered on the process of utilizing and calibrating seismic ground motions for performing response-history studies based on the NIST GCR 11-917-15 publication. Subsequently, for the purpose of conducting nonlinear response history analysis, two different assumptions are made: firstly, that the structure is attached to the ground, and secondly, that the interaction between the soil and the structure is taken into account. The methodologies developed by researchers at the Pacific Earthquake Engineering Research (PEER) Center are utilized to investigate the behavior of soil and its interaction with structures. Their simulation has the ability to faithfully depict the properties of soil, such as its elastic and flexible nature, and accurately simulate the upward displacement of a foundation. To conduct performance-based assessment, the methodology originally introduced by PEER researchers is once again utilized. OpenSeespy is employed to simulate soil and evaluate structure performance.

3.2 Structural Analyze

The chosen structure is a regular building consisting of 7, 10, and 15 stories, as defined in ASCE7-16.[33] It has a dual system contain special shear wall and special frame base on definition ACI318-19[34] to resistant the earthquake. According to the ASCE7-16, in a dual system, the moment frames must have the ability to withstand a minimum of 25% of the seismic forces specified in the design.[33] The plane of structure is depicted in Figure 13. Shear walls are positioned on all four sides of the plane, with the staircase located in the center of the plane. Each segment of this construction measures 5 meters.

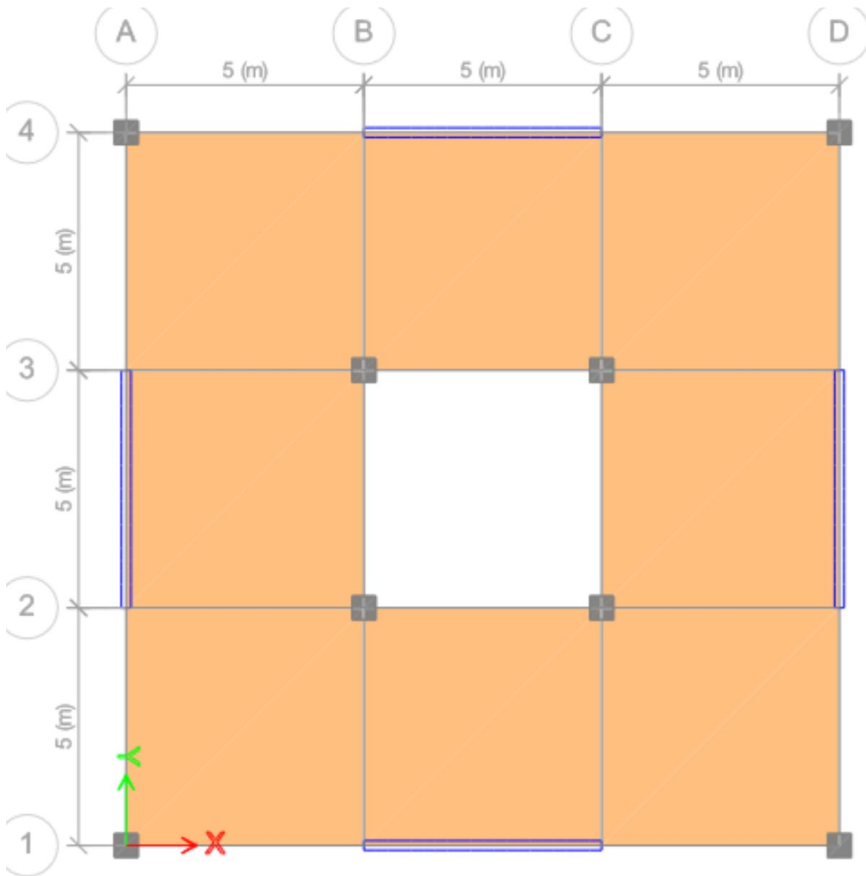


Figure 13: Plan of structure

In Appendix D, Figures 107, 108, and 109 provide detailed 3D models of the structures, fully illustrating key aspects of the work.

The spectrum analysis and design of this structure utilize Hazard tools, offered by the National Emergency for the USA, to assess the hazardous location. Figure 14 represents the spectral analysis for clay soil, while Figure 15 represents the analysis for sand.[33]

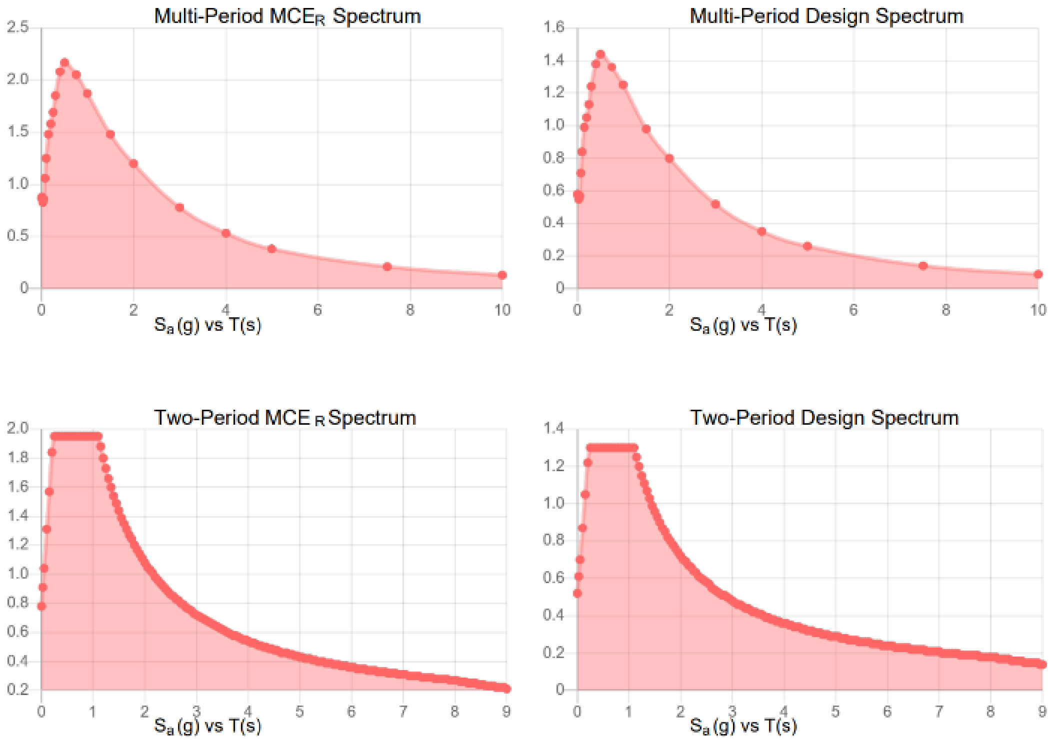


Figure 14: MCE$_R$ and design Spectrum for clay[33]

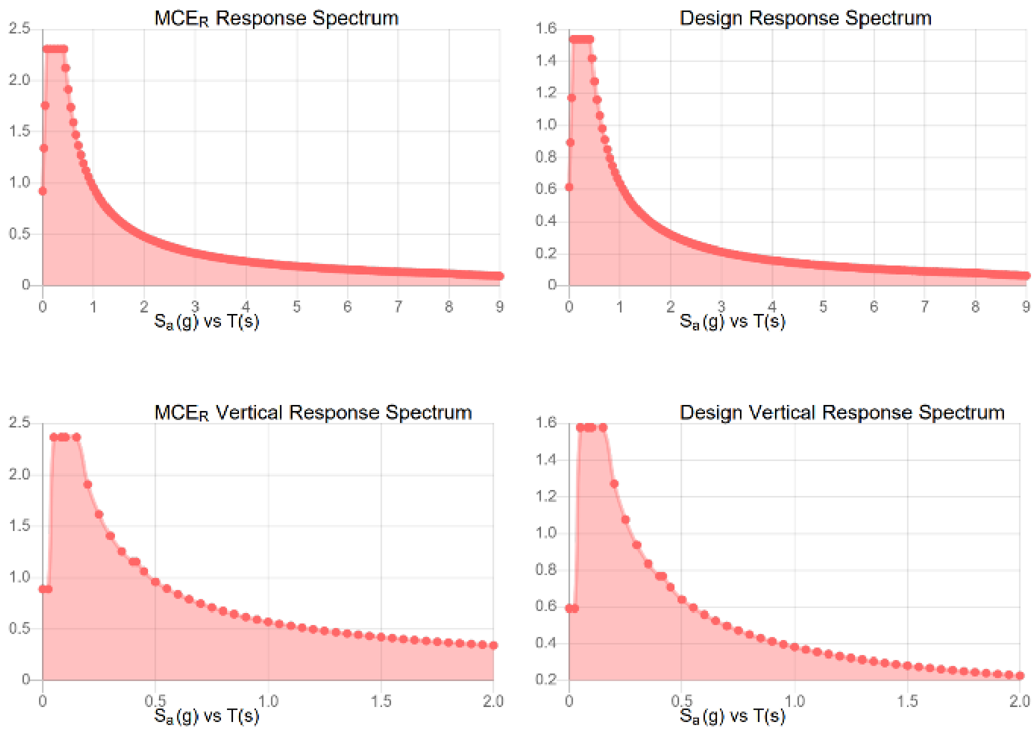


Figure 15: MCE_R and design Spectrum for sand[33]

The design of the seismic force-resisting systems was based on the coefficients and factors provided in Table 12.2-1 of ASCE7-16 illustrates in table 1.[33]

Table 1: Design coefficients and factors for seismic force-resisting systems [33]

Response Modification Coefficient, R	Overstrength Factor, Ω0	Deflection Amplification Factor, Cd
7	2.5	5.5

The properties of materials are provided in table 2 for concrete 34000KN/m^2 which contains f_c, E, G and Poisson's ratio and table 3 for steel bars A615Gr60 which contains E, f_y

Table 2: Properties of concrete

Material	$E(MN/m^2)$	$G(MN/m^2)$	$f_c(MN/m^2)$	Poisson's ratio
Concrete 34000 kN/m^2	27,800	11,580	34.4	0.2

Table 3: Properties of steel bar

Material	E (MN/m^2)	f_y(MN/m^2)
A615Gr60	200,000	413

In tables 4 to 9 the properties of columns are provided for both type of soil (type C and type D)

Table 4: Columns properties for 7 story on clay (type D)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2	0.4*0.4	16 #20
3-4-5	0.4*0.4	12 # 20
6-7	0.4*0.4	8 # 20

Table 5: Columns properties for 7 story on sand (type C)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2-3-4	0.4*0.4	12 # 20
5-6-7	0.4*0.4	8 # 20

Table 6: Columns for 10 story on clay (type D)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2-3-4	0.50*0.50	20 # 20
5-6-7-8	0.50*0.50	16 # 20
9-10	0.50*0.50	12 # 20

Table 7: Columns for 10 story on sand (type C)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2-3-4	0.50*0.50	16#20
5-6-7-8	0.50*0.50	12#20
9-10	0.50*0.50	8#20

Table 8: Columns for 15 story on clay (type D)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2-3-4	0.8*0.8	24 #20
5-6-7-8	0.8*0.8	20#20
9-10-11-12	0.6*0.6	16#20
13-14-15	0.6*0.6	12#20

Table 9: Columns for 15 story on sand (type C)

Story	Dimension of column(m)	The number and size of bar(mm)
1-2-3-4	0.5*0.5	20 #20
5-6-7-8	0.5*0.5	16#20
9-10-11-12	0.5*0.5	12#20
13-14-15	0.5*0.5	8#20

For beams, I divided two group beams one for beams which located in outer and beams located inner. The properties of these beams are illustrated in table 10 to 15 for 7 story,10 and 15 stories.

Table 10: Beams for 7 story on clay (type D)

Story	Dimension Of beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom Out(mm)	Size and Number of Top-In(mm)	Size and Number of Bottom-In(mm)
1	0.4*0.4	4#20	3#20	3#20	2#20
2	0.4*0.4	5#20	3#20	3#20	2#20
3, 4, 5, 6, 7	0.4*0.4	7#20	5#20	3#20	2#20

Table 11: Beams for 7 story on sand (type C)

Story	Dimension Of beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom Out(mm)	Size and Number of Top-In(mm)	Size and Number of Bottom-In(mm)
1-2	0.4*0.4	3#20	2#20	3#20	2#20
3-4-5-6-7	0.4*0.4	4#20	2#20	3#20	2#20

Table 12: Beams for 10 story on clay (type D)

Story	Size of Beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom-Out(mm)	Size and Number of -Top-In(mm)	Size and Number of Bottom-In(mm)
1	0.5*0.5	7#20	5#20	5#20	3#20
2	0.5*0.5	10#20	8#20	7#20	5#20
3	0.5*0.5	12#20	10#20	9#20	7#20
4, 5, 6, 7	0.5*0.5	10#20	9#20	9#20	7#20
8, 9, 10	0.*0.5	9#20	8#20	7#20	5#20

Table 13: Beams for 10 story on sand (type C)

Story	Size of Beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom-Out(mm)	Size and Number of -Top-In(mm)	Size and Number of Bottom-In(mm)
1	0.5*0.5	5#20	3#20	3#20	2#20
2	0.5*0.5	6#20	4#20	3#20	2#20
3	0.5*0.5	7#20	5#20	4#20	3#20
4-5-6-7	0.5*0.5	6#20	4#20	4#20	3#20
8-9-10	0.5*0.5	5#20	4#20	4#20	3#20

Table 14: Beams in 15 stories on clay (type D)

Story	Size of Beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom-Out(mm)	Size and Number of Top-In(mm)	Size and Number of Bottom-In(mm)
1	0.6*0.6	7 #20	6#20	4#20	3#20
2	0.6*0.6	10#20	10#20	6#20	5#20
3	0.6*0.6	13#20	13#20	8#20	7#20
4-5-6-7-8	0.6*0.6	14#20	14#20	8#20	7#20
9-10	0.6*0.6	12#20	12#20	5#20	5#20
11	0.6*0.6	10#20	10#20	5#20	4#20
12	0.6*0.6	9#20	9#20	4#20	4#20
13-14-15	0.6*0.6	7#20	7#20	4#20	4#20

Table 15: Beams in 15 stories on sand (type C)

Story	Size of Beam(m)	Size and Number of Top-Out(mm)	Size and Number of Bottom-Out(mm)	Size and Number of Top-In(mm)	Size and Number of Bottom-In(mm)
1	0.5*0.5	4 #20	3#20	3#20	3#20
2	0.5*0.5	5#20	4#20	4#20	3#20
3	0.5*0.5	6#20	5#20	5#20	3#20
4-5-6-7-8	0.5*0.5	7#20	6#20	5#20	3#20
9-10	0.5*0.5	6#20	5#20	4#20	3#20
11	0.5*0.5	6#20	5#20	4#20	3#20
12	0.5*0.5	5#20	4#20	4#20	3#20
13-14-15	0.5*0.5	5#20	4#20	4#20	3#20

The properties of shear wall for buildings of 7, 10, and 15 stories are listed in table 16 to 21. These tables include information on the thickness, specific edges, and the ratio of bars.

Table 16: Properties of shear wall in 7 story on clay (type D)

Story	Thickness(m)	Boundary condition from each side(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	1	1	0.054	0.0025
2	0.2	No	1	0.047	0.0025
3	0.2	No	1	0.042	0.0025
4	0.2	No	1	0.026	0.0025
5-6-7	0.2	No	1	0.0025	0.0025

Table 17: Properties of shear wall in 7 story on sand (type C)

Story	Thickness (m)	Boundary condition from each side(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	No	1	0.023	0.0025
2	0.2	No	1	0.020	0.0025
3	0.2	No	1	0.016	0.0025
4	0.2	No	1	0.0025	0.0025
5-6-7	0.2	No	1	0.0025	0.0025

Table 18: Properties of shear wall in 10 story on clay (type D)

Story	Thickness(m)	Boundary Condition(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	1	1	0.056	0.0025
2	0.2	1	1	0.045	0.0025
3	0.2	1	1	0.037	0.0025
4	0.2	1	1	0.062	0.0025
5	0.2	No	1	0.053	0.0025
6	0.2	No	1	0.032	0.0025
7	0.2	No	1	0.019	0.0025
8-9-10	0.2	No	1	0.0025	0.0025

Table 19: Properties of shear wall in 10 story on sand (type C)

Story	Thickness(m)	Boundary Condition(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	1	1	0.044	0.0025
2	0.2	1	1	0.035	0.0025
3	0.2	1	1	0.033	0.0025
4	0.2	No	1	0.025	0.0025
5	0.2	No	1	0.024	0.0025
6	0.2	No	1	0.017	0.0025
7-8-9-10	0.2	No	1	0.0025	0.0025

Table 20: Properties of shear walls in 15 story on clay (type D)

Story	Thickness(m)	Boundary Condition(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	1	1	0.052	0.0025
2	0.2	1	1	0.052	0.0025
3	0.2	1	1	0.052	0.0025
4	0.2	1	1	0.052	0.0025
5	0.2	1	1	0.045	0.0025
6	0.2	1	1	0.048	0.0025
7	0.2	1	1	0.041	0.0025
8,9,10,11, 12,13,14,15	0.2	No	1	0.0025	0.0025

Table 21: Properties of shear walls in 15 story on sand (type C)

Story	Thickness(m)	Boundary Condition(m)	Special edge from each side(m)	Ratio of bars in special edge	Ratio of bars in web
1	0.2	1	1	0.050	0.0025
2	0.2	1	1	0.052	0.0025
3	0.2	1	1	0.052	0.0025
4	0.2	1	1	0.050	0.0025
5	0.2	1	1	0.045	0.0025
6	0.2	1	1	0.052	0.0025
7	0.2	1	1	0.045	0.0025
8,9,10,11, 12,13,14,15	0.2	No	1	0.0025	0.0025

The aforementioned properties were derived using the ETABS software through static seismic analysis, considering two types of soil: Type C and Type D. For all structures, the assumption of a dual system requires that the frame alone must be capable of resisting 25 percent of the earthquake load without the contribution of the shear wall. Additionally, in all designs, a beam-to-column strength ratio of 1.2 was consistently applied, ensuring a "strong column-weak beam" behavior that enhances seismic performance by promoting energy dissipation in beams while safeguarding the structural integrity of columns.

3.3 Ground Motion Selection and Scaling

Once the design process was completed, the selection of ground motion was finalized. The optimal quantity of ground motions is contingent upon the specific application, including the structural response(s) that need to be forecasted, whether mean values or distributions of responses are preferred, the level of accuracy required for estimating mean and variance values, the potential prediction of maximum responses or collapse responses, and the anticipated extent of inelastic response. When choosing ground movements to scale to a target spectrum for near-field sites, the most crucial considerations are the spectral shape and the potential occurrence of velocity pulses. [35]

This study focuses on ground motion occurring at near-fault sites. Such sites are classified as near-fault if they meet one of the following criteria: 1. Within 9.5 miles (15 km) of the surface projection of an active fault known to produce magnitude 7 or larger earthquakes, or 2. Within 6.25 miles (10 km) of the surface projection of an active fault capable of producing magnitude 6 or larger earthquakes.[33]. Ground motions should be chosen from events that occur within the same tectonic domain as the target spectrum. They should also match the magnitude and fault distance of the target spectrum and have a similar spectral shape. The selected ground motions must correspond to the magnitudes, source characteristics, fault distances, and site conditions that define the target spectrum.[33]

11 ground motions each for clay soil and sand soil are selected from the Peer Ground Motion Database. Their properties such as name, year, and the factors that influenced the selection of these ground motions, such as the type of soil (related to Vs30), Rjb, and magnitude, which specify the near fault are included. These ground motions were chosen based on the design spectrum (Figure 14, 15). The details of these ground motions are provided in table 22 and 23, respectively.

Table 22: Properties of ground motion for clay soil

Name	Year	Mechanism	Rjb(km)	Magnitude	Vs30(m/s)
Chalfant Valley 02	1986	Strike Slip	6.44	6.19	316.19
Chi Chi	1999	ReverseOblique	9.94	7.62	258.89
Darfield	2010	Strike Slip	7.29	7	326.01
El Mayor	2010	Strike Slip	8.88	7.2	242.05
ImperialValley02	1940	Strike Slip	6.09	6.95	213.44
ImperialValley06	1979	Strike Slip	0	6.53	259.86

Kocaeli	1999	Strike Slip	1.38	7.51	297
Managua	1972	Strike Slip	3.51	6.24	288.77
Morgan Hill	1984	Strike Slip	3.45	6.19	281.61
Parkfield	2004	Strike Slip	4.36	6	340.45
Parkfield-02	2004	Strike Slip	1.66	6	364

Table 23:Properties of ground motion for sandy soil

Name	Year	Mechanism	Rjb(km)	Magnitude	Vs30(m/s)
Bam	2003	Strike Slip	0.05	6.6	487.4
Big Bear	1992	Strike Slip	7.31	6.46	430.36
Cape Mendocino	1992	Reverse	0	7.01	567.78
Chi Chi	1999	Strike Slip	6.02	6.2	553.43
Iwate	2008	Reverse	0	6.9	506.44
Kobe	1995	Strike Slip	7.08	6.9	609
Landers	1992	Strike Slip	11.03	7.28	379.32
LAquila	2009	Normal	0	6.3	552
Montenegro	1979	Reverse	0	7.1	462.23
Tabas	1978	Reverse	0	7.35	471.53
Tottori	2000	Strike Slip	9.1	6.61	616.55

Their spectrum is illustrating in figure 16 and in figure 17 depicts the spectrum for sandy soil. Then A maximum-direction spectrum will be created for each pair ofhorizontal ground motion components. Every ground motion must be proportionally adjusted, with the same scaling factor applied to both horizontal components.[33, 35]

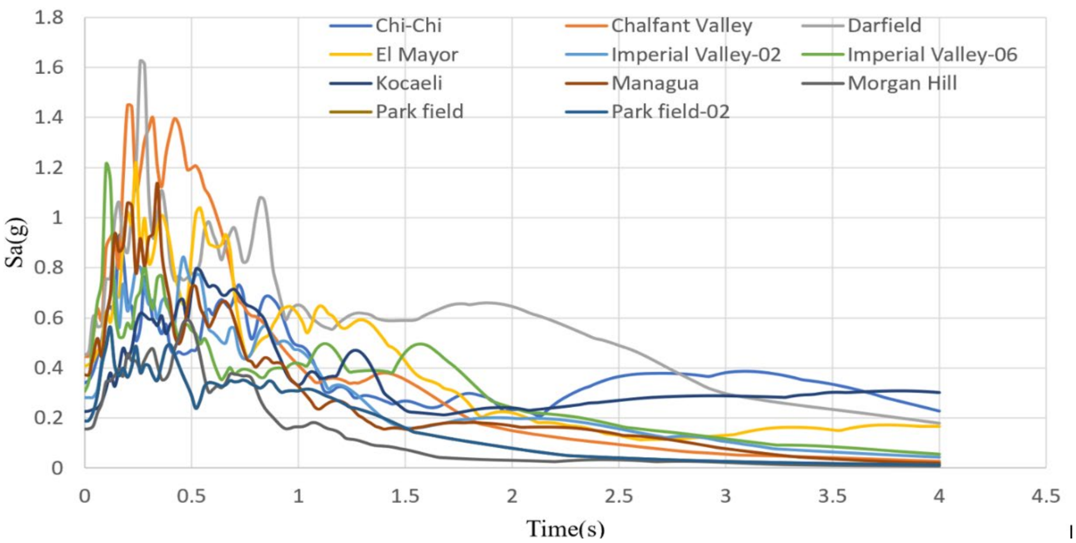


Figure 16: Spectrum of ground motions for clay soil

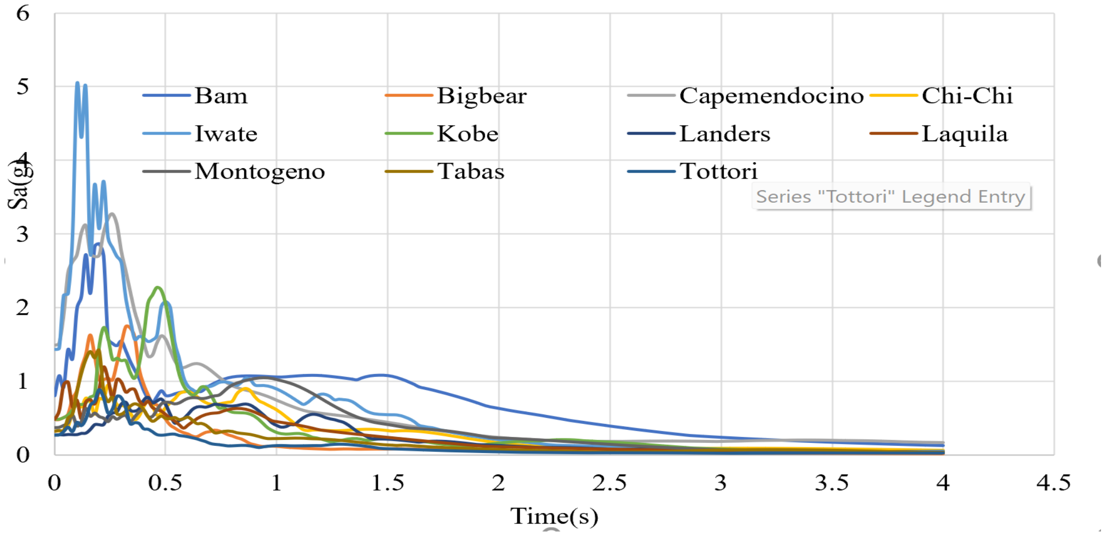


Figure 17: Spectrum of ground motions for sandy soil

Among all the ground motions analyzed, for each soil type, three ground motions are selected to report based on their potential to push the structure into nonlinear behavior. For Type D soil, the chosen ground motions are Darfield, Chi-Chi, and El Mayor, while for Type C soil, they are Kobe, Chi-Chi, and Iwate. A key parameter in this selection is energy flux, which represents the rate at which seismic energy is transferred through the ground. Energy flux provides insight into the intensity and distribution of seismic energy as it impacts the structure, making it a critical factor in assessing the potential for nonlinear response and damage. In Figures 18 and 19, I present diagrams illustrating the energy flux of the Chi-Chi ground motion recorded at two different stations, each located on different soil types.

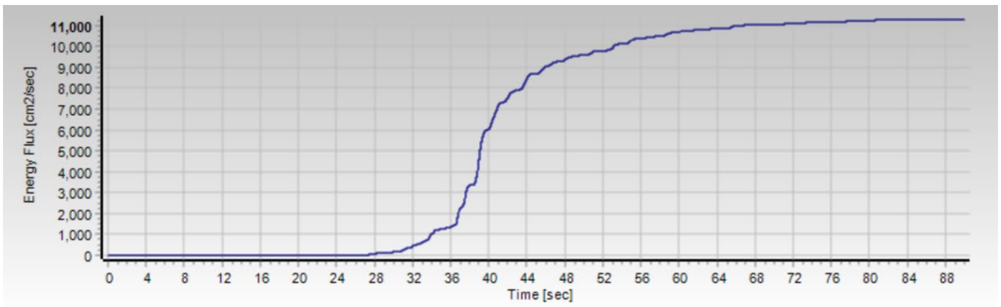


Figure 18: Energy flux for Chi-Chi ground motion which recorded on soil type D

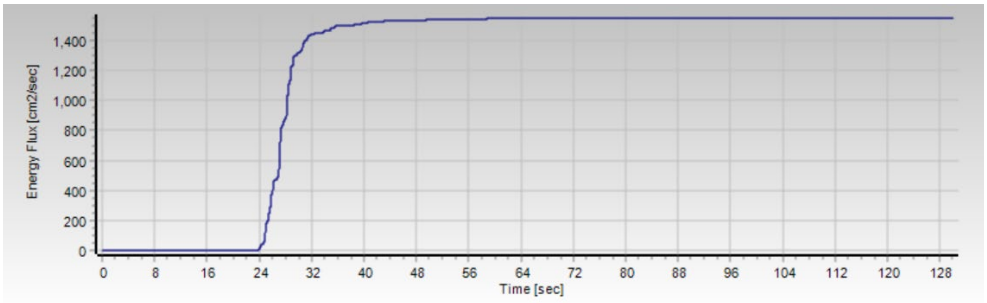


Figure 19: Energy flux for Chi-Chi ground motion which recorded on soil type C

These diagrams clearly show the differences in energy flux between Type C and Type D soils, highlighting how soil conditions can significantly influence the seismic energy transmitted to a structure. Understanding energy flux allows for better estimation of the expected damage, as higher energy flux values are typically associated with stronger ground shaking and greater potential for structural damage.

3.4 Modeling Soil and Foundation

The model is a simplified representation of a shallow foundation. It's like a flexible beam resting on springs. The beam can bend and twist in all directions, and the springs represent the ground's resistance to movement. Each nonlinear Winkler spring is modeled as an independent six-dimensional zero-length element within the OpenSees framework. Nonlinearity in the model is captured using modified versions of the Qzsimple1, PySimple1, and TzSimple1 material models, as implemented by Boulanger in OpenSees.[8]

The axial and moment responses are measured using a network of nonlinear q-z springs, while p-x and t-x springs are placed horizontally to record passive and sliding resistances, respectively. The Qzsimple1 material model features asymmetrical hysteresis, with stronger response on the compression side and reduced strength in tension (capped at 10% of the ultimate compressive load). It also accommodates soil-footing separation by incorporating a gap element along with elastic and plastic components. The elastic material simulates behavior at a distance from the source, whereas the plastic component represents permanent displacements near the source.[8] The PySimple1 and TzSimple1 materials are similar to the QzSimple1 material in their composition, consisting of an elastic and plastic material arranged in series, as illustrated in Figure 20. However, unlike QzSimple1, their backbone curves are symmetric, as depicted in Figure 21.[36] The equations defining the material models for PySimple1, QzSimple1, and TzSimple1 can be found in Boulanger's paper.[37, 38]

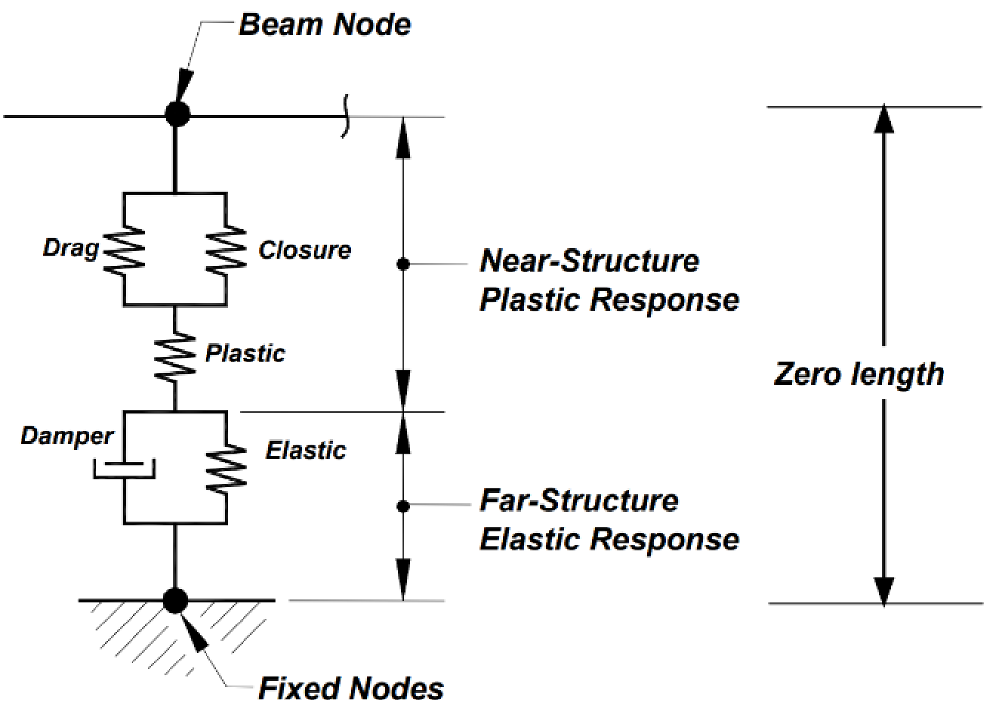


Figure 20: Zero length spring [34]

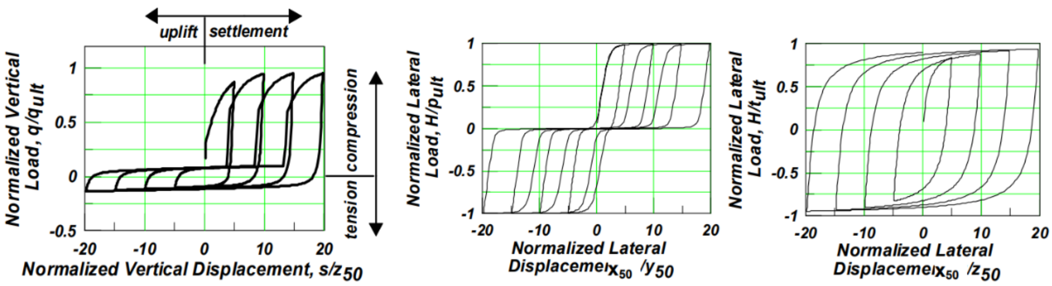
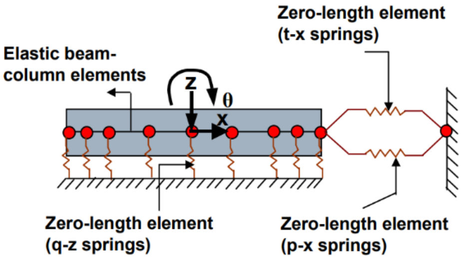


Figure 21: Cyclic behavior of uni-directional zero-length spring models: (a) axial displacement response for QzSimple1 material, (b) lateral passive response for PySimple1 material, and (c) lateral sliding response for TzSimple1 material.[36]

The shallow footing BNWF model, depicted in Figure 22, features several important characteristics. It is designed to handle nonlinear, inelastic soil behavior and geometric nonlinearity, including uplift. This model addresses various types of nonlinearities, such as nonlinear relation of moment with rotation, of shear force with sliding, and of axial force with vertical displacement. It accounts for inelastic behavior resulting from gap created between foundation and soil under cyclic loading, allowing it to accurately represent various movements of the foundation, such as rocking, settlement and sliding. Additionally, the model measures hysteretic energy dissipation and incorporates radiation damping at the foundation base. The model also supports a variable stiffness distribution along the length of the foundation, which is important for addressing the increased reaction at the ends of stiff footings subjected to vertical loads. By improving stiffness representation and achieving more precise vertical spring spacing at the edges of the footing, the BNWF model ensures accurate consideration of rotational stiffness.[36]



Axis	Force	Displacement
x	H	U
z	Q	S
θ	M	Θ

Figure 22: BNWF schematic[8]

3.4.1 QzSimple1 Material

The QzSimple1 material demonstrates asymmetrical hysteretic behavior, with its backbone curve defined by a maximum compressive load and reduced tensile strength to reflect the soil's low tensile capacity. The elastic material models long-range behavior, while the plastic component addresses short-range permanent displacements, as shown in Figure 20. To accurately model foundation uplifting, additional gap components—a drag spring and a closure spring in series—are included with the plastic components. Radiation damping effects are considered by integrating a dashpot into the elastic component for the far-field, with its viscous force proportional to the velocity of the elastic component. The backbone curve starts with an initial elastic segment, which is followed by a progressively expanding inelastic portion, as illustrated in Figure 23. The equation describing the elastic region of this backbone curve is provided by Boulanger.[39]

$$q = K_{in} Z \tag{3.1}$$

The size of the elastic area is defined by below equatin:

$$q_o = C_r q_{ult} \tag{3.2}$$

where:

K_{in} = initial stiffness for the elastic phase,

q = immediate force,

Z = immediate movement,

q_0 = force which lead to yield,

C_r = parameter governing the extent of the elastic region

In the nonlinear (post-yield) region, the backbone curve is characterized by

$$q = q_{ult} - (q_{ult} - q_0)\left[\frac{cZ50}{cZ50+|z-z0|}\right]^n \quad (3.3)$$

where:

z50 = displacement at which 50% of the maximum load is reached,

z0 = displacement at the point of yielding, c and n constitutive parameters that govern the shape of the post-yield section of the backbone curve. Although the formulas for PySimple1 and TzSimple1 are quite similar, the constants that determine the overall shape of the curve are different. (c, n and Cr), are different.[36]

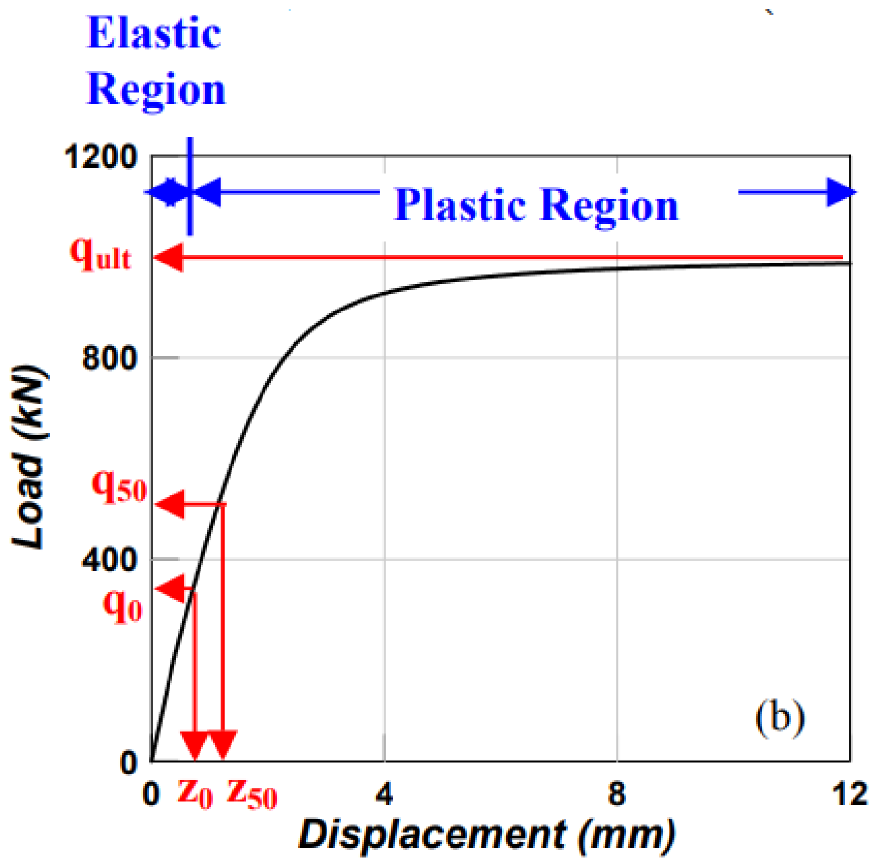


Figure 23: Nonlinear backbone curve for QzSimple1 material[36]

3.4.2 PySimple1 Material

The PySimple1 material was originally created to model how the ground resists the movement of piles. This paper uses a modified version of this model to accurately represent how shallow foundations can move and resist forces during earthquakes. This model shows a specific type of hysteresis that accurately reflects the gaps that can form between the foundation and the ground.[8]

3.4.3 TzSimple1 Material

The TzSimple1 material was created to accurately represent the friction that occurs between a pile and the ground. It shows a strong initial resistance and a wide loop in its hysteresis curve, which are typical of the way foundations slide.[8]

3.4.4 Bearing Capacity

Users have the option to either calculate the ultimate load capacities for vertical bearing, horizontal passive resistance, and horizontal sliding using the mesh generator, or they can directly specify these values as Qult, Pult, and Tult. For calculations, users must provide specific information on the footing dimensions (width (B), length (L), thickness (H)), embedment depth (Df), soil unit weight (γ), cohesion (c), angle of internal friction (Φ), and load inclination angle (β). By default, (β) is set to zero, implying no vertical load unless otherwise specified. After input or calculation, the total ultimate load capacity is internally distributed by the code to different springs based on their tributary area for vertical springs, while Pult and Tult are assigned directly to horizontal springs. The ultimate bearing capacity, when calculated internally, uses Terzaghi's bearing capacity theory (1943) and applies Meyerhof's (1963) general bearing capacity equation, which considers depth, shape, and load inclination factors.[40].

$$q_{ult} = cN_cF_{cs}F_{cd}F_{ci} + \gamma D_f N_q F_{qs}F_{qd}F_{qi} + 0.5\gamma B N_\gamma F_{\gamma s}F_{\gamma d}F_{\gamma i} \quad (3.4)$$

where

qult = maximum supportable load per unit area of the footing

c = cohesion

Nc, Nq, $N\gamma$ = load-bearing capacity coefficients,

F_{cs}, F_{qs}, $F_{\gamma s}$ = Geometry correction coefficients,

F_{cd}, F_{qd}, $F_{\gamma d}$ = Depth correction coefficients,

F_{ci}, F_{qi}, $F_{\gamma i}$ = Inclination correction coefficients.

It is important to note that this study excludes load inclination, despite its focus on seismic loading, to avoid the complexities related to the time-dependent behavior of the foundation's bearing capacity. The formulas for the elements discussed are available in foundation engineering textbooks, such as those by Das (2006) and Salgado (2006). The ultimate lateral load capacity of the PySimple1 material is determined by the total passive resisting force on the front side of the embedded footing. To compute the passive resisting force in a homogeneous backfill, a linearly varying pressure distribution can be applied. This is expressed as[41, 42]:

$$P_{ult} = 0.5\gamma K_p D_f^2 \quad (3.5)$$

where

P_{ult} = Resisting soil pressure per unit length of footing,

K_p = Coulomb passive earth pressure coefficient.

Coulomb's equations are used in this study. However, users have the option to explicitly enter the value of Pult if they prefer to use different earth pressure coefficients. The lateral load capacity of the TzSimple1 material represents the total resistance to sliding, primarily due to friction. This frictional resistance is assessed by evaluating the shear strength between the soil and the footing, which depends on the friction angle between the footing base and the soil material, as well as the cohesion at the base.[36]

$$t_{ult} = W_g \tan \delta + c\, A_f \tag{3.6}$$

where,

t_{ult} = The frictional force experienced by each unit of area on the foundation surface,

W_g = The vertical force applied at the base of the foundation,

δ = The friction angle between the foundation and the soil.

A_f = The surface area of the footing base that is in contact with the soil(=L x B).

3.4.5 Vertical and Lateral Stiffness

The stiffness of shallow bearing footings, which are rigid relative to the supporting soil, should be modeled using an uncoupled spring system, as shown in Figure 24. The spring constants should be calculated based on the guidelines provided in Figure 25.[17]

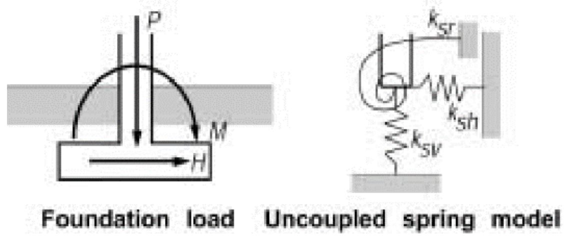
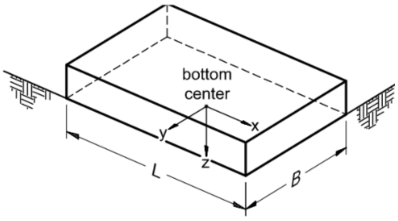


Figure 24: Uncoupled spring model for rigid footings[17]

The stiffness of a foundation depends a lot on the properties of the soil and the size of the foundation. If the soil's stiffness is estimated incorrectly, it can significantly affect the foundation's behavior. It's important to carefully choose the right value for the soil's stiffness.[36]

Degree of Freedom	Stiffness of Foundation at Surface
Translation along x-axis	$K_{x,\,sur} = \frac{GB}{2-\nu}\left[3.4\left(\frac{L}{B}\right)^{0.65} + 1.2\right]$
Translation along y-axis	$K_{y,\,sur} = \frac{GB}{2-\nu}\left[3.4\left(\frac{L}{B}\right)^{0.65} + 0.4\frac{L}{B} + 0.8\right]$
Translation along z-axis	$K_{z,\,sur} = \frac{GB}{1-\nu}\left[1.55\left(\frac{L}{B}\right)^{0.75} + 0.8\right]$
Rocking about x-axis	$K_{xx,\,sur} = \frac{GB^3}{1-\nu}\left[0.4\left(\frac{L}{B}\right) + 0.1\right]$
Rocking about y-axis	$K_{yy,\,sur} = \frac{GB^3}{1-\nu}\left[0.47\left(\frac{L}{B}\right)^{2.4} + 0.034\right]$
Torsion about z-axis	$K_{zz,\,sur} = GB^3\left[0.53\left(\frac{L}{B}\right)^{2.45} + 0.51\right]$

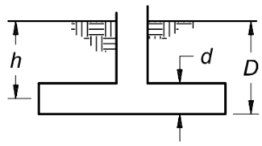
Note



Orient axes such that $L \geq B$

Degree of Freedom	Correction Factor for Embedment
Translation along x-axis	$\beta_x = \left(1 + 0.21\sqrt{\frac{D}{B}}\right)\cdot\left[1 + 1.6\left(\frac{hd(B+L)}{BL^2}\right)^{0.4}\right]$
Translation along y-axis	$\beta_y = \beta_x$
Translation along z-axis	$\beta_z = \left[1 + \frac{1}{21}\frac{D}{B}\left(2 + 2.6\frac{B}{L}\right)\right]\cdot\left[1 + 0.32\left(\frac{d(B+L)}{BL}\right)^{2/3}\right]$
Rocking about x-axis	$\beta_{xx} = 1 + 2.5\frac{d}{B}\left[1 + \frac{2d}{B}\left(\frac{d}{D}\right)^{-0.2}\sqrt{\frac{B}{L}}\right]$
Rocking about y-axis	$\beta_{yy} = 1 + 1.4\left(\frac{d}{L}\right)^{0.6}\left[1.5 + 3.7\left(\frac{d}{L}\right)^{1.9}\left(\frac{d}{D}\right)^{-0.6}\right]$
Torsion about z-axis	$\beta_{zz} = 1 + 2.6\left(1 + \frac{B}{L}\right)\left(\frac{d}{B}\right)^{0.9}$

Note



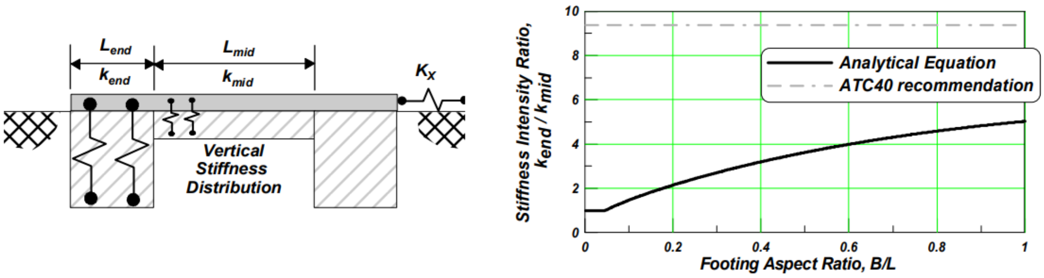
d = height of effective sidewall contact (may be less than total foundation height)

h = depth to centroid of effective sidewall contact

For each degree of freedom, calculate $K_{emb} = \beta\, K_{sur}$

Figure 25: Elastic solutions for rigid footing spring constraints[17]

The distribution and magnitude of vertical stiffness Figure 26 illustrates that the distribution and magnitude of vertical stiffness throughout the length of the footing are determined by two elements. (1) The stiffness intensity ratio, R_k, is the quotient of the stiffness at the end, K_{end}, divided by the stiffness at the midway, K_{mid}. The end length ratio, R_e, is the quotient of the end length, Lend, divided by the entire length, L. The results are displayed in Figure 26, together with the recommendations given by ATC-40 (1996). ATC-40 specifies a fixed value of R_k = 9.3. The study incorporates the suggestion of Harden et al. (2005) as illustrated in Figure 26.[36]



(a) **(b)**

Figure 26: Increased end stiffness (a) spring distribution and (b) stiffness intensity ratio versus footing aspect ratio[8]

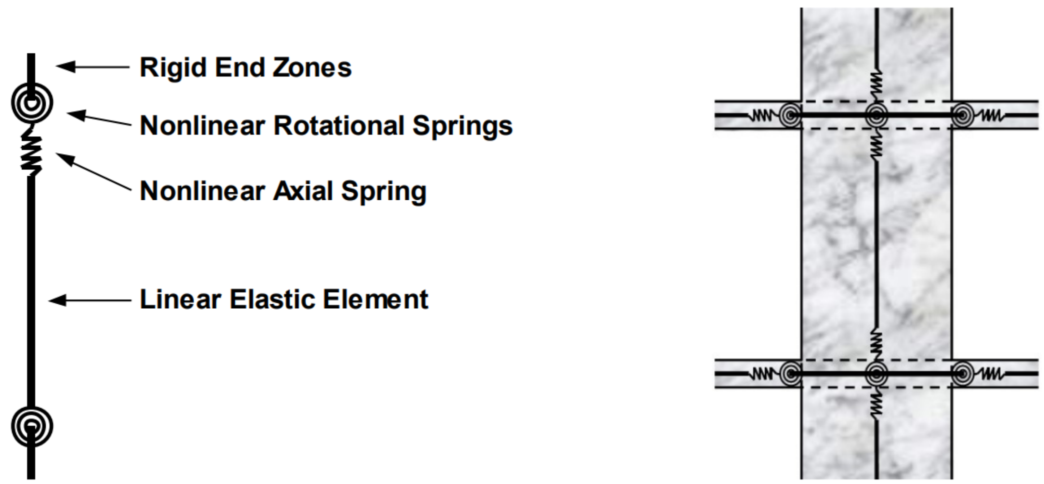
The end region Lend is defined as the length of the edge region over which the stiffness is increased. ATC-40 suggests Lend = B/6 from each end of the footing.[1]

3.5 Modeling Shear Wall

The primary objective of modeling shear walls is to precisely capture the load-deformation responses related to flexure, shear, and bond. The following sections will discuss the modeling techniques typically used in the analysis and design of tall wall structures.[43]

The growing significance of detailed modeling and simulation of the nonlinear seismic behavior of reinforced concrete (RC) structural systems stems from the adoption of performance-based approaches in contemporary seismic design codes and evaluation standards. While modeling the linear elastic response of systems with complex geometries is no longer a primary design challenge, there remains a crucial need for reliable modeling methods to accurately simulate the nonlinear hysteretic behavior of RC structural elements.[44]

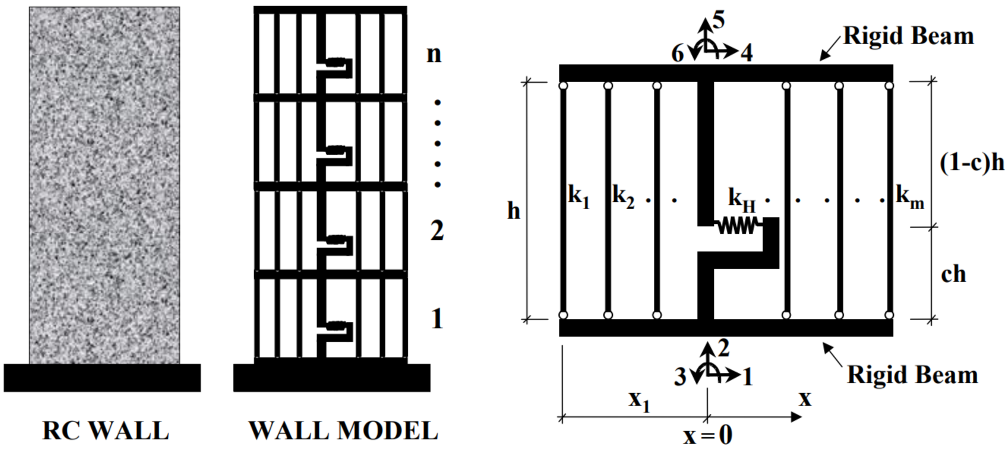
Several analytical approaches have been developed to predict the nonlinear behavior of reinforced concrete walls. The hysteresis of structural walls is commonly modeled using a simplified beam-column element aligned along the wall's centroidal axis, with rigid connections at the beam girder points. This model comprises a linear-elastic flexural element, which is extending along the length of the wall. In addition, inelastic rotational and axial springs are placed at each end of the element to accurately capture the nonlinear behavior of critical regions (Figure 27). The nonlinearity of the flexural and axial springs is defined by the relationships between moment and rotation for flexural springs, and force and deformation for axial springs.[45]



(a) Beam-column element (b) Model configuration

Figure 27: Beam-column element model[45]

Although this model is simple and computationally efficient, it does not consider the variation of the neutral axis across the depth of the wall cross-section, the rocking motion of the wall, or the proper interaction with frame members connected to the wall (both in-plane and out-of-plane). These factors are essential for accurately predicting the inelastic behavior of the walls.[46] The Multiple-Vertical-Line-Element-Model (MVLEM) effectively merges the benefits of both macroscopic and microscopic models by providing a simple formulation, numerical stability, efficiency, and relatively accurate predictions of flexural responses.[47] The structural wall is modeled using a series of (n) MVLEM elements stacked vertically, as shown in Figure 28(a). To simulate the flexural response, a multi-uniaxial-element-in-parallel model is employed, with infinitely rigid beams placed at the top of each floor level. The axial stiffness of the boundary columns is represented by two outer components (k_1) and (k_m), while the axial and flexural stiffness of the central wall panel is represented by two or more internal elements (k_2, …, k_{m-1}), as depicted in Figure 28(b). The wall's shear response is captured by integrating a horizontal spring located at the height ch (illustrated in Figure 28(b)). The shear spring's deformations are determined by the displacements across the element's six degrees of freedom (DOFs), and its response is governed by an adopted hysteretic shear force-deformation model. It is crucial to note that the behavior of the shear spring is considered independent of the longitudinal element fibers, with the assumption that axial and flexural behaviors are decoupled from the shear behavior in the element's formulation.[48]



(a) MVLE Wall Model (b) Model Element

Figure 28: Multiple-vertical-line-element-model[48]

The proposed analytical model provides the potential for improved three-dimensional and system-level modeling of reinforced concrete (RC) wall components or entire building systems, along with a flexible framework for future model refinements.[48]

3.6 Uniaxial Concrete Constitutive Behavior

The stress-strain relationship for concrete used in OpenSees[38] is the uniaxial constitutive model developed by Chang and Mander [1994]. This model is characterized by hysteresis. The Chang and Mander (1994) model is a sophisticated constitutive model that is based on rules, is generalized, and is non-dimensional. It allows for the calibration of material modeling parameters for both monotonic and hysteretic behavior. The model can accurately simulate the cyclic compression and tension behavior of both confined and unconfined concrete, including ordinary and high-strength concrete. This behavior is illustrated in Figure 29. The model specifically considers significant behavioral characteristics, including consistent hysteresis behavior during repeated compression and tension, gradual reduction in stiffness as strain values increase, and the gradual closing of cracks.[49]

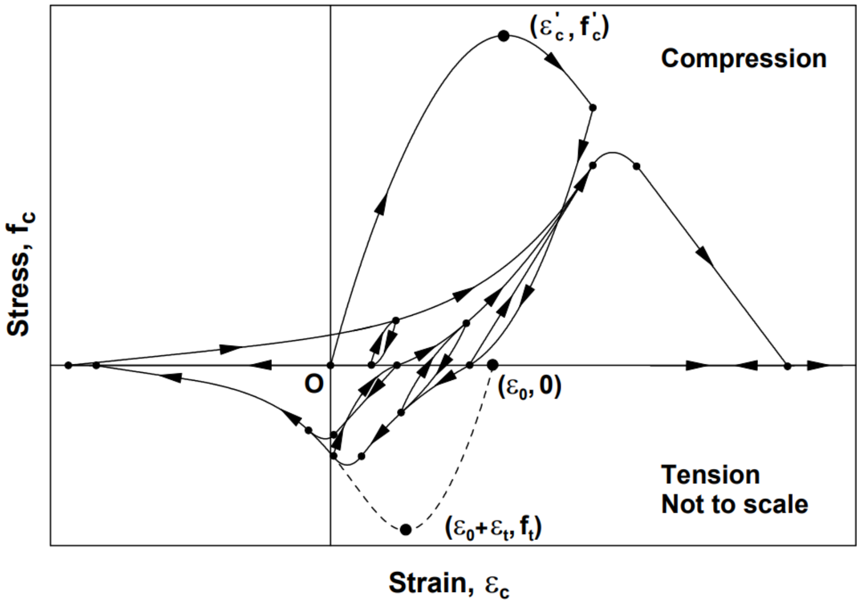


Figure 29: Hysteretic constitutive model for concrete by Chang and Mander[49]

3.7 RC Frame Model

The modeling of reinforced concrete frame elements, such as beams and columns, involved the use of elastic beam-column elements. Plastic hinges were assumed to be located at the faces of beam-column joints. Their behavior was replicated by employing zero-length elements at hinge sites and the elasto-plastic moment-rotation hysteretic model, specifically utilizing the Ibarra Krawinkler Deterioration Model.[19] The modeling parameters were selected based on the flexural capacities of the beams and columns, as determined by Fardis and Haselton. Additionally, the loss in flexural stiffness following cracking was taken into account by utilizing stiffness modifiers for the elastic sections of the beam and column elements, as specified in Table 10-5 of ASCE 41-13.[18, 29, 50].

To accurately simulate the structural response up to collapse, analysis models must capture the full range of behavior, from flexural yielding to the subsequent reduction in strength and stiffness after the peak is reached (Ibarra et al., 2005; Haselton and Deierlein, 2007). Post-peak softening in RC beam-columns dominated by flexure typically results from physical phenomena such as concrete crushing, reinforcing bar buckling and fracture, or bond failure. The complex interaction of various geometric and material behaviors makes it difficult to replicate this cyclic deterioration using fiber-type beam-column models.[51]

Component response is often characterized by two commonly used curves: a monotonic loading curve and a cyclic envelope curve. The differences between these curves are illustrated by the RC column test data shown in Figure 30. The monotonic loading (backbone) curve can be obtained exclusively and directly from a monotonic test. In contrast, the cyclic envelope curve is derived from cyclic test data by connecting the maximum load points at each increment of deformation (PEER/ATC 2010). Unlike the monotonic curve, the cyclic envelope is not unique and is influenced by the specific loading protocol used during testing. The cyclic envelope is frequently used to describe the force-displacement relationship, as specified in ASCE/SEI 41, and it accounts for both in-cycle and cyclic strength degradation.[51]

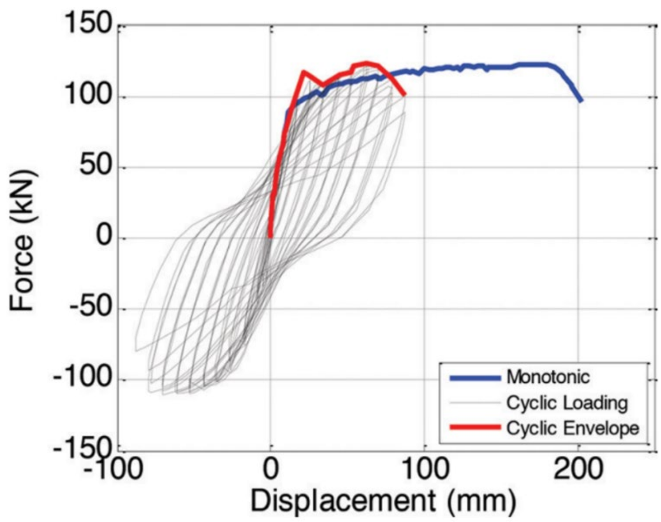


Figure 30: Experimental data from cyclic and monotonic tests of two identical RC columns illustrating definitions of monotonic loading curve and cyclic envelope curve[51]

The monotonic loading curve, depicted in Figure 31, is represented by a trilinear approximation of the relationship between moment and chord rotation. This approximation is characterized by five parameters: My, Ke, Mc/My, θcap,pl, and θpc. Based on these characteristics, it is possible to calculate the yield rotation θy, the inelastic strain-hardening Ksh, and the post-peak strain-softening (Kss) stiffness. It is crucial to account for the decrease in stiffness due to strain in order to accurately represent the weakening of strength during the cycle. The residual strength at significant rotations is presumed to be insignificant and is not measured due to inadequate test data. The lower dashed curve in Figure 31 depicts a cyclic envelope.

This envelope is formed by a curve that surrounds the response curves of a member subjected to cyclic loading demonstrates that cyclic loading has an impact on the ability to rotate without permanent deformation, the maximum strength achieved, and the stiffness during both the hardening and softening phases.[51]

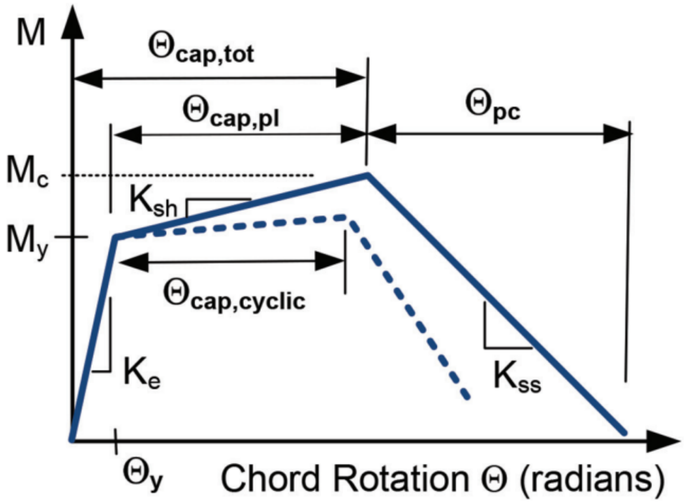


Figure 31: Idealized trilinear end moment versus chord rotation response of equivale[51]

Based on the formulations and equations proposed by Fardis and Haselton In this study, I computed the parameter for simulating the behavior of concrete elements. I have included the calculations in the appendix B and appendix C, which can be readily utilized in Python.

3.8 Implementation of OpenSeespy

This study utilizes The Open System for Earthquake Engineering Simulation (OpenSees), a software framework designed to model the dynamic response of structural and geotechnical systems. OpenSees was originally developed as a computational platform for conducting research in performance-based seismic engineering at the Pacific Earthquake Engineering Research Center. Today, finite element analysis using OpenSees is widely applied to study the behavior of structures under various natural hazards, including fire, wind, earthquakes, and wave actions resulting from tsunamis or storm surges.[38]

In this section, I elucidate the composition and characteristics of the materials employed in this investigation. I utilize the studies conducted by Harden, Boulanger, Gajan, and Raychowdhury to incorporate the features of the Winkler spring into my modeling of soil and soil structure interaction, as well as shallow foundation.[8, 36, 39, 52] Furthermore, I employ the model presented by Kolozvari for the purpose of simulating shear walls.[44, 45, 49] and for modeling the behaviors of frame I use the studies of Haselton, Fardis, Ibarra and Lignnos[15, 18, 19, 29, 51, 53]. Moreover I use some reference book like ATC72-1 and FEMA 356[17, 43]

3.8.1 Nonlinear Winkler Foundation Beam Model.

The BNWF model uses a grid of flexible beam elements to represent the foundation and special elements to simulate the interaction between the foundation and the ground. This allows for a more accurate representation of the foundation's behavior. Each of these one-dimensional zero-length springs operates independently, with their nonlinear inelastic properties represented by modified versions of the QzSimple1, PySimple1, and TzSimple1 material models, which were integrated into OpenSees by Boulanger.[39, 52] In table 24 and 25, I provided the properties of soil and foundation which use in this study base on Harden research[52] and designing foundation. In table 26 and 27, I provided the stiffness of foundation at surface base on Fema-356 for clay and sand soil which using in this study.[17]

Table 24: Properties of clay soil and foundation

Story	ɣ(kN/m^3)	G(kN/m^2)	D_f(m)	B/ L(m)	δ	c(kN/m^2)
7	16.2	4,000	1.5	1.8/ 16.8	0	80
10	16.2	4,000	1.5	2.4/ 17.4	0	80
15	16.2	4,000	2	3.4/ 18.4	0	80

Table 25: Properties of sand soil and foundation

Story	γ (kN/m^3)	G(kN/m^2)	D_f(m)	B/ L(m)	δ	c(kN/m^2)
7	16.2	26,000	1.5	1.8/ 16.8	42	0
10	16.2	26,000	1.5	2.4/ 17.4	42	0
15	16.2	26,000	2	3.4/ 18.4	42	0

Table 26: Properties of stiffness of foundation at surface on caly

Story	K_x(kN/m)	K_y(kN/m)	K_z(kN/m)	K_{xx}(kN/rad)	K_{yy}(kN/rad)
7	82,000	99,000	112,000	343,000	6,200,000
10	108,000	535,000	121,000	535,000	7,900,000
15	109,000	126,000	141,000	1,200,000	11,600,000

Table 27: Properties of stiffness of foundation at surface on sand

Story	K_x(kN/m)	K_y(kN/m)	K_z(kN/m)	K_{xx}(kN/rad)	K_{yy}(kN/rad)
7	530,000	641,000	725,000	2,226,000	40,000,000
10	590,000	699,000	784,000	3,475,000	51,000,000
15	704,000	813,000	912,000	7,403,000	75,000,000

3.8.1.1 PySimple1 Material

Base on the OpenSeespy, the command of PySimple1 Material is[38]:

“uniaxialMaterial('PySimple1', matTag, soilType, pult, Y50, Cd, c=0.0)”

This study is based on the definition provided in the OpenSeespy paper, namely in figure 32.[38] The goal is to simulate the horizontal passive earth pressure and stiffness that develop as the foundation slides, compacts, and potentially forms gaps.

matTag (int)	integer tag identifying material
soilType (int)	soilType = 1 Backbone of p-y curve approximates Matlock (1970) soft clay relation. soilType = 2 Backbone of p-y curve approximates API (1993) sand relation.
pult (float)	Ultimate capacity of the p-y material. Note that “p” or “pult” are distributed loads [force per length of pile] in common design equations, but are both loads for this uniaxialMaterial [i.e., distributed load times the tributary length of the pile].
Y50 (float)	Displacement at which 50% of pult is mobilized in monotonic loading.
Cd (float)	Variable that sets the drag resistance within a fully-mobilized gap as Cd*pult.
c (float)	The viscous damping term (dashpot) on the far-field (elastic) component of the displacement rate (velocity). (optional Default = 0.0). Nonzero c values are used to represent radiation damping effects

Figure 32: Properties of PySimple1 material[38]

The P_{ult} base on the study of Raychowdhury and Harden[8, 52] is

$$p_{ult} = 0.5\gamma K_p D_f^{\wedge 2} \quad (3.7)$$

which Kp is the passive pressure is 1 for caly soil and 11 for sand and it depends on internal fraction of soil. These figures derived from Harden studies[52]. D_f is the height of embedment of foundation.

$$Y_{50} = \frac{0.542 \times Pult}{Kx} \text{ for clay} \quad (3.8)$$

And

$$Y_{50} = \frac{8 \times Pult}{Kx} \text{ for sand} \quad (3.9)$$

Kx is the stiffness of foundation at surface base on the table 26 and 27. Cd and C are assumed 0.05 base on Harden studies.[52]

I prepared the result of the calculation of these parameters for 7, 10, 15 stories for clay and sand soil in tables 28 and 29. In addition, in Appendix 1. I have given the code that is capable of computing these parameters. The Y_{50} value is derived for the long side and short side of the foundation in tables 28 and 29.

Table 28: PySimple1 properties for clay soil

Story	P_{ult}(kN)-Long side	Y_{50}(m)	P_{ult}(kN)-Short side	Y_{50}(m)-short
7	610	0.06	66	0.0053
10	635	0.0559	88	0.0065
15	1,100	0.088	221	0.014

Table 29:PySimple1 properties for sand soil

Story	P_{ult}(kN)-Long side	Y_{50}(m)	P_{ult}(kN)-Short side	Y_{50}(m)-short
7	6,700	0.0068	360	0.0003
10	6,980	0.0063	480	0.00036
15	13,000	0.0099	1,200	0.000798

3.8.1.2 TzSimple1 Material

Base on the OpenSeespy, the command of TzSimple1 Material is[38]

“uniaxialMaterial('TzSimple1', matTag, soilType, tult, z50, c=0.0)”

which their definitions provided by OpenSees document as figure 33[38]

`matTag` (int)	integer tag identifying material
`soilType` (int)	soilType = 1 Backbone of t-z curve approximates Reese and O'Neill (1987). soilType = 2 Backbone of t-z curve approximates Mosher (1984) relation.
`tult` (float)	Ultimate capacity of the t-z material. SEE NOTE 1.
`z50` (float)	Displacement at which 50% of tult is mobilized in monotonic loading.
`c` (float)	The viscous damping term (dashpot) on the far-field (elastic) component of the displacement rate (velocity). (optional Default = 0.0). See NOTE 2.

Figure 33: Properties of TzSimple1 material[38]

I prepared the result of the calculation of these parameters for 7, 10, 15 stories for clay and sand soil in tables 30 and 31. In addition, in Appendix 1. I provided the code which can calculate these parameters base on Harden studies[52]

$$Z_{50}=\frac{0.708\times tult}{Kx} \text{ for clay soil} \tag{3.10}$$

And

$$Z_{50}=\frac{2.05\times tult}{Kx} \text{ for sand soil} \tag{3.11}$$

$$t_{ult} = w_g \tan\delta + cA_f \tag{3.12}$$

Which t_{ult} is explained in formulation 3.6. Kx is the stiffness of foundation base on the Figure 23. The Z_{50} value is derived for the long side and short side of the foundation in tables 30 and 31 for clay and sand soil.

Table 30: Properties of TzSimple1 for clay soil

Story	t_{ult}(kN)-Long side	Z_{50}(m)	t_{ult}(kN)-Short side	Z_{50}(m)-short
7	6,500	0.056	2800	0.02

10	7,500	0.0586	3900	0.0257
15	11,000	0.0712	6000	0.0344

Table 31:Properties of TzSimple1 for sand soil

Story	t_{ult}(kN)-Long side	Z_{50}(m)	t_{ult}(kN)-Short side	Z_{50}(m)-short
7	680	0.00259	680	0.00213
10	1,700	0.0058	1,700	0.0049
15	2,700	0.0074	2,700	0.0064

3.8.1.3 QzSimple1 Material

Base on the OpenSeespy, the command of QzSimple1 Material is[38]

“uniaxialMaterial('QzSimple1', matTag, qzType, qult, Z50, suction=0.0, c=0.0)”

which their definitions provided by OpenSees document as figure 34.[38]

`matTag` (int)	integer tag identifying material
`qzType` (int)	qzType = 1 Backbone of q-z curve approximates Reese and O’Neill’s (1987) relation for drilled shafts in clay. qzType = 2 Backbone of q-z curve approximates Vijayvergiya’s (1977) relation for piles in sand.
`qult` (float)	Ultimate capacity of the q-z material. SEE NOTE 1.
`Z50` (float)	Displacement at which 50% of qult is mobilized in monotonic loading. SEE NOTE 2.
`suction` (float)	Uplift resistance is equal to suction*qult. Default = 0.0. The value of suction must be 0.0 to 0.1.
`c` (float)	The viscous damping term (dashpot) on the far-field (elastic) component of the displacement rate (velocity). Default = 0.0. Nonzero c values are used to represent radiation damping effects.*

Figure 34: Properties of QzSimple1 material[38]

Base on formulations which provided by Harden[52]

$$Z_{50}=\frac{0.525\times qult}{Kz} \quad \text{for clay soil} \tag{3.13}$$

$$Z_{50}=\frac{1.39\times qult}{Kz} \quad \text{for sand} \tag{3.14}$$

$$q_{ult} = cN_cF_{cs}F_{cd}F_{ci} + \gamma D_fN_qF_{qs}F_{qd}F_{qi} + 0.5\gamma BN_\gamma F_{\gamma s}F_{\gamma d}F_{\gamma i} \tag{3.15}$$

I provided the properties for QzSimple1 Material for clay and sand soil in table 32 and

33 respectively.

Table 32: Properties of QzSimple1 material for clay

Story	q_{ult}(kN/m^2)	Z_{50}(m)	suction
7	15,000	0.015	0
10	21,000	0.021	0
15	32,000	0.032	0

Table 33: Properties of QzSimple1 material for sand

Story	q_{ult}(kN/m^2)	Z_{50} (m)	suction
7	156,000	0.062	0
10	245,000	0.16	0
15	515,000	0.20	0

3.8.2 Shear Wall Modeling in OpenSeespy

To model a shear wall with flexure-controlled base, I utilize the MVLEM_3D element in OpenSees, as it aligns with the definition provided in ATC-72-1.[43] This command is sourced from the OpenSees documentation.[38]

“element('MVLEM_3D', eleTag, *eleNodes, m, '-thick', *thick, '-width', *widths, '-rho',*rho, '-matConcrete', *matConcreteTags, '-matSteel', *matSteelTags, ' matShear', matShearTag, <'-CoR', c>, <'-ThickMod', tMod>, <'-Poisson', Nu>, <'-Density', Dens>)” and these parameters are defined in the figure 35 base on document of OpenSeespy.[38]

`eleTag` (int)	unique element object tag
`eleNodes` (list (int))	a list of four element nodes defined in the counter-clockwise direction
`m` (int)	number of element uniaxial fibers
`thick` (list (float))	a list of `m` macro-fiber thicknesses
`widths` (list (float))	a list of `m` macro-fiber widths
`rho` (list (float))	a list of m reinforcing ratios corresponding to macro-fibers; for each fiber: $rho_i = A_{s,i}/A_{gross,i}(1 < i < m)$
`matConcreteTags` (list (int))	a list of `m` uniaxialMaterial tags for concrete
`matSteelTags` (list (int))	a list of `m` uniaxialMaterial tags for steel
`matShearTag` (int)	tag of uniaxialMaterial for shear material
`c` (float)	location of center of rotation from the base (optional; default = 0.4 (recommended))
`tMod` (float)	thickness multiplier (optional; default = 0.63 equivalent to 0.25Ig for out-of-plane bending)
`Nu` (float)	Poisson ratio for out-of-plane bending (optional; default = 0.25)
`Dens` (float)	density (optional; default = 0.0)

Figure 35: Definition of Parameters for MVLEM_3D[38]

There are a total of 5 uniaxial fibers. Each shear wall is equipped with five macro fibers, each having a width of 1 meter. The ratio of reinforcing for each fiber to the thickness of the shear wall is determined using tables 10, 11, and 12. ConcreteCM was selected in OpenSees as the material that exemplifies concrete behavior. Below is the command basis for the OpenSees paper.[38]

“uniaxialMaterial('ConcreteCM', matTag, fpcc, epcc, Ec, rc, xcrn, ft, et, rt, xcrp, mon , '-GapClose', GapClose=0)” and the definition of parameters of this material are illustrated in figure 36.[38]

`matTag` (int)	integer tag identifying material
`fpcc` (float)	Compressive strength (f'_c)
`epcc` (float)	Strain at compressive strength (ϵ'_c)
`Ec` (float)	Initial tangent modulus (E_c)
`rc` (float)	Shape parameter in Tsai's equation defined for compression (r_c)
`xcrn` (float)	Non-dimensional critical strain on compression envelope (ϵ^-_{cr}, where the envelope curve starts following a straight line)
`ft` (float)	Tensile strength (f_t)
`et` (float)	Strain at tensile strength (ϵ_t)
`rt` (float)	Shape parameter in Tsai's equation defined for tension (r_t)
`xcrp` (float)	Non-dimensional critical strain on tension envelope (ϵ^+_{cr}, where the envelope curve starts following a straight line - large value [e.g., 10000] recommended when tension stiffening is considered)
`mon`	optional, monotonic stress-strain relationship only: mon=1 (invoked in FSAM only), mon=0 (no impact since monotonic)
`'-GapClose'` (str)	optional, denote next parameter is `GapClose`
`GapClose` (float)	optional, GapClose = 0, less gradual gap closure (default); GapClose = 1, more gradual gap closure

Figure 36: Properties of concreteCM material[38]

Based on whether the concrete is located in a boundary condition part or not, it can be classified as confined or unconfined. I refer to the properties of confined and unconfined concrete, including properties like compressive strength, strain at compressive strength, initial tangent modulus, tensile strength, and strain at tensile strength, as specified in Table 34. These properties are based on the Kolozvari and Orakcal studies.[48, 49]

Table 34: Properties of concreteCM

Type Con.	fc(kN/m^2)	epcc	Ec(MN/m^2)	ft(kN/m^2)	et
Confined	53,000	0.005	34,500	1,800	0.00008
Unconfined	34,500	0.002	27,800	1,800	0.00008

And for modeling steel rebar, I use SteelMPF in OpenSees, which depicts the renowned uniaxial nonlinear hysteretic material model for steel introduced by Menegotto and Pinto [54] and extended by Filippou[55]. This command represent this material in OpenSeespy[38]

“uniaxialMaterial('SteelMPF', matTag, fyp, fyn, E0, bp, bn, *params, a1=0.0, a2=1.0, a3=0.0, a4=1.0)” and in figure 37, the OpenSees document explain these variables.[38]

`matTag` (int)	integer tag identifying material
`fyp` (float)	Yield strength in tension (positive loading direction)
`fyn` (float)	Yield strength in compression (negative loading direction)
`E0` (float)	Initial tangent modulus
`bp` (float)	Strain hardening ratio in tension (positive loading direction)
`bn` (float)	Strain hardening ratio in compression (negative loading direction)
`params` (list (float))	parameters to control the transition from elastic to plastic branches. `params=[R0,cR1,cR2]`. Recommended values: `R0=20`, `cR1=0.925`, `cR2=0.15` or `cR2=0.0015`
`a1` (float)	Isotropic hardening in compression parameter (optional, default = 0.0). Shifts compression yield envelope by a proportion of compressive yield strength after a maximum plastic tensile strain of a2(fyp/E0)
`a2` (float)	Isotropic hardening in compression parameter (optional, default = 1.0).
`a3` (float)	Isotropic hardening in tension parameter (optional, default = 0.0). Shifts tension yield envelope by a proportion of tensile yield strength after a maximum plastic compressive strain of a3(fyn/E0).
`a4` (float)	Isotropic hardening in tension parameter (optional, default = 1.0). See explanation of a3.

Figure 37: Properties of steelMPF[38]

The properties utilized in this study, such as yield strength in tension and compression, initial tangent modulus, and strain hardening ratio in tension and compression, were presented in table 35.

Table 35: Features of steelMPF

Steel Bar	fyp(kN/m^2)	fyn(kN/m^2)	E0(MN/m^2)	bp	bn
SteelMPF	413000	413000	200000	0.02	0.02

The shear stiffness, which represents the connection between shear force and deformation in a linear elastic system, is a crucial modeling parameter that can significantly impact the projected response of a wall in analytical models. PEER/ATC 72 proposes a secant-to-yield shear modulus of 0.05G for walls that have a maximum shear stress capacity less than or equal to 5√f'c, and 0.1G for walls having a shear stress capacity between 5 and 10√f'c. For this work, I utilized a gravitational force of 0.1 G.[43, 48]

3.8.3 Modeling of Hinges

In this research, I use Modified Ibarra-Medina-Krawinkler material for modeling hinges in OpenSeespy. The command is based on document of OpenSees is[38]

"uniaxialMaterial('ModIMKPeakOriented', matTag, K0, as_Plus, as_Neg, My_Plus, My_Neg, Lamda_S, Lamda_C, Lamda_A, Lamda_K, c_S, c_C, c_A, c_K, theta_p_ Plus, theta_p_Neg, theta_pc_Plus, theta_pc_Neg, Res_Pos, Res_Neg, theta_u_Plus, t heta_u_Neg, D_Plus, D_Neg)" and this command is utilized to create a ModIMKPeakOriented material. This material emulates the modified Ibarra-Medina-Krawinkler degradation model with a hysteresis response focused on the peak. Be aware that the hysteresis behavior of this material has been adjusted based on 200 experimental data points from RC beams. This adjustment was done to accurately determine the degradation parameters of the model.[15, 38] In figure 38 and 39 the explanation of variables are provided[38]

`matTag` (int)	integer tag identifying material
`K0` (float)	elastic stiffness
`as_Plus` (float)	strain hardening ratio for positive loading direction
`as_Neg` (float)	strain hardening ratio for negative loading direction
`My_Plus` (float)	effective yield strength for positive loading direction
`My_Neg` (float)	effective yield strength for negative loading direction (negative value)
`Lamda_S` (float)	Cyclic deterioration parameter for strength deterioration [E_t=Lamda_S*M_y, see Lignos and Krawinkler (2011); set Lamda_S = 0 to disable this mode of deterioration]
`Lamda_C` (float)	Cyclic deterioration parameter for post-capping strength deterioration [E_t=Lamda_C*M_y, see Lignos and Krawinkler (2011); set Lamda_C = 0 to disable this mode of deterioration]
`Lamda_A` (float)	Cyclic deterioration parameter for accelerated reloading stiffness deterioration [E_t=Lamda_A*M_y, see Lignos and Krawinkler (2011); set Lamda_A = 0 to disable this mode of deterioration]
`Lamda_K` (float)	Cyclic deterioration parameter for unloading stiffness deterioration [E_t=Lamda_K*M_y, see Lignos and Krawinkler (2011); set Lamda_K = 0 to disable this mode of deterioration]

Figure 38: Properties of ModIMKPeakOriented material[38]

In Appendix 1, I provide the code which calculate these properties automatically just with the properties of beam and column. For calculating My, I use Fardis study[29, 56] and for plastic hinges and modeling deterioration I use the equations which proposed by Haselton[51] which I provide the code that calculate these parameters in python environment. I explained the formulations which I use to calculate these variables.

c_S (float)	rate of strength deterioration. The default value is 1.0.
c_C (float)	rate of post-capping strength deterioration. The default value is 1.0.
c_A (float)	rate of accelerated reloading deterioration. The default value is 1.0.
c_K (float)	rate of unloading stiffness deterioration. The default value is 1.0.
theta_p_Plus (float)	pre-capping rotation for positive loading direction (often noted as plastic rotation capacity)
theta_p_Neg (float)	pre-capping rotation for negative loading direction (often noted as plastic rotation capacity) (must be defined as a positive value)
theta_pc_Plus (float)	post-capping rotation for positive loading direction
theta_pc_Neg (float)	post-capping rotation for negative loading direction (must be defined as a positive value)
Res_Pos (float)	residual strength ratio for positive loading direction
Res_Neg (float)	residual strength ratio for negative loading direction (must be defined as a positive value)
theta_u_Plus (float)	ultimate rotation capacity for positive loading direction
theta_u_Neg (float)	ultimate rotation capacity for negative loading direction (must be defined as a positive value)
D_Plus (float)	rate of cyclic deterioration in the positive loading direction (this parameter is used to create assymetric hysteretic behavior for the case of a composite beam). For symmetric hysteretic response use 1.0.
D_Neg (float)	rate of cyclic deterioration in the negative loading direction (this parameter is used to create assymetric hysteretic behavior for the case of a composite beam). For symmetric hysteretic response use 1.0.

Figure 39: Properties of ModIMKPeakOriented material[38]

Mathematical expressions that define the point at which a material transitions from elastic to plastic behavior under a uniaxial moment, My, and curvature, φy, have been provided for sections with rectangular compression zone, allowing the avoidance of a comprehensive moment-curvature analysis. These expressions are derived from plane section analysis and linear material σ-ε laws. The section's effective depth is designated as d, while the width is denoted as b. The tension and compression reinforcement areas are represented by As1 and As2, respectively. These areas are normalized by multiplying them with the product of b and d, resulting in the ratios ρ1 and ρ2. The reinforcement is spread nearly evenly between the tension and compression bars. The curve of the tension bars at the point of yielding is:[29, 56]

$$\Phi y = \frac{f_{y1}}{E_s(1-\varepsilon_y)d} \tag{3.16}$$

where fy1 represents the The stress level at which the tension reinforcement begins to deform inelastically and ξy denotes the depth of the neutral axis at yielding, normalized to d.

$$\varepsilon_y = (n^2A^2 + 2nB)^{0.5} \tag{3.17}$$

where n represents the ratio of elastic moduli(n=Es/Ec), and A and B are given by:

$$A = \rho 1 + \rho 2 + \frac{N}{bdf_{y1}} \tag{3.18}$$

$$B = \rho 1 + \rho 2\delta' + \frac{N}{bdf_{y1}} \tag{3.19}$$

The symbol δ' represents the ratio of d' to d, where d' refers to the separation between the middle of the compression reinforcement and the extreme compression fibers. The axial force, N, is considered positive when it is in compression. A segment with a high axial load ratio, N/Acfc, may appear to yield due to the substantial nonlinearity of the compressed concrete before the tension steel reaches its yield point. The concept can be simplified by defining it as the point at which a particular strain is exceeded at the extreme compression fibers, assuming both steel and concrete are elastic up to that point.

Then apparent yielding of the member takes place at curvature:

$$\Phi y = \frac{1.8f_c}{E_c\,\varepsilon_y d} \tag{3.20}$$

The yield curvature is determined by selecting the smaller value between the two values obtained from equations 3-16 and 3-20.

Equilibrium gives the yield moment:

$$\frac{M_y}{bd^3} = \Phi y \left\{ E_c \left[\frac{\varepsilon_y{}^2}{2}\left(\frac{1+\delta\prime}{2} - \frac{\varepsilon_y}{3}\right)\right] + \frac{E_s(1-\delta\prime)}{2}\left[(1-\varepsilon_y)\rho 1 + (\varepsilon_y - \delta')\rho 2\right]\right\} \quad (3.21)$$

To calculate the parameters depicted in figure 29, refer to the formulae provided by Haselton. Plastic rotation capacity ($\theta_{cap,pl}$) refers to the maximum rotation a material or structure can endure before failing or undergoing permanent deformation. Unlike stiffness or strength, which can be calculated through mechanics-based models, the formula for calculating deformation limits, $\theta_{cap,pl}$, relies mostly on empirical evidence. After assessing possible correlations with probable design variables, the subsequent equation is suggested for calculating the rotation capacity (expressed in radians) between the yield and the maximum moment resistance.[51]

$$\theta_{cap,pl} = 0.1(1 + 0.55a_{sl})(0.16)^{\mathrm{v}}\,(0.02 + 40\rho_{sh})^{0.43}(0.54)^{0.01c_{unit}f_c} \quad (3.22)$$

The coefficient "a_{sl}" is normally set to 1, although it can also be set to 0. Its value depends on whether bond-slip is included. v represents the axial load ratio, which is calculated by dividing the load P by the product of $A_g f_c'$. ρ_{sh} is the ratio of transverse reinforcement, which is obtained by dividing Ash by sb. cunits is a conversion ratio used when converting the value of fc′ from MPa to ksi (1.0 for MPa and 6.9 for ksi). ρ_{sh} is the transverse reinforcement ratio A_{sh}/sb. The post-capping rotation capacity (θ_{pc}) is determined by several key parameters that have the greatest impact on deformation capacity. These parameters include the axial load ratio, transverse steel ratio, reinforcing bar buckling coefficient, stirrup spacing, concrete strength, and longitudinal steel ratio.

$$\theta_{pc} = (0.76)(0.031)^{\mathrm{v}}(0.02 + 40\rho_{sh})^{1.02} \leq 0.1 \quad (3.23)$$

And assumed:

$$\frac{M_c}{M_y} = 1.13 \quad (3.24)$$

Chapter 4

RESULTS

4.1 Introduction

This chapter presents the results of analyzing structures subjected to earthquakes under two conditions: one where the structures are fixed to the ground and another where soil-structure interaction is considered. The subjects which are studied, including the reaction in foundation, acceleration in stories, drifts, the behaviors of shear force and its components like steel bars, concrete, curvature, shear force and deformation and the moment-rotation response of beams and columns in both condition and I compare the response of structure in two conditions with fixed foundation and when included SSI analysis.

4.2 Structural Behavior on Clayey Soil With and Without SSI

In this section, the analysis results for 7-, 10-, and 15-story buildings constructed on clay soil and subjected to the ground motions of Darfield, Chi Chi, and El Mayor are presented. Table 36 displays the first-period values under two scenarios: one with a fixed foundation assumption and the other with an SSI analysis.

Table 36: First period for fixed foundation and SSI analysis

Story	T1(s)-Fixed	T1(s)-SSI
7	0.652	0.906
10	0.884	1.192
15	1.22	1.886

The first natural period is shorter (lower T1) for the fixed foundation scenario compared to the SSI scenario. This difference occurs because, in reality, the interaction between the building and the soil tends to make the structure more flexible, leading to longer natural period. The increased flexibility in the SSI scenario is due to the compliance (deformability) of the soil, which adds to the overall deformability of the structure.

The pick of drifts of stories for the 7, 10 and 15-story buildings affected by the earthquake are shown in Table 37, 38 and 39. In certain stories, the maximum drift is reduced, while in others, it is increased.

Table 37: Maximum drift of 7 stories on clay soil (Type D) subjected to Darfield ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	8.57	0	9.8
1	10.1	5.43	4.68	4.8
2	16.6	12	7.12	8.1
3	17.5	13.5	8.9	9.2
4	19	16.2	9.3	10.4
5	18.4	14.8	6.3	10.3
6	18	14.1	5	10.7
7	16.4	13.4	6	10.2

Table 38: Maximum drift of 10 stories subjected to El Mayor ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	6.26	0	6.14
1	4.53	18.66	4.49	15.65
2	21.63	8.9	9.21	6.69
3	12.1	10.7	10.3	8.6
4	13.6	11.1	11.6	9.4
5	11.4	10.9	11	10
6	14.6	10.4	11.4	10
7	12.2	11.3	9.9	9.3
8	11	9.8	8.5	9.4

9	8	9.3	8	5.7
10	10	6.2	6.9	8.5

Table 39: Maximum drift of 15 stories subjected to Chi-Chi ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	1.48	0	2.71
1	5.55	6.25	20.6	23.01
2	10.75	10.97	36	31.9
3	13.8	12.5	37.8	36.2
4	10.6	12.8	37.6	35.6
5	12.6	13	38	35
6	11.7	13.2	36	32
7	144.3	12.1	38	36
8	12.3	13.6	26	28
9	11.4	14.1	12	20
10	11	12	17	22
11	9	7	8	15
12	7	11	4	9
13	6	9	3	9
14	4	3	0	4
15	4	3	0	4

Figures 40, 41 and 42 illustrate the acceleration (m/s²) over time for the top floor of 7, 10, and 15-story structures subjected to ground motions, under both fixed and SSI conditions.

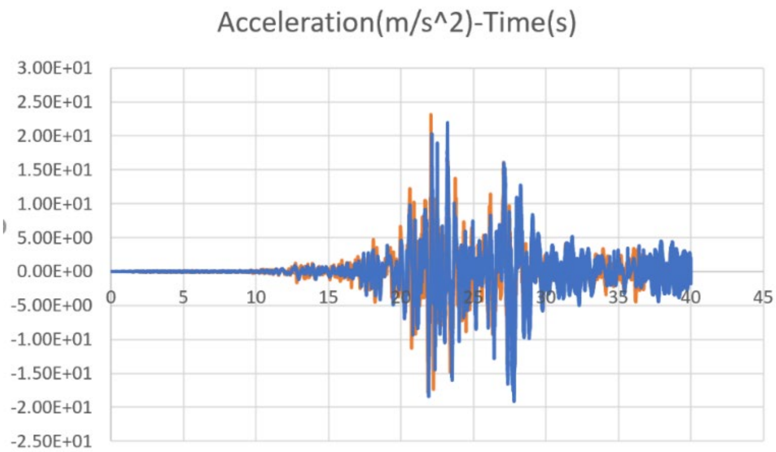


Figure 40: Acceleration at the top floor of a 7-story building under Darfield in the Y axe - blue line for SSI condition, Orange line for fixed foundation Condition

The maximum acceleration sometimes is reduced in the SSI analysis and sometimes is increased. For instance, as illustrated in Figure 40, the maximum acceleration in SSI condition is reduced, whereas in Figures 41 and 42, the maximum acceleration is increased in SSI in comparison to fixed foundation condition.

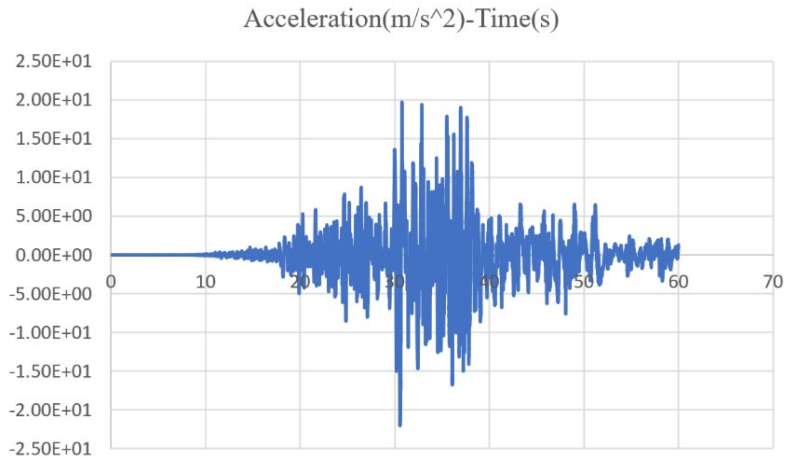


Figure 41: Acceleration response on the tenth floor under El Mayor in the X axe - fixed foundation condition

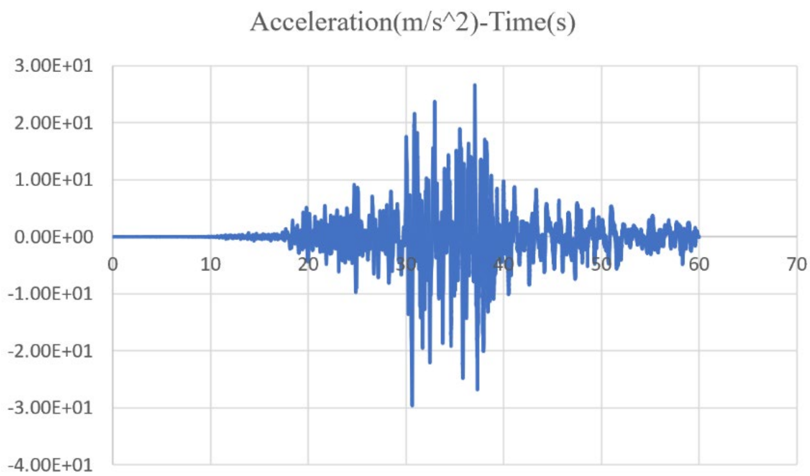


Figure 42: Acceleration response on the tenth floor under the El Mayor earthquake - SSI condition

In order to successfully record the reaction of shear walls and their components to earthquakes, I utilized the MVLEM-3D model, as discussed in Chapter 3. The results are provided as follows. Figures 43 to 46 illustrate the hysteresis behavior of the steel bar in the shear wall on the first and sixth floors, respectively, When the multy story building (7 story) undergoes the Darfield earthquake. From these and other results not provided here for the shear walls in the 7, 10 and 15-story structures, we can conclude the following:

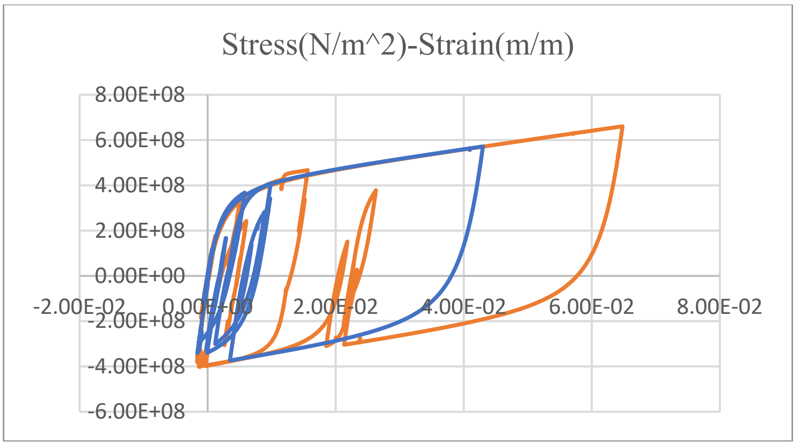


Figure 43: Hysteretic response of steel bar in first-floor shear wall - blue line represents SSI condition, orange line represents fixed foundation condition under Darfield ground motion

The orange line indicates that under fixed conditions, the steel bar exhibits greater energy dissipation during cyclic loading, as shown by the wider hysteresis loops, suggesting higher levels of inelastic deformation. The stress values under SSI conditions are generally lower than those under fixed conditions, indicating that soil interaction helps reduce the peak stresses experienced by the steel bar.

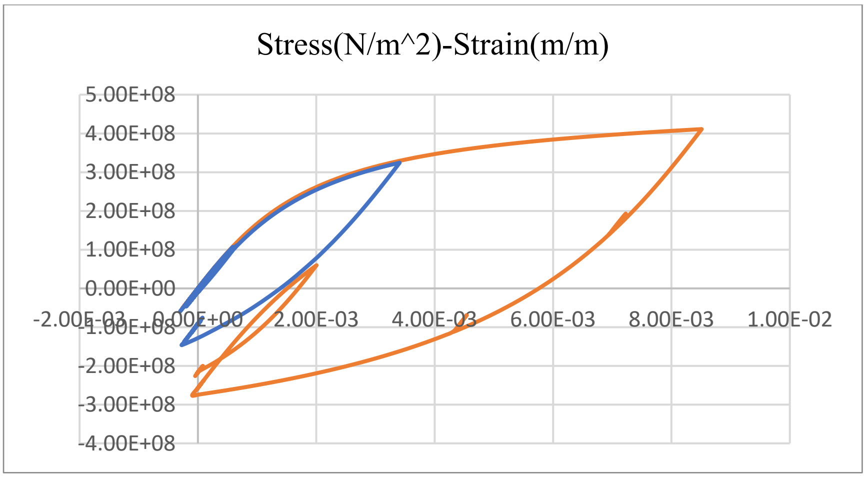


Figure 44: Hysteretic behavior of steel bar in shear wall on tenth floor - blue line for SSI condition, orange line for fixed foundation condition under El Mayor ground motion

The orange line also demonstrates more ductile behavior, with a more gradual decrease in stiffness over cycles, allowing the steel bar to undergo more deformation before failure.

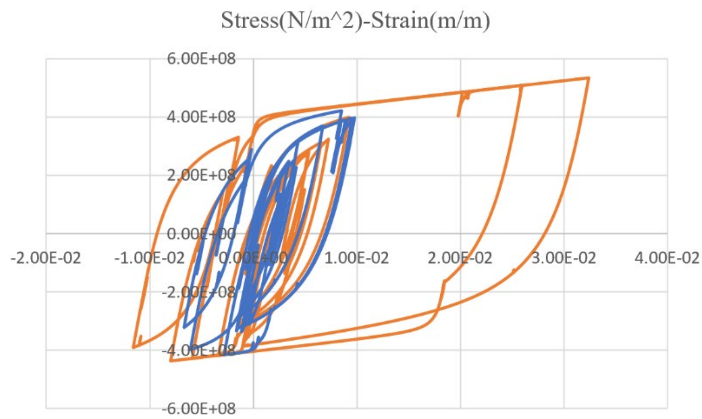


Figure 45: Hysteretic behavior of steel bar in shear wall on first floor of a 10- story- blue line for SSI condition, orange line for fixed foundation condition under El Mayor ground motion

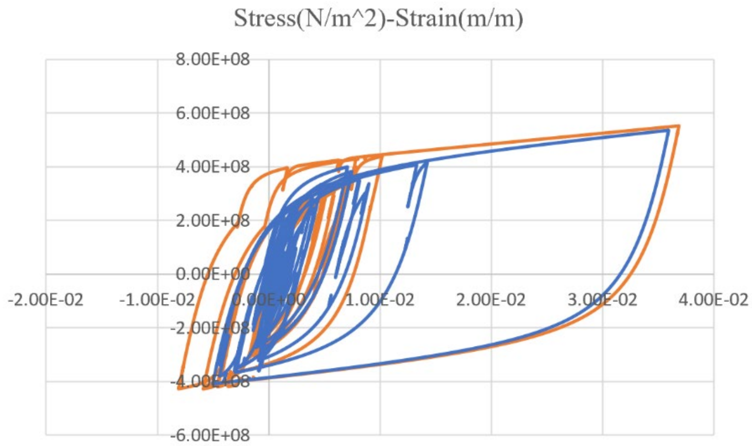


Figure 46: Hysteretic behavior of steel bar in shear wall on first floor of a 15-Story building - blue line for SSI condition, orange line for fixed foundation condition under the Chi-Chi earthquake ground motion.

In contrast, the blue line, representing the SSI condition, shows narrower hysteresis loops, indicating less energy dissipation and lower levels of inelastic deformation. The stress values in the fixed condition are higher, suggesting that the steel bar experiences greater stresses without the mitigating effects of soil interaction. Additionally, the blue line shows more brittle behavior with sharper changes in stiffness, indicating that the steel bar is less able to accommodate deformation before experiencing significant strength degradation.

Figures 47 and 48 presents the hysteresis behavior of unconfined concrete in the shear wall on the sixth floor and tenth floor under SSI conditions.

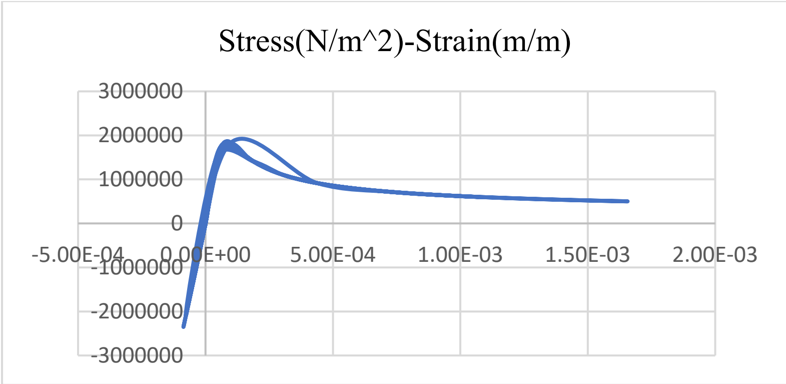


Figure 47: Hysteretic behavior of unconfined concrete in shear wall on sixth floor - SSI condition under Darfield ground motion

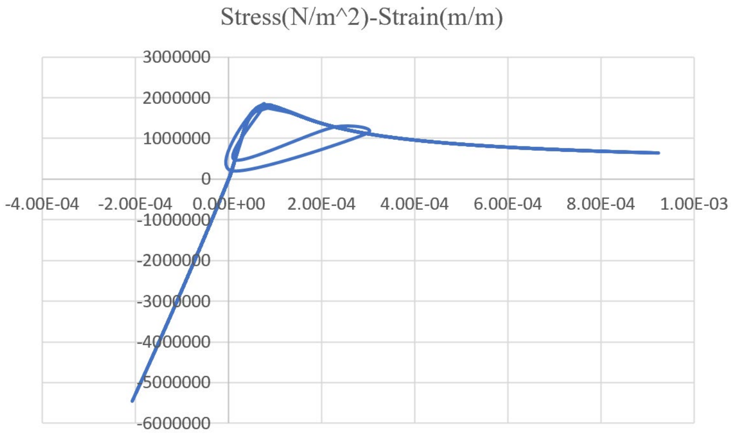


Figure 48: Hysteretic behavior of unconfined concrete in the shear wall on the tenth floor under SSI conditions under El Mayor ground motion

Figures 49 and 50 demonstrate the hysteresis behavior of confined concrete in a shear wall subjected to Chi-Chi ground motion in a 15-story structure, comparing fixed foundation and SSI conditions. From these figures, it is evident that under SSI conditions, the concrete experiences less strain and stress compared to the fixed foundation condition.

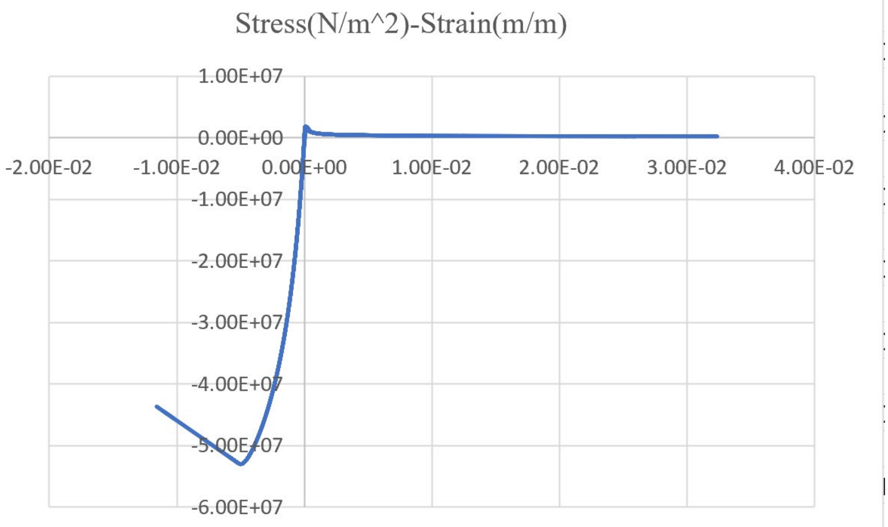


Figure 49: Hysteretic behavior of confined concrete in the shear wall on the first floor under fixed foundation conditions under Chi - Chi ground motion

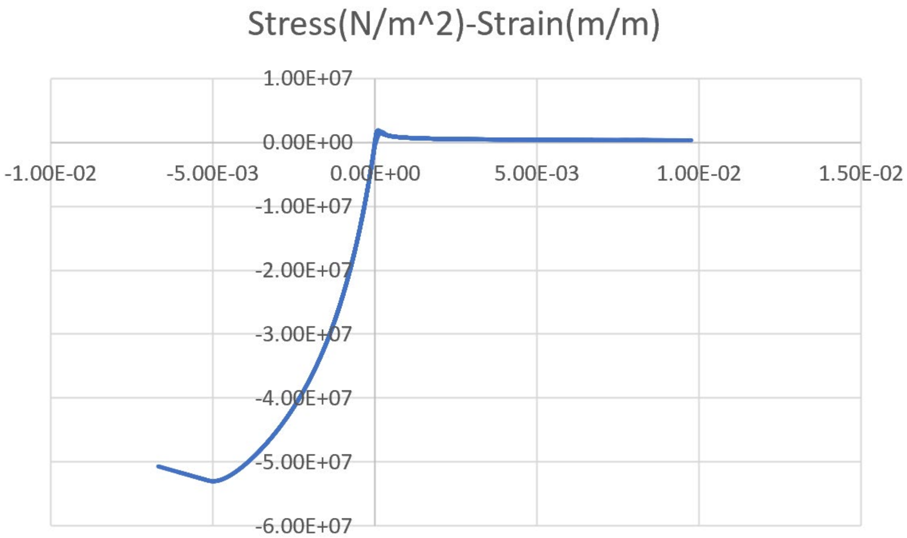


Figure 50: Hysteretic behavior of confined concrete in the shear wall on the first floor under SSI conditions under Chi - Chi

For clarity, a specific area in Figure 51 that shows the hysteretic behavior of concrete has been highlighted. Due to the proximity of the lines, this behavior is difficult to see in the larger image, as shown in the earlier figures.

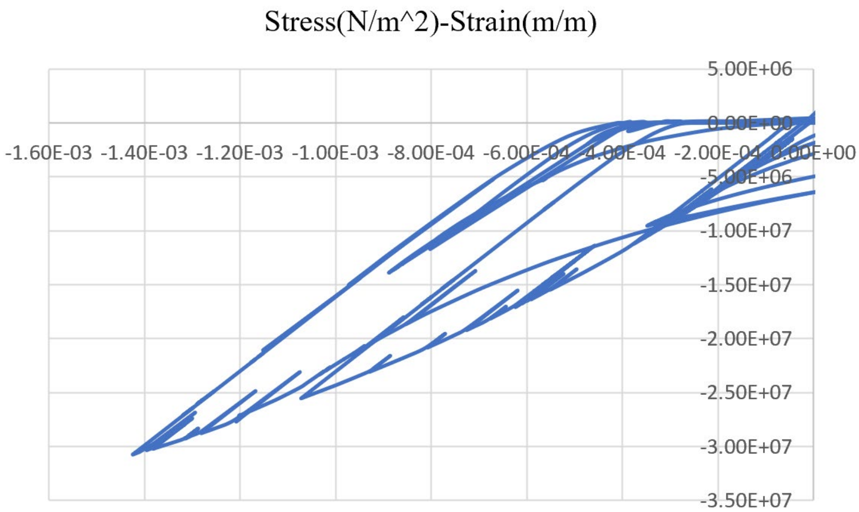


Figure 51: Hysteretic behavior of confined concrete in the shear wall

Figures 52, 53 and 54 illustrates the curvature versus lateral load in the shear wall for both fixed and SSI analyses. The graph shows the lateral load-curvature relationship for a shear wall on the first floor subjected to the Darfield and El Mayor ground motion. The line (blue color one) indicates the SSI condition, whereas the orange one demonstrated the fixed foundation.

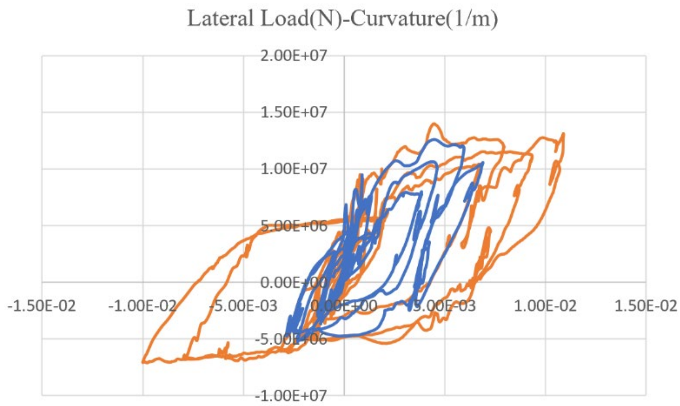


Figure 52: Lateral load-curvature relationship in the shear wall of 7-story on the first floor subjected to Darfield ground motion. The blue curve represents the SSI condition, while the orange curve represents the fixed foundation condition

Curvature in a shear wall refers to the measure of bending deformation experienced by the wall when affected by lateral forces, such as those caused by earthquakes. It is a key indicator of the wall's flexural behavior and is crucial for understanding how the wall responds to such forces. Curvature indicates the extent to which the shear wall bends under lateral loads. Higher curvature values signify greater bending and deformation of the wall. The curvature helps in assessing the structural performance and stability of the wall during seismic events. It provides insights into the wall's ability to withstand lateral forces and maintain its integrity. Curvature is associated with the wall's ability to dissipate energy during cyclic loading (such as during an earthquake). A wall that can bend more (higher curvature) without failing can absorb and dissipate more energy, reducing the overall seismic impact on the building.

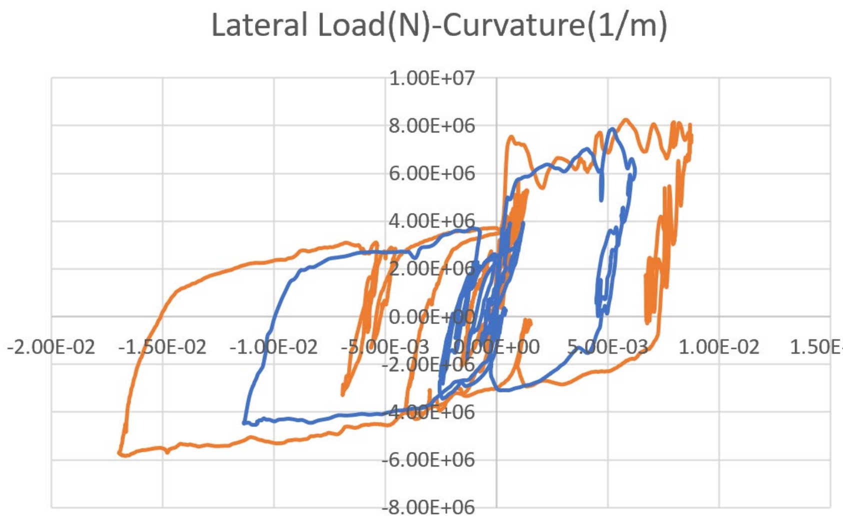


Figure 53: Lateral load-curvature in shear wall of 10-story on the first floor subjected to El Mayor ground motion - blue line for SSI condition, orange line for fixed foundation condition

By comparing the curvature under different foundation conditions (SSI vs. fixed foundation), engineers can assess the influence of SSI on the shear wall's response. The fixed foundation condition typically shows greater curvature, indicating that the wall experiences more deformation due to the rigid nature of the foundation. In contrast, the SSI condition generally results in lower curvature, reflecting a stiffer response with less deformation.

The orange line exhibits wider hysteresis loops, indicating higher energy dissipation during cyclic loading and greater inelastic deformation. This suggests that under fixed conditions, the shear wall can deform more extensively before reaching its ultimate capacity. Additionally, the peak lateral loads are slightly higher, meaning the wall can sustain higher loads before failure.

Conversely, the blue line displays narrower hysteresis loops, indicating lower energy dissipation and less inelastic deformation. The curvature range is more limited, suggesting that the shear wall reaches its failure point with less deformation compared to the fixed condition. Furthermore, the peak lateral loads are slightly lower, implying that the SSI condition restricts the load-carrying of the shear wall.

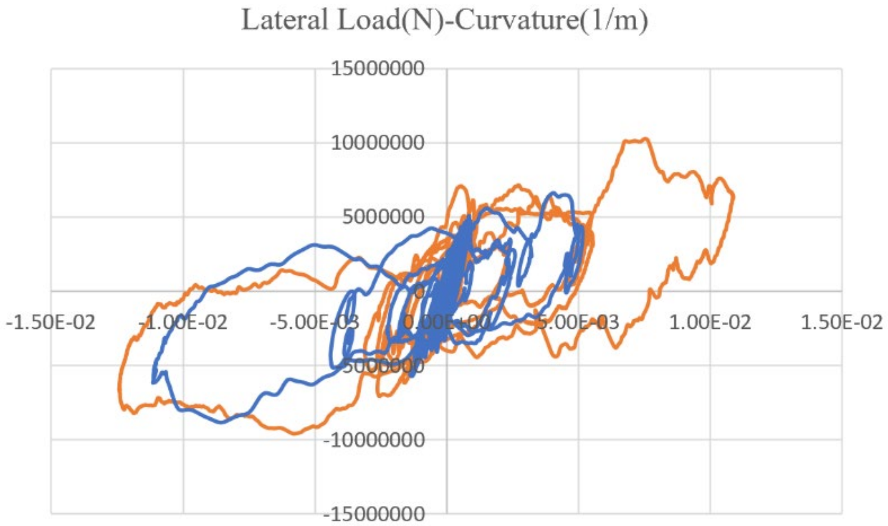


Figure 54: Lateral load-curvature in shear wall of 15-story on the first floor subjected to Chi - Chi ground motion - blue line for SSI condition, orange line for fixed foundation condition

Figures 55 to 61 present the analysis results for columns and beams under fixed and SSI conditions. A zero-length spring was employed to simulate the deterioration strength of the columns and beams in response to earthquakes, as discussed in Chapter 3. The results demonstrate a significant difference between the two conditions.

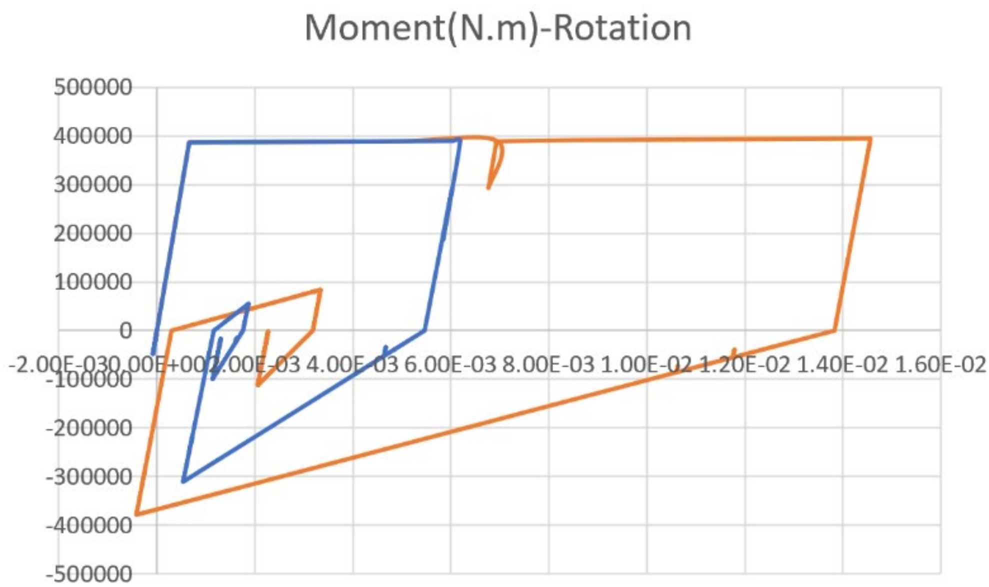


Figure 55: Hysteretic behavior of column (0, 0, 0.2) in 7-story building subjected to Darfield ground motion - blue line for SSI condition, orange line for fixed foundation condition

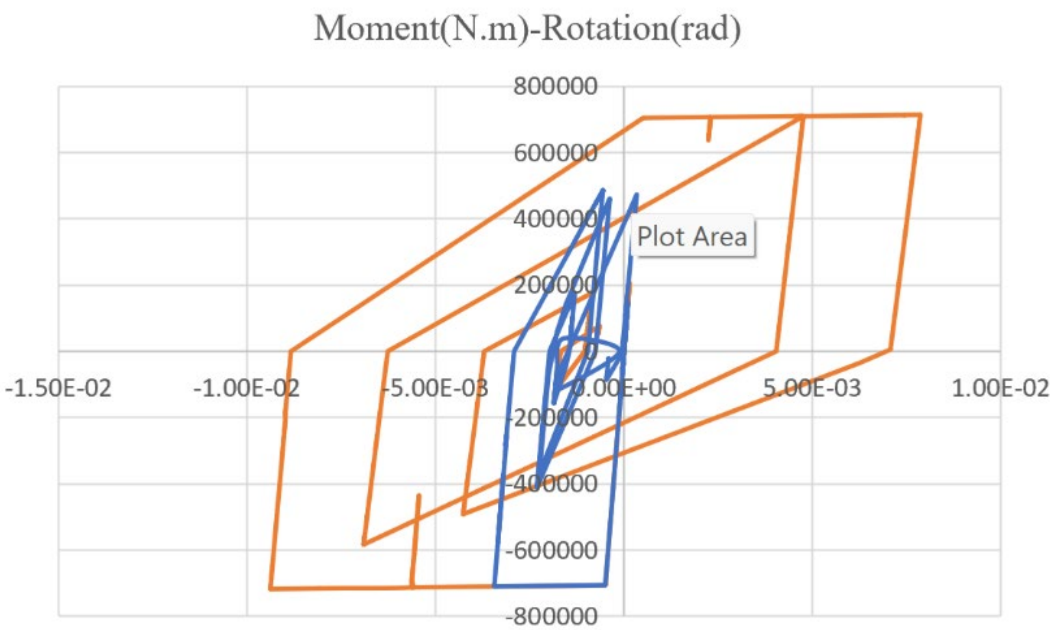


Figure 56: Hystertic behavior of column (0, 0, 0.2) in 10- story- subjected to El Mayor ground motion- blue line for SSI condition, orange line for fixed foundation condition

For both beam and column, blue line shows narrower hysteresis loops, indicating less energy dissipation during cyclic loading. This suggests that the beam and column under SSI conditions experience lower levels of inelastic deformation.

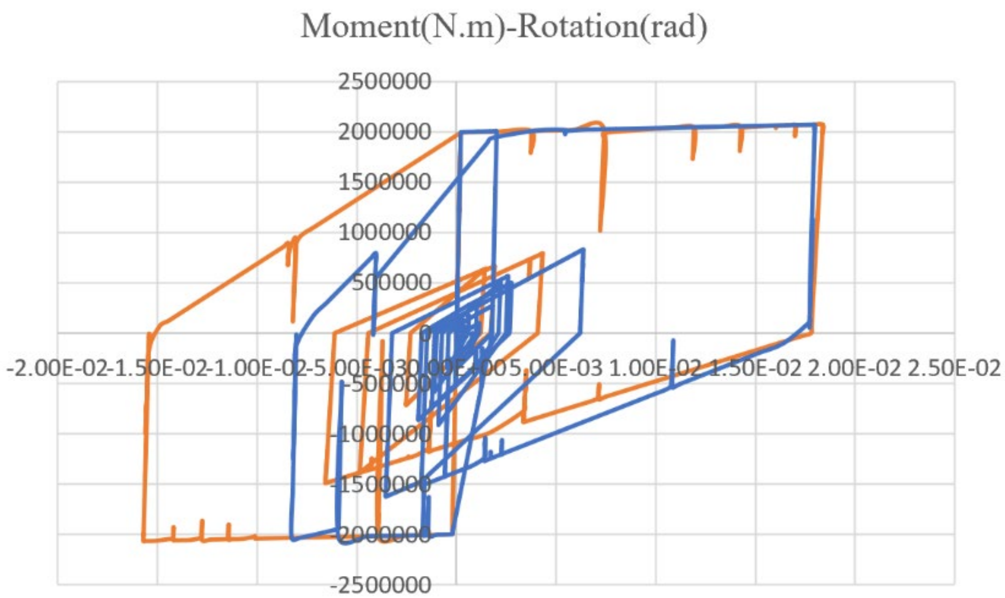


Figure 57: Hystertic behavior of column (0, 0, 0.2) in 15- story- subjected to Chi - Chi ground motion- blue line for SSI condition, orange line for fixed foundation condition

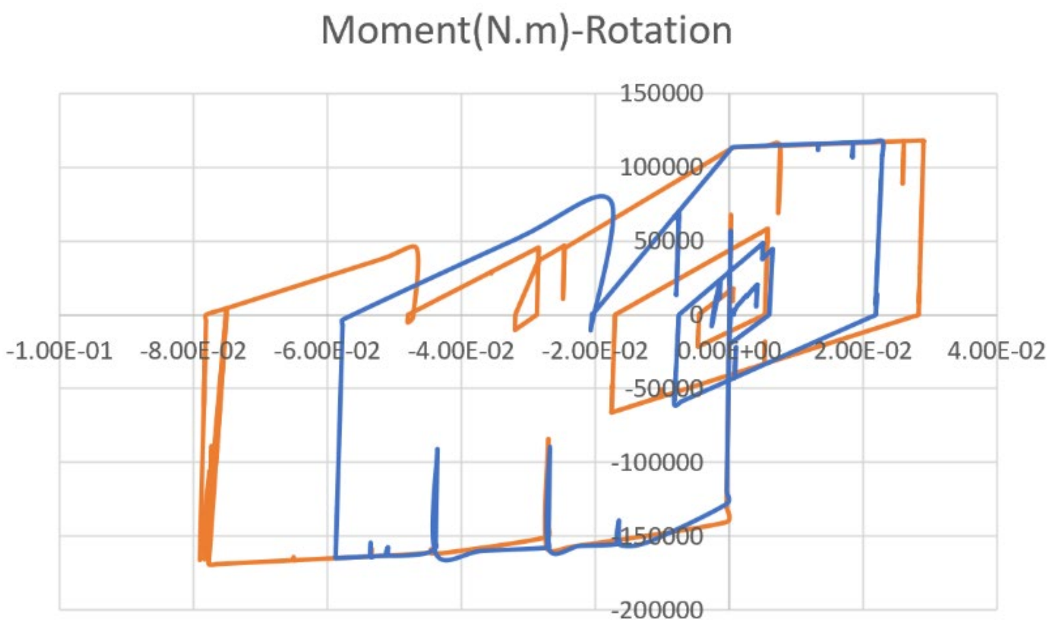


Figure 58: Hysteretic response in beam (0, 0.2, 3) in 7-story building subjected to Darfield ground motion - blue line for SSI condition, orange line for fixed foundation condition

The peak moments (both positive and negative) are lower for the SSI foundation, implying that the beam and column do not develop as high moment capacities as they do under fixed foundation conditions.

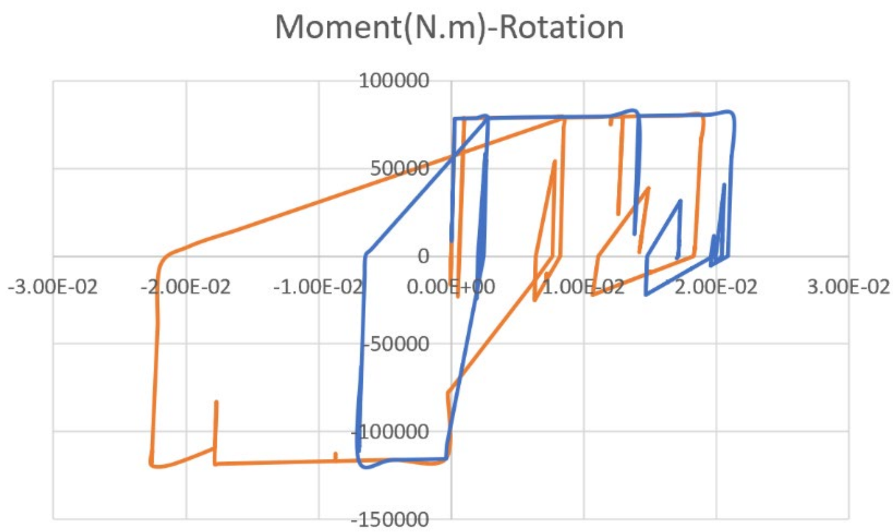


Figure 59: Hysteretic response in beam (10, 4.8, 3) in 7-Story building subjected to Darfield ground motion - blue line for SSI condition, orange line for fixed foundation condition

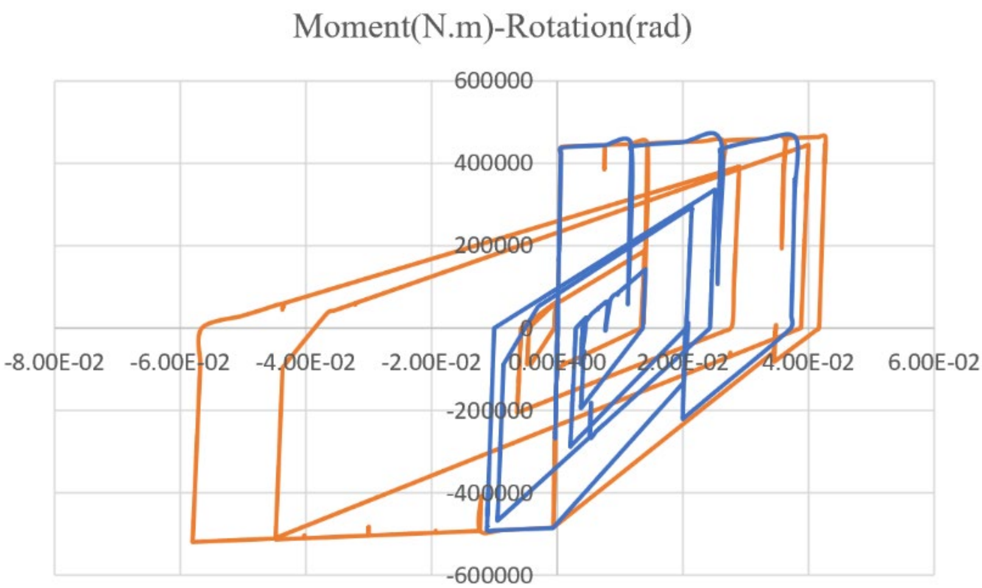


Figure 60: Hysteretic response in beam (0.25, 0, 21) in 10-story building subjected to El Mayor ground motion - blue Line for SSI condition, orange line for fixed foundation condition

The hysteresis loops for the SSI foundation are relatively stable, demonstrating consistent cyclic behavior with less scatter and degradation. Conversely, the orange line shows wider hysteresis loops, indicating more energy dissipation during cyclic loading.

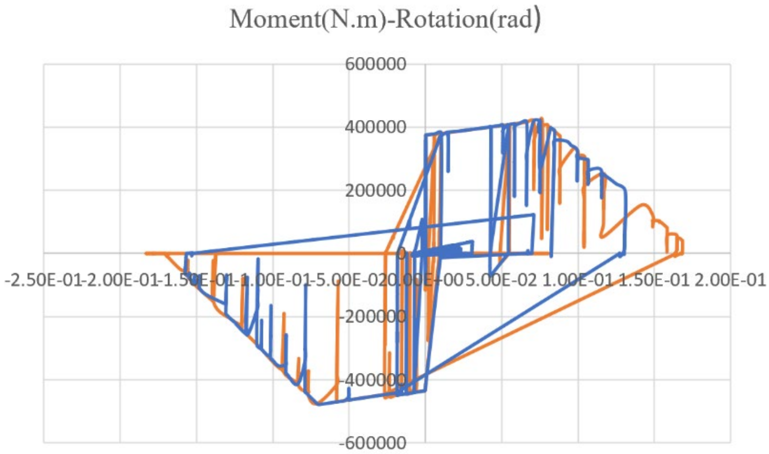


Figure 61: Hysteretic response in beam (0.4, 0, 3) in 15-story building subjected to Chi - Chi ground motion - blue Line for SSI condition, orange line for fixed foundation condition

This suggests that the beam and column under fixed conditions experiences higher levels of inelastic deformation. The peak moments (both positive and negative) are higher for the fixed foundation condition.

To capture the lateral displacement and behavior of soil, I used PySimple1 and TzSimple1, as explained in Chapter 3. Figures 62 to 67 illustrate the load versus lateral displacement behavior of clay modeled with PySimple1 and TzSimple1, as subjected to the Darfield, El Mayor, and Chi-Chi ground motions.

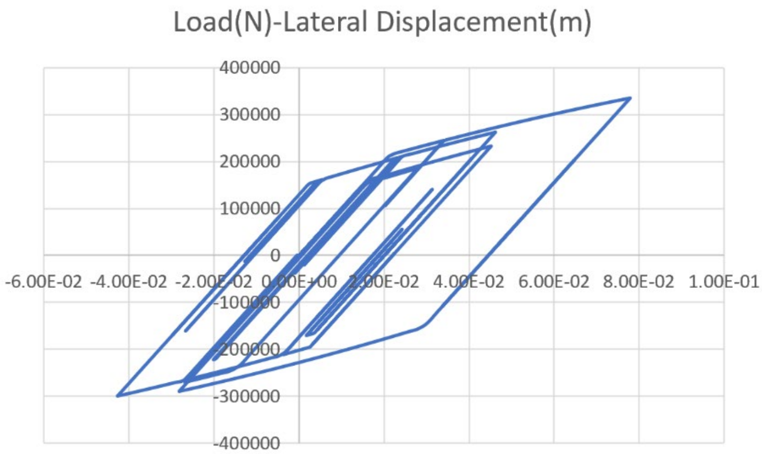


Figure 62: Seismic response of clay (PySimple1 material) beneath a 7-story subjected to Darfield ground motion

The plots depict multiple hysteresis loops, which signify the relationship between lateral displacement and load during cyclic loading and unloading. The nonlinear nature of these plots shows that the clay does not behave purely elastically but undergoes plastic deformation, especially under high stress levels, typical for materials like clay during seismic events.

The area within each loop represents the energy dissipated through the hysteresis mechanism. A larger enclosed area suggests more energy is being dissipated, which is a characteristic of damping behavior in the material. The dense concentration of loops indicates significant energy dissipation and cyclic degradation. The maximum and minimum points on the loops indicate the peak load and displacement values the clay experiences during the seismic events.

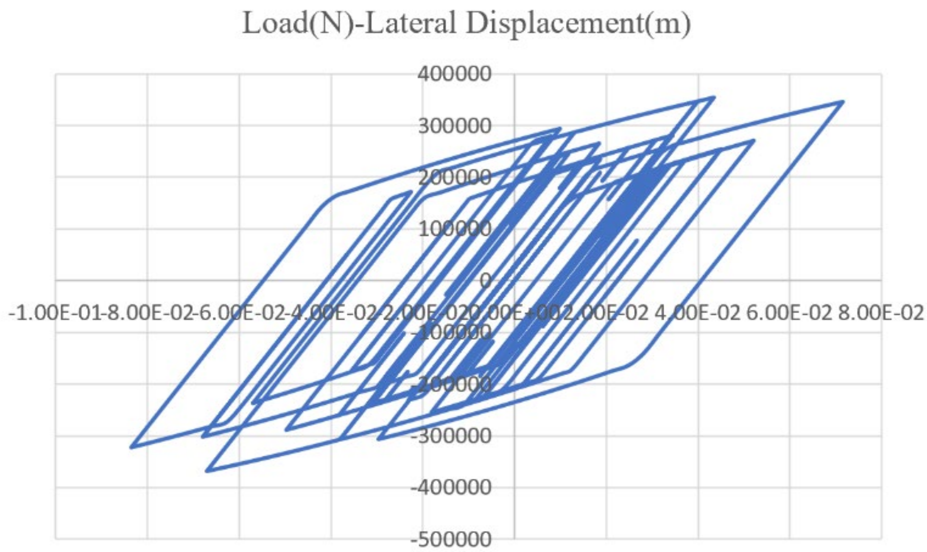


Figure 63: Seismic response of clay (PySimple1material) beneath a 10-story subjected to El Mayor ground motion

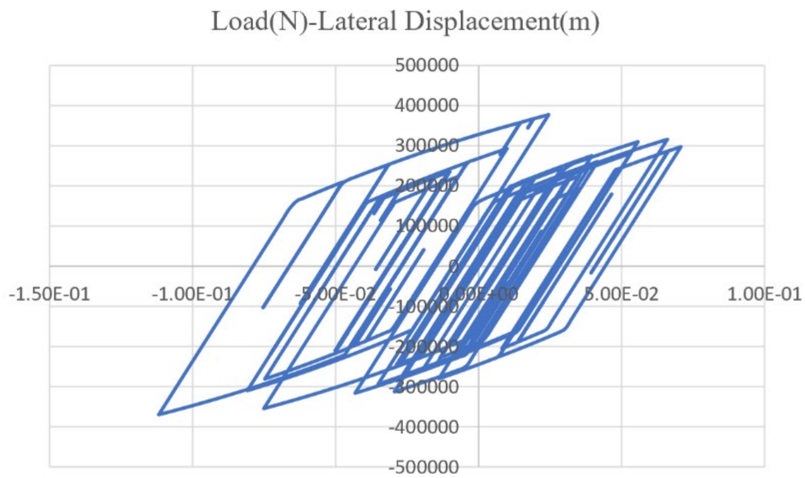


Figure 64: Seismic response of clay (PySimple1Material) beneath a 15-story structure subjected to Chi - Chi ground motion

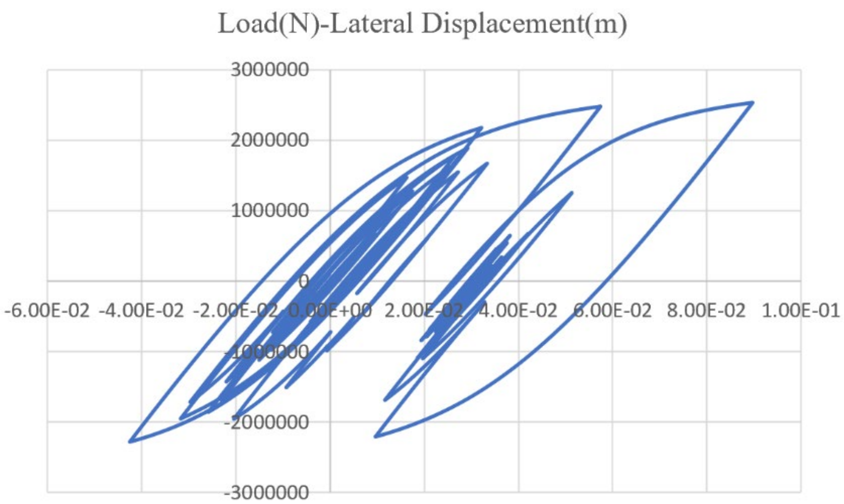


Figure 65: Seismic behavior of clay (TzSimple1 material) beneath a 7-story structure exposed to Darfield ground motion

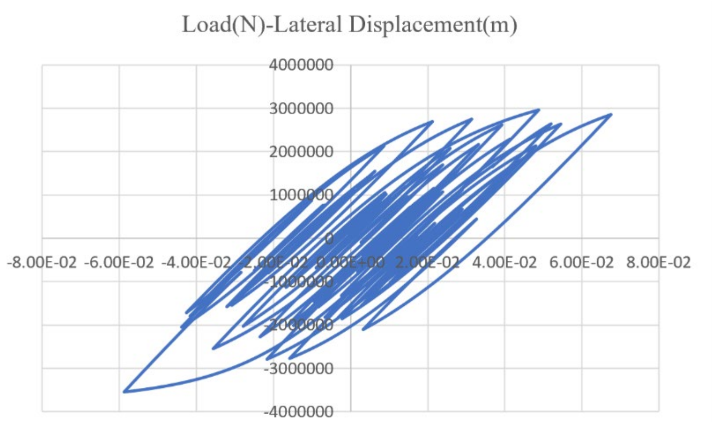


Figure 66: Seismic response of clay (TzSimple1 material) beneath a 10-story subjected to El Mayor ground motion

The plots 68 to 70 illustrate the seismic response of a clay material modeled using the QzSimple1 material in OpenSeesPy, as detailed in Chapter 3. This response is influenced by the Darfield, El Mayor, and Chi-Chi ground motions. The plots depict the relationship between load (in Newtons, N) and vertical displacement (in meters). The loops represent energy dissipation during cyclic loading, indicating the nonlinear response of the clay material under seismic stress.

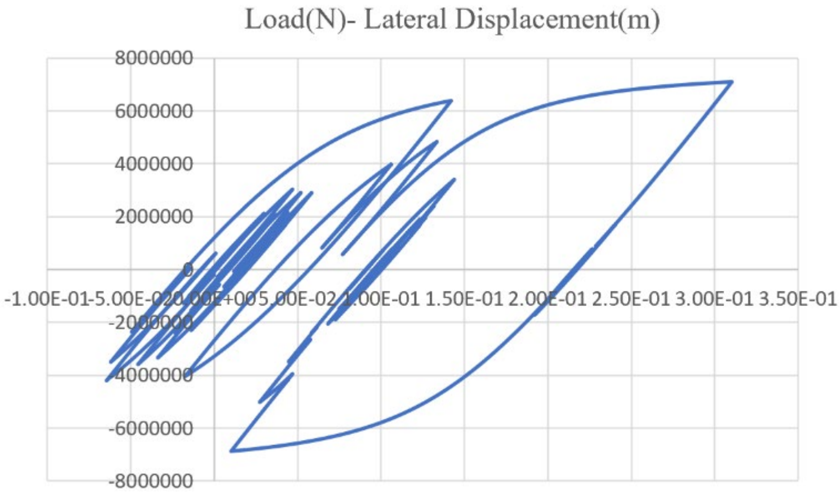


Figure 67: Seismic response of clay (TzSimple1 material) beneath a 15-story subjected to Chi – Chi ground motion

The x-axis represents vertical displacement, downward movement corresponds to negative values, and upward movement to positive values. The y-axis shows the load applied to the material, with negative values representing compressive load and Tensile load is shown by positive values

The initial sharp incline of the curve signifies the material stiffness. As the load increases, the curve flattens, indicating softening behavior. The further extension of the loops and the flattening of the curve suggest plastic deformation or failure in the clay material under the given loading conditions.

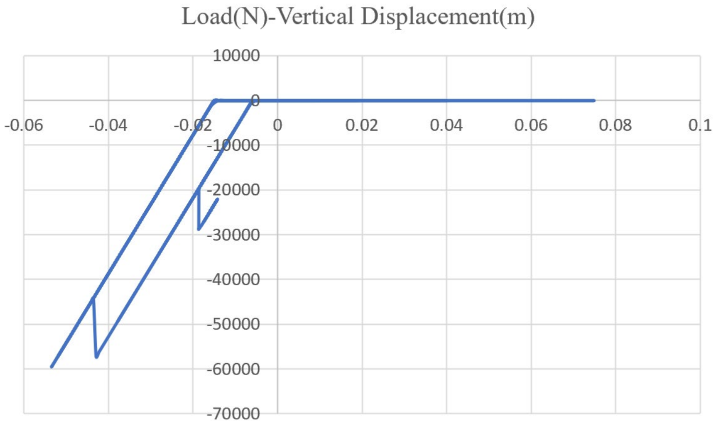


Figure 68: Seismic response of clay (QzSimple1 material) beneath a 7-story subjected to Darfield ground motion

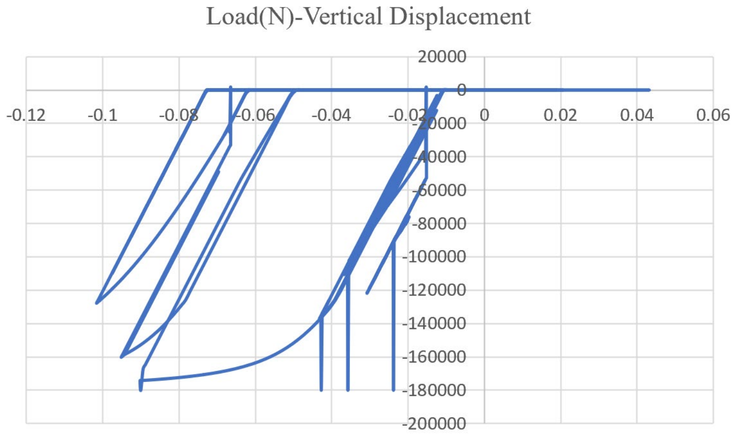


Figure 69: Seismic response of clay (QzSimple1 material) beneath a 10-story subjected to El Mayor ground motion

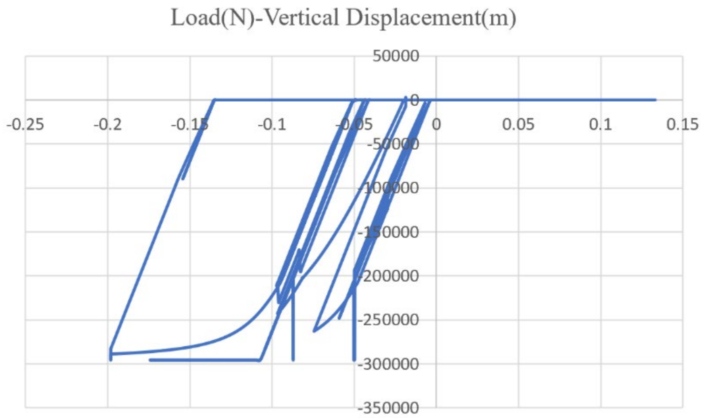


Figure 70: Seismic response of clay (QzSimple1 material) beneath a 15-story subjected to Chi - Chi ground motion

The provided plot (71 to 73) illustrates the vertical displacement over time for a structure affected by the Darfield, El Mayor, and Chi-Chi ground motions. The two lines in the plot, shown in different colors, represent the rocking movement on opposite sides of the structure. Vertical displacement is measured in meters, whereby upward progress is indicated by positive values and downward movement is shown by negative values. The differential displacement between these two sides, depicted by the rocking movement, shows how the structure tilts back and forth during the seismic event.

Initially, the vertical displacement is relatively small, indicating the structure's stability with minor movements. Subsequently, there is a suggestive growth in vertical displacement during the period of intense seismic activity. The peaks and troughs in this range reflect the rocking motion. The irregular and oscillating patterns of the lines during this period indicate energy dissipation through rocking, with the highest peaks and deepest troughs representing the maximum rocking motion experienced by the structure.

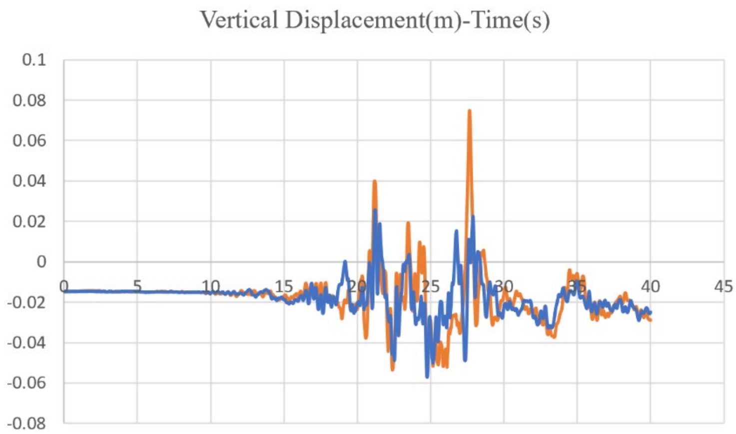


Figure 71: Rocking movement of 7-story structure subjected to Darfield ground motion on two opposite sides

Over time, the vertical displacement gradually decreases, suggesting that the structure is returning to a more stable state as the seismic activity subsides.

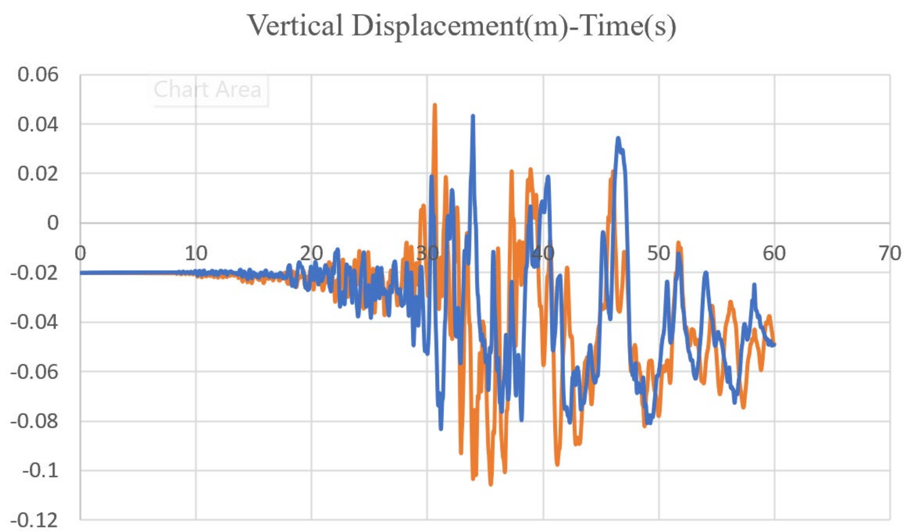


Figure 72: Rocking movement of 10-story structure subjected to El Mayor ground motion on two opposite sides

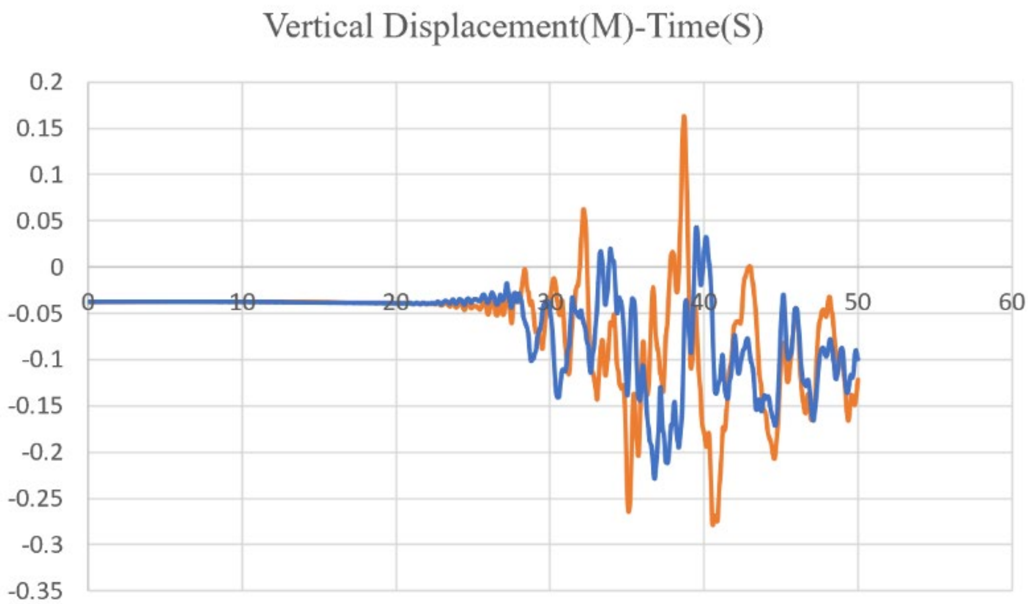


Figure 73: Rocking movement of 15-story structure subjected to Chi - Chi ground motion on two opposite sides

To capture the foundation's rotation during the earthquake, I used a zero-length spring, as explained in Chapter 3. These plots (74, 75) illustrate how the foundation rotates around the coordinate axes in response to the Darfield and El Mayor ground motions, highlighting the periods of maximum rotational movement and the eventual return to stability.

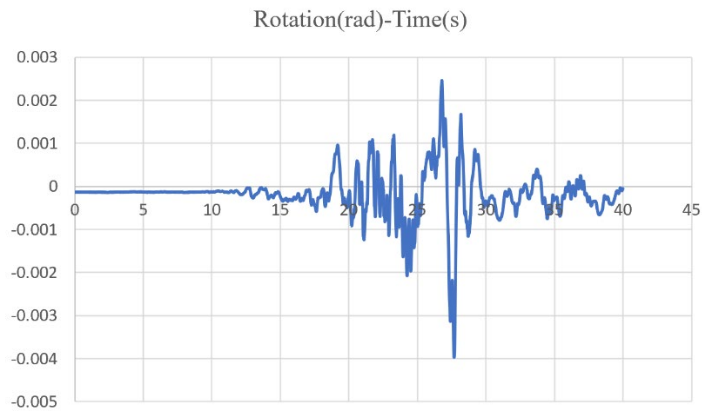


Figure 74: Rotation of foundation around X direction in response to Darfield ground motion

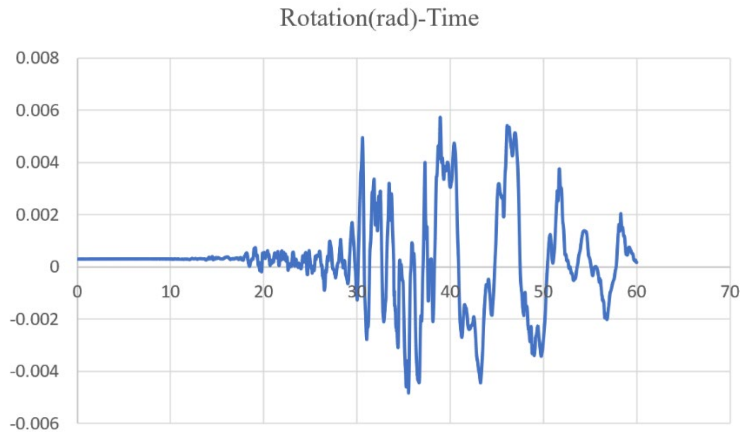


Figure 75: Rotation of foundation around Y Direction in response to El Mayor ground motion

4.3 Structural Behavior on Sand Soil With and Without SSI

In this section, the analysis results for 7-, 10-, and 15-story buildings constructed on sand soil and subjected to the ground motions of Kobe, Iwate, and Chi- Chi are presented. Table 40 displays the first-period values under two scenarios: one with a fixed foundation assumption and the other with an SSI analysis.

Table 40: First period for fixed foundation and SSI analysis

Story	T1(s)-Fixed	T1(s)-SSI
7	0.686	0.735
10	0.894	0.956
15	1.512	1.575

The first natural period is longer for the SSI scenario compared to the fixed foundation scenario. This difference occurs because, in reality, the interaction between the building and the soil tends to make the structure more flexible, leading to a longer natural period. However, this increasing in comparison to clay soil is less.

The maximum drift of stories for the 7, 10 and 15-story buildings subjected to the Kobe, Iwate and Chi-Chi ground motions are presented in Table 41, 42 and 43. In certain stories, the maximum drift is reduced, while in others, it is increased.

Table 41: Maximum drift of 7 stories on sand soil (Type C) subjected to Kobe ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	1.56	0	1.43
1	2.37	1.26	2.41	1.4
2	2.28	2.1	3	2.91
3	2.58	2.61	3.83	3.44
4	2.66	2.45	3.75	3.72
5	2.81	2.52	3.9	3.6
6	3.6	2.8	3.5	4
7	3.6	3.2	4.1	3.8

Table 42: Maximum drift of 10 stories on sand soil (Type C) subjected to Iwate ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	2.03	0	1.95
1	2.67	5.48	1.9	0.7
2	3.18	2.43	2.1	1.94
3	3.59	3.17	2.7	2.22
4	3.96	3.75	2.1	2.32
5	4.1	3.4	2.7	2.57
6	4	3.8	3.1	2.5
7	3.8	3.6	2.9	2.7
8	3.7	3.4	2.1	2.5
9	2.4	3	2.2	1.8
10	3	3.1	2.2	1.7

Table 43: Maximum drift of 15 stories on sand soil (Type C) subjected to Chi- Chi ground motion

Story	Max Drift-Fix-X(cm)	Max Drift-SSI-X(cm)	Max Drift-Fix-Y(cm)	Max Drift-SSI-Y(cm)
Base	0	0.52	0	0.6
1	2.52	2.12	3.95	1.35
2	5	1.92	7.95	6.1
3	1.68	1.8	10.52	2.72
4	1.86	1.94	2.68	2.88
5	14.72	1.93	21.5	2.85
6	2.7	20.02	3.1	2.7
7	2.9	3.1	3	2.1
8	3	3.1	2.8	2.3
9	3.2	2.8	1.9	1
10	3.1	3.2	2.1	3
11	3.2	3.1	1.8	3.3
12	3.2	2.7	1.5	3.2
13	3	2.4	1.9	2.7
14	3.4	2.1	1.9	2.4
15	3.2	2.2	1.8	2.2

Figures 76 to 78 illustrate the acceleration (m/s²) over time for the top floor of 7, 10, and 15-story structures under Kobe, Iwate and Chi- Chi earthquake ground motions, under both fixed and SSI conditions.

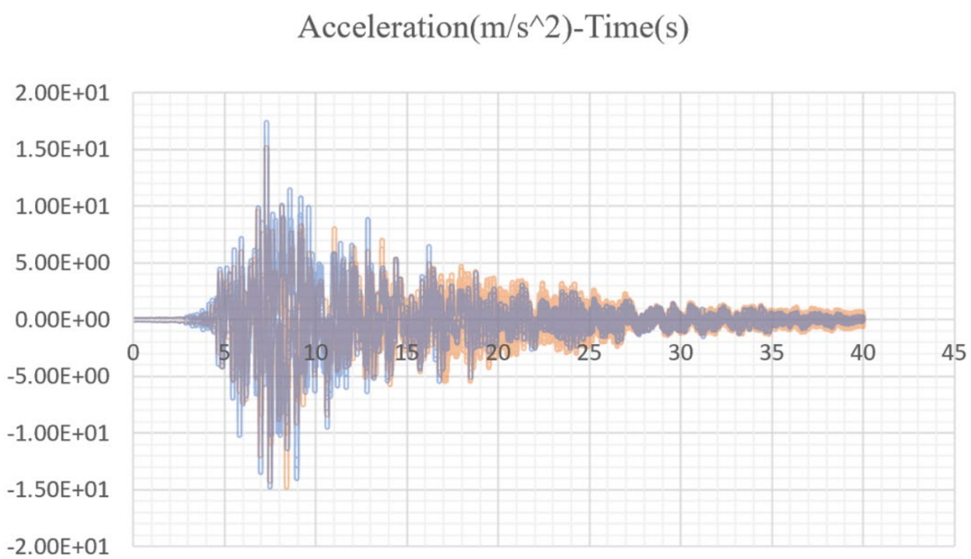


Figure 76: Acceleration on roof of a multi-story(7-story) building under Kobe in the X axe – orange line for fixed foundation Condition- blue line for SSI condition

The maximum acceleration sometimes is increased in the SSI analysis. For example, as illustrated in picture 77, the pick acceleration in SSI condition is grown in comparison to fixed foundation condition.

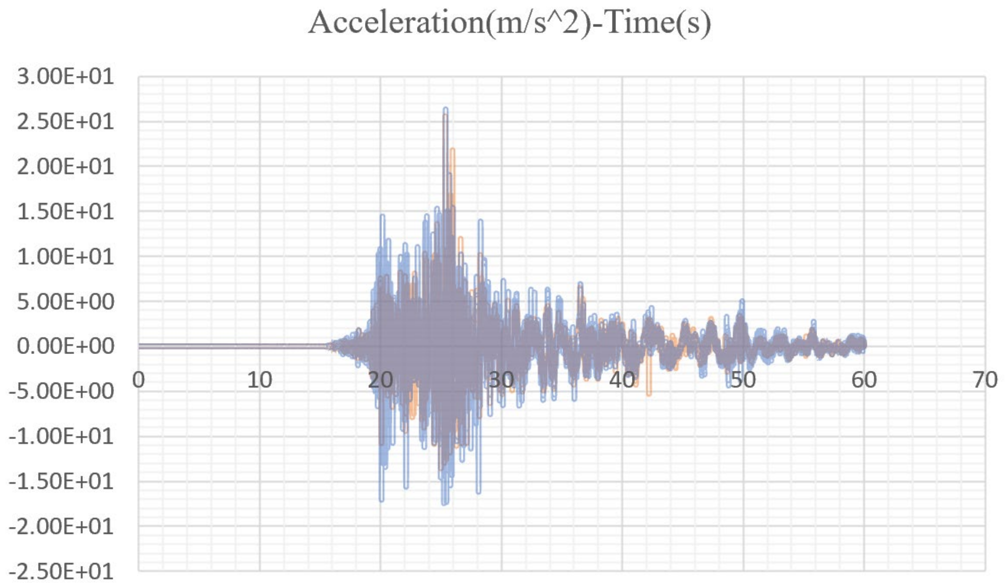


Figure 77: Acceleration on the top floor of a 10-Story building under Iwate in the X direction – orange line fixed foundation condition- blue line for SSI condition

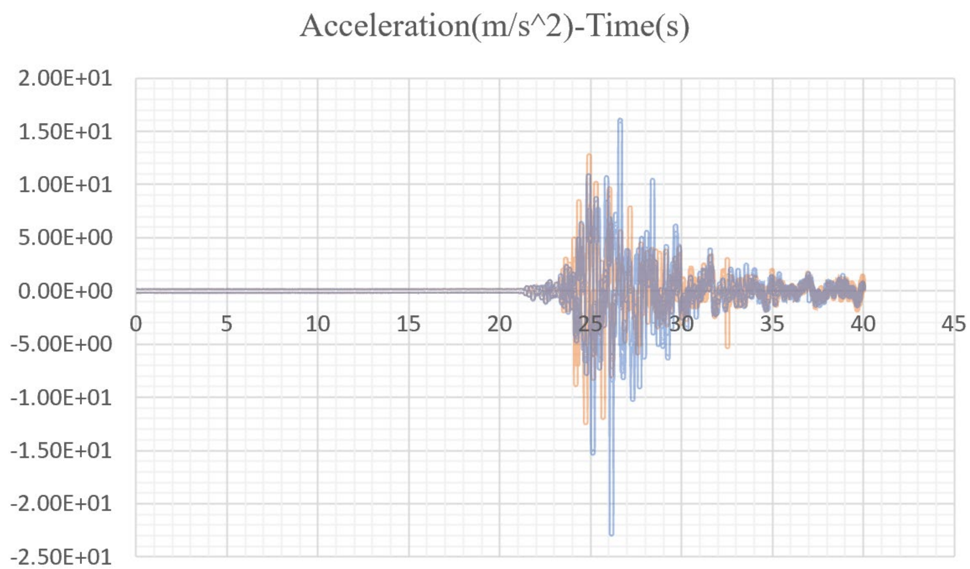


Figure 78: Acceleration on the top floor of a 15-Story building under Chi-Chi in the Y direction – orange line for fixed foundation condition- blue line for SSI condition

Figures 79 to 84 present the analysis results for columns and beams under fixed and SSI conditions. In this model, a zero-length spring was used to simulate the deterioration strength of the columns and beams in response to earthquakes, as discussed in Chapter 3. The results demonstrate a significant difference between the two conditions. The provided figures illustrate the hysteresis behavior of columns and beams in 7-story, 10-story, and 15-story buildings subjected to the Kobe, Iwate, and Chi-Chi ground motions, respectively. These graphs plot moment (N.m) versus rotation (radians) for two different foundation conditions.

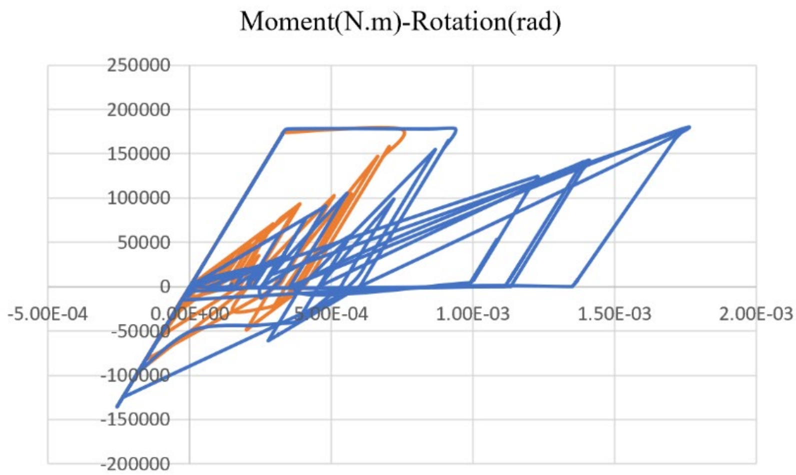


Figure 79: Hystertic behavior of column (10, 5, 20.8) in 7-Story building subjected to Kobe ground motion - blue line for SSI Condition, orange line for fixed foundation condition

The line with color of blue represents the moment-rotation behaviour of the columns and beams considering SSI. In this condition, the flexibility of the foundation and its interaction with the soil are accounted for.

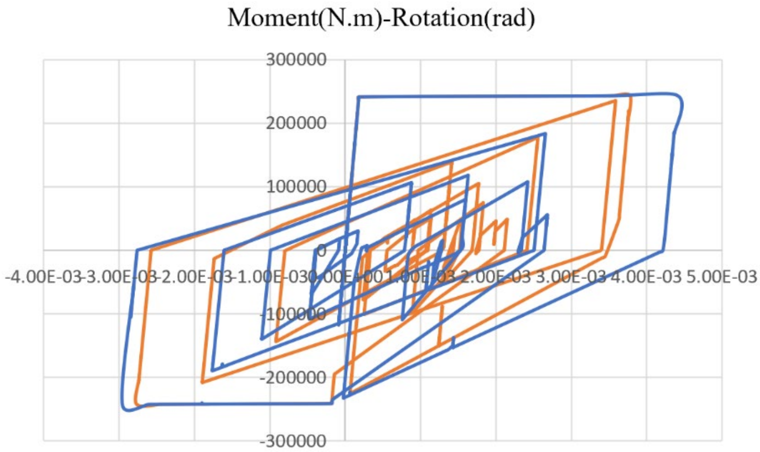


Figure 80: Hystertic behavior of column (10, 5, 29.75) in 10-Story building subjected to Iwate ground motion - blue line for SSI Condition, orange line for fixed foundation condition

The resulting hysteresis loops indicate the absorbing of energy and stiffness degradation characteristics of the column under seismic loading. The orange line shows the moment-rotation behaviour of the same column assuming a fixed foundation, meaning it does not account for SSI. The hysteresis loops for this condition are typically smaller and exhibit less energy dissipation compared to the SSI condition because the foundation's flexibility is not considered. However, in some cases, such as illustrated in Figure 81, this behavior can be reversed.

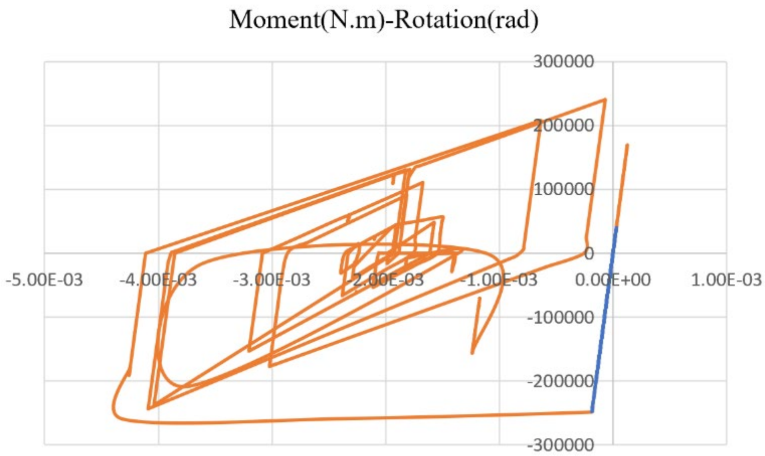


Figure 81: Hystertic behavior of column (0, 0, 44.75) in 15-Story building subjected to Chi-Chi ground motion - blue line for SSI Condition, orange line for fixed foundation condition

The differences between the two hysteresis curves highlight the impact of SSI on the seismic behaviour of the column. The SSI condition generally results in larger rotations and moments, indicating a more flexible system that can absorb and dissipate more energy during an earthquake. In contrast, the fixed foundation condition exhibits stiffer behavior with less energy dissipation capacity.

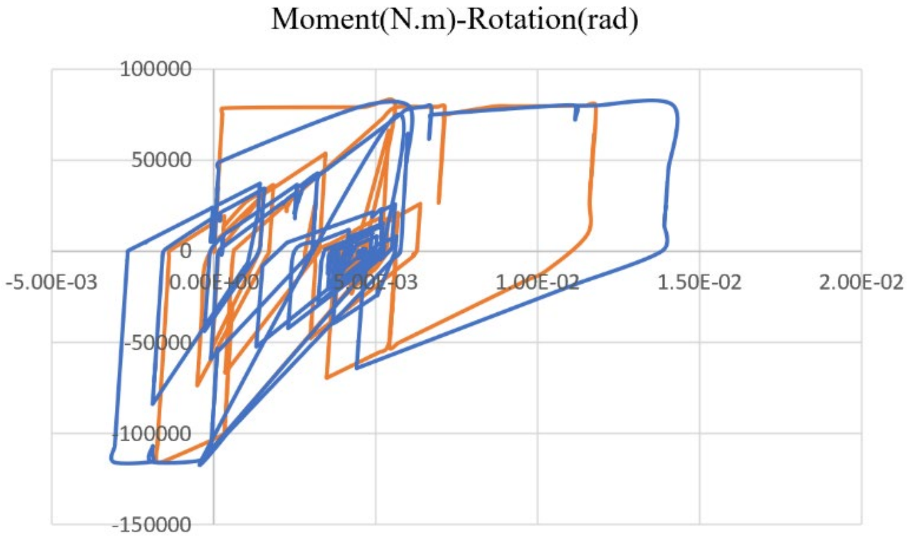


Figure 82: Hysteretic response in beam (10, 4.8, 21) in 7-story building subjected to Kobe ground motion - blue line for SSI condition, orange line for fixed foundation condition

Overall, the figures emphasize the importance of considering SSI in seismic analysis to capture a more realistic response of the structural elements. The blue line's loops (SSI condition) exhibit larger moment values and broader rotation ranges compared to the orange line (fixed foundation condition).

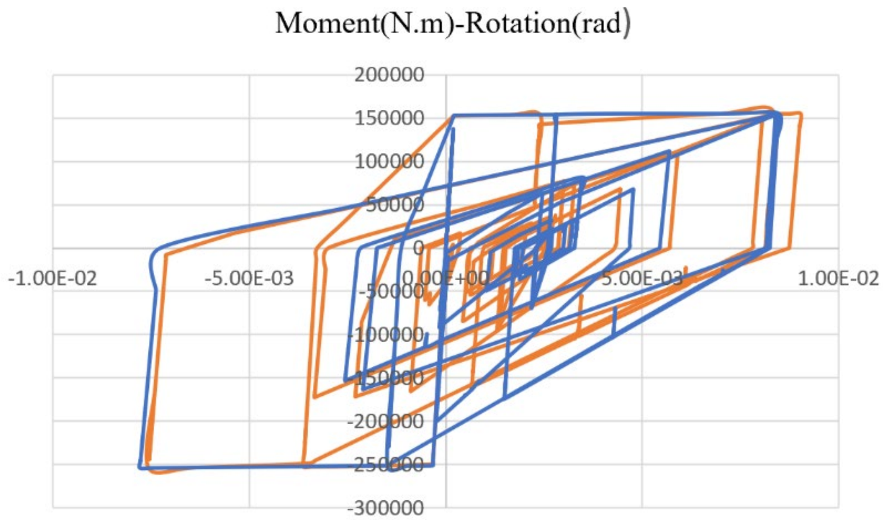


Figure 83: Hysteretic response in beam (0, 0.25, 3) in 10-story building subjected to Iwate ground motion - blue line for SSI condition, orange line for fixed foundation condition

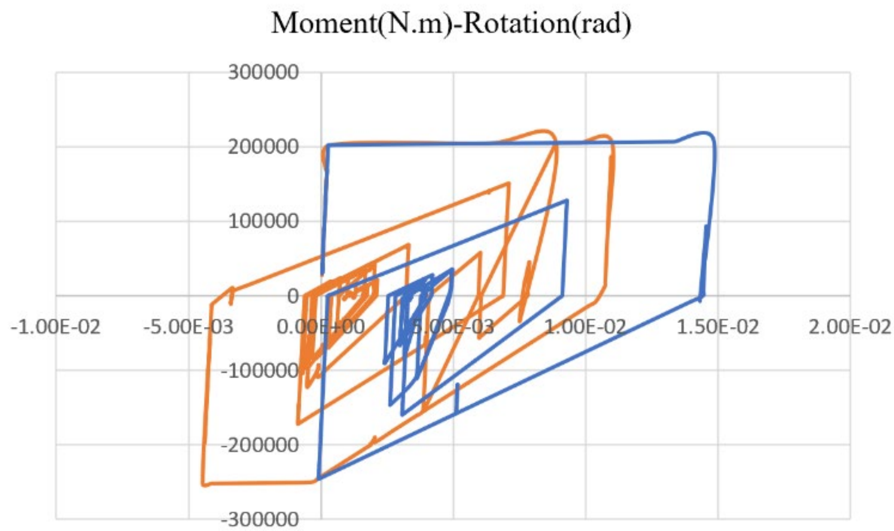


Figure 84: Hysteretic response in beam (0, 4.75, 6) in 15-story building subjected to Kobe ground motion - blue line for SSI condition, orange line for fixed foundation condition

This suggests that SSI amplifies the interactions in the column. The fixed condition (orange line) shows relatively smaller hysteresis loops, indicating a stiffer and less flexible response compared to the SSI condition. However, as discussed in section 4.2, when clay is present beneath the structure, the fixed condition can result in higher moments and rotations with greater energy dissipation in the elements when fixed.

Figures 85 to 87 depict the hysteresis response of a steel bar in the first-floor shear wall of 7-story, 10-story, and 15-story buildings under the Kobe, Iwate, and Chi-Chi ground motions. The graphs present hysteresis loops that show the relationship between stress (y-axis, in N/m²) and strain (x-axis, in m/m). These loops represent the cyclic loading and offloading behavior of the steel bar during the ground motion.

The blue line represents the hysteresis response under the SSI condition. This leads to more complex behavior compared to a fixed foundation. The blue loops illustrate how the steel bar responds to cyclic loads when soil flexibility is taken into account.

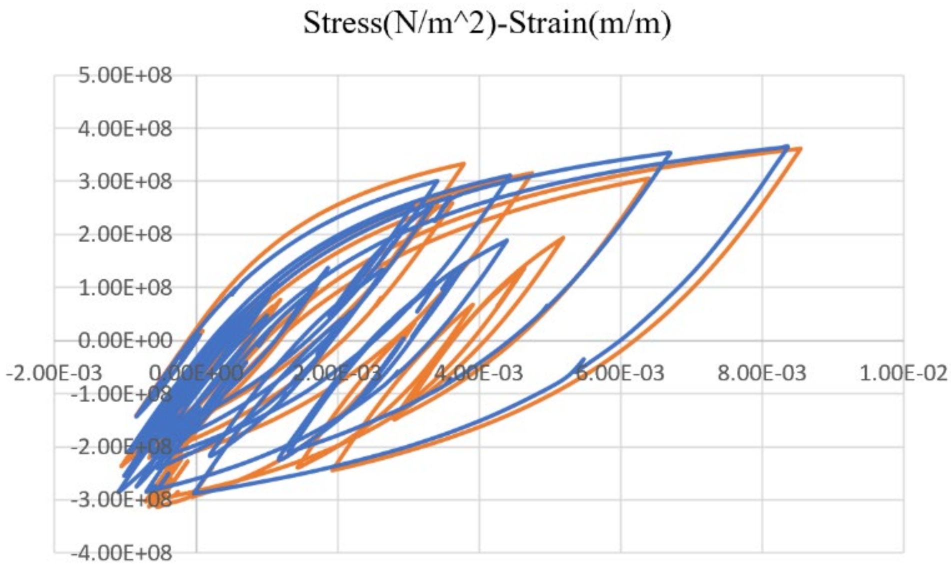


Figure 85: Hysteretic response of steel bar in first-floor shear wall - blue line represents SSI condition, orange line represents fixed foundation condition under Kobe ground motion

The orange line represents the hysteresis response under a fixed foundation condition. This condition assumes that the base of the shear wall is perfectly rigid and does not interact with the underlying soil. The orange loops demonstrate the response of the steel bar under this idealized condition.

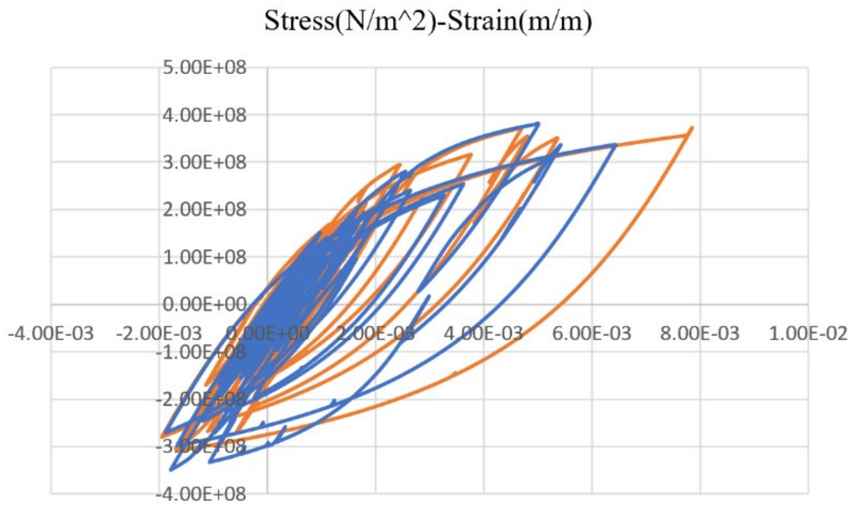


Figure 86: Hysteretic response of steel bar in first-floor shear wall - blue line represents SSI condition, orange line represents fixed foundation condition under Kobe ground motion

The two sets of hysteresis loops highlight the differences in structural response due to the foundation conditions. Typically, the SSI condition (blue line) shows less spread-out and irregular loops due to the additional deformations and energy dissipation mechanisms introduced by the soil-structure interaction. The fixed foundation condition (orange line) shows more regular and unconfined loops, indicating a more straightforward cyclic response.

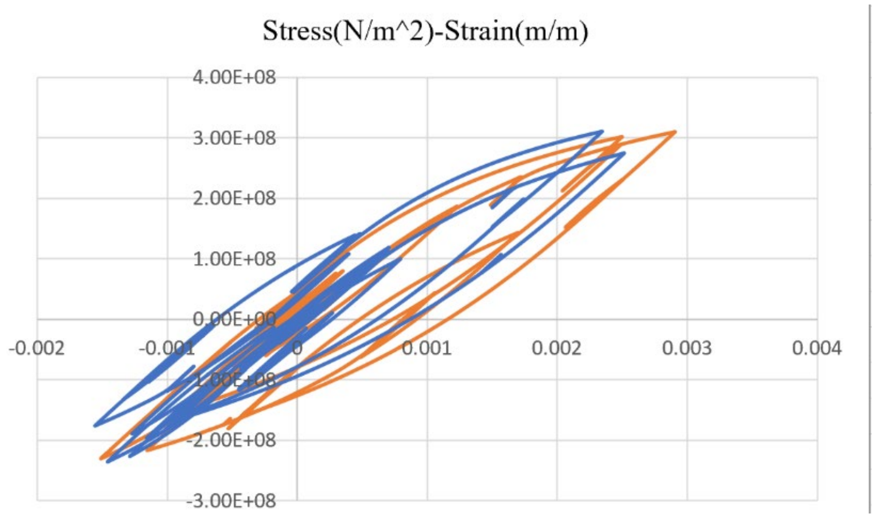


Figure 87: Hysteretic response of steel bar in first-floor shear wall - blue line represents SSI condition, orange line represents fixed foundation condition under Chi- Chi ground motion

However, when clay is present under the structure, the behavior can be reversed. In such cases, the stress and strain are higher under the fixed foundation condition. This phenomenon is due to the clay's influence on the foundation's rigidity and the overall structural response. These figures emphasize the importance of considering SSI in seismic assess to capture a more realistic response of structural elements. The hysteresis loops provide insight into the energy dissipation and stiffness degradation characteristics of the steel bar under different foundation conditions.

The two images (88, 89) provided appear to represent the hysteresis behavior of unconfined concrete in shear walls under different conditions when has been affected by Kobe. The graph shows the relationship between stress (N/m^2) and strain (m/m). Both figures reach a peak stress of approximately 2.0E+06 N/m^2, but the minimum stress in Figure 88 is significantly lower, indicating a more extensive stress range due to SSI conditions. Both figures have similar strain ranges, indicating that the deformation under both conditions is comparable. The loop in Figure 88 is larger and more spread out, indicating higher energy dissipation and more significant cyclic effects due to soil-structure interaction.

These differences highlight the impact of SSI on the hysteresis response of unconfined concrete in shear walls. The presence of SSI conditions results in greater energy dissipation and a broader range of stress values, affecting the overall structural response to seismic events.

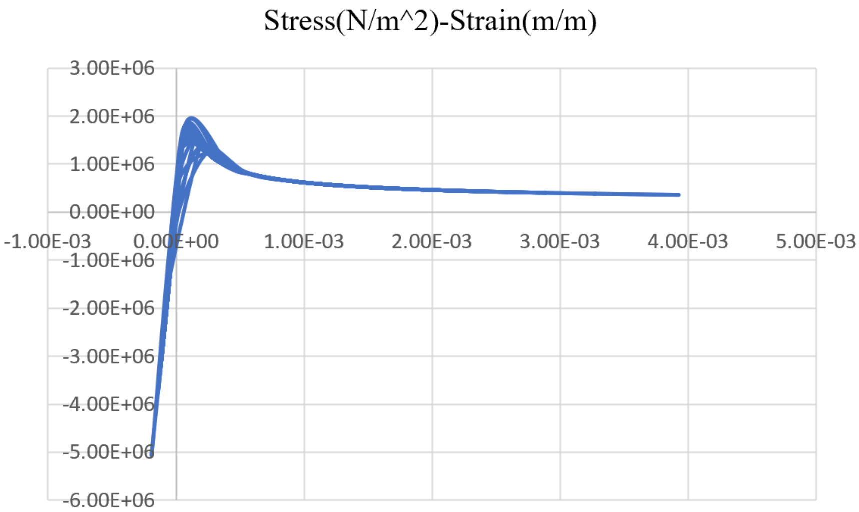


Figure 88: Hysteretic behavior of unconfined concrete in the shear wall of 7-story on the first floor under fixed foundation conditions under Kobe ground motion

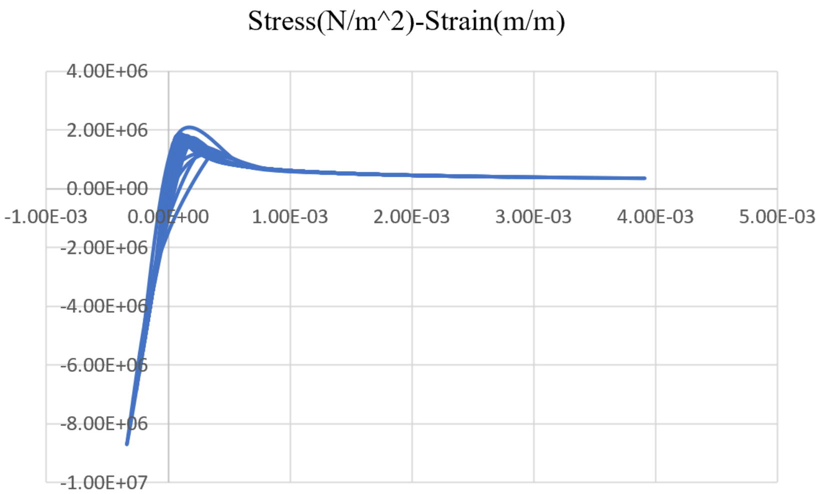


Figure 89: Hysteretic behavior of unconfined concrete in the shear wall on the first floor under SSI conditions under Kobe ground motion

Figures 90 to 92 illustrate the relationship between lateral load (N) and curvature (1/m) in the shear wall subjected to the Kobe, Iwate, and Chi-Chi ground motions, respectively. Each graph features two curves: the blue curve represents the soil-structure interaction (SSI) condition, while the orange curve represents the fixed foundation condition.

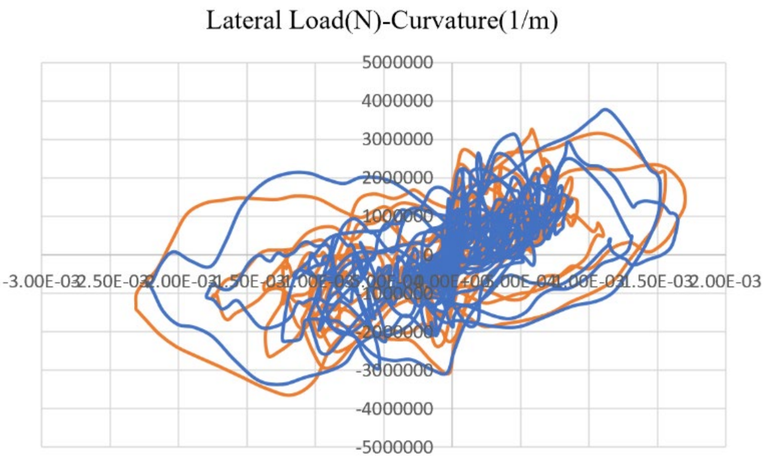


Figure 90: Lateral load-curvature relationship in the shear wall on the first floor subjected to Kobe ground motion. The blue curve represents the SSI condition, while the orange curve represents the fixed foundation condition.

Curvature in a shear wall measures the bending deformation experienced by the wall under lateral loads, such as those from seismic activity. It is a crucial indicator of the wall's flexural behavior and overall response to these forces. Higher curvature values indicate greater bending and deformation of the wall, which helps assess the structural performance and stability of the wall during seismic events. This measurement provides insights into the wall's ability to withstand lateral forces and maintain its integrity.

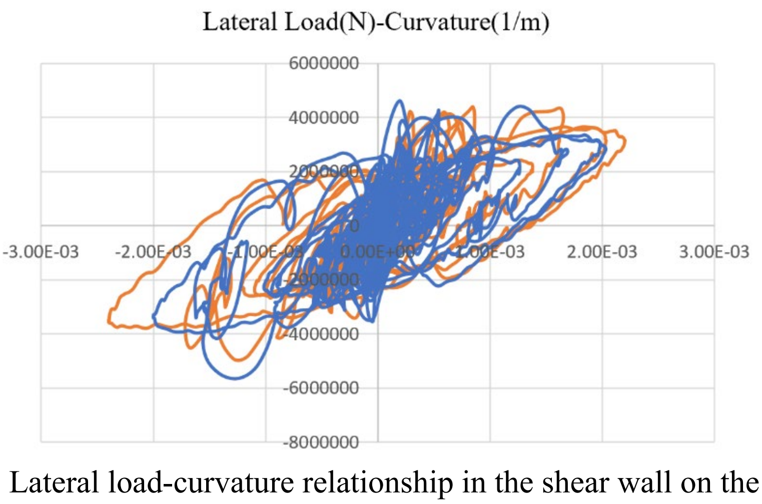


Figure 91: Lateral load-curvature relationship in the shear wall on the first floor subjected to Kobe ground motion. The blue curve represents the SSI condition, while the orange curve represents the fixed foundation condition.

Curvature is also associated with the wall's ability to dissipate energy during cyclic loading, such as during an earthquake. A wall that can bend more (higher curvature) without failing can absorb and dissipate more energy, thereby reducing the overall seismic impact on the building. By comparing the curvature under different foundation conditions, engineers can Assess the impact of soil-structure interaction on the performance of the shear wall.

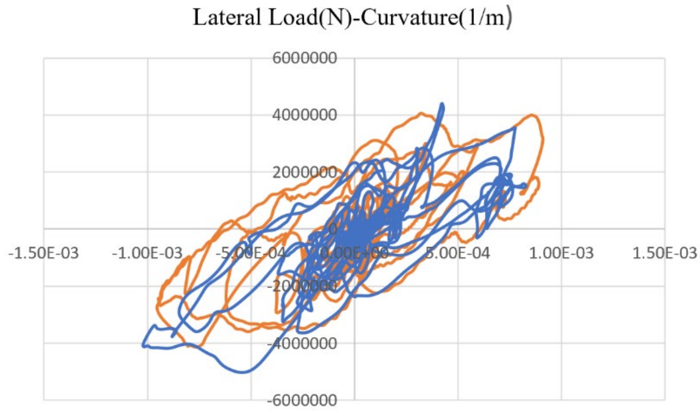


Figure 92: Lateral load-curvature relationship in the shear wall on the first floor subjected to Chi-Chi ground motion. The blue curve represents the SSI condition, while the orange curve represents the fixed foundation condition.

In more detail, stiff sand provides a firmer support for the structure, leading to reduced flexibility and deformation under horizontal loads. As a result, the behavior of the shear wall under the SSI condition with stiff sand is closer to that of the fixed foundation condition. This means that the shear wall's response, in terms of bending deformation (curvature) and lateral load capacity, shows less variation between these two conditions when supported by stiff sand.

Figures 93 to 95 depict the horizontal forces (N) versus horizontal movement (m) relationship for sand modeled using the PySimple1 material beneath a 7-story, 10-story and 15-story buildings subjected to Kobe, Iwate and Chi- Chi ground motions.

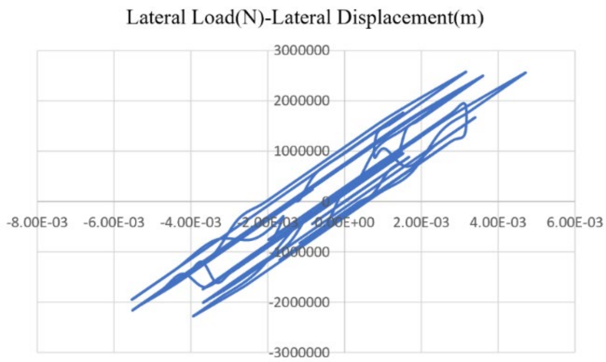


Figure 93: Seismic response of sand (PySimple1 material) beneath a 7-story subjected to Kobe ground motion

These graphs aid in comprehending the soil-structure interaction and the seismic behavior of the foundation soil. The graphs show multiple loops indicating cyclic behavior typical in seismic response analysis. These loops represent the absorption of energy in the soil resulting from the cyclic loading and unloading during the earthquake. The inclination and shape of the loops provide insight into the stiffness and damping characteristics of the soil.

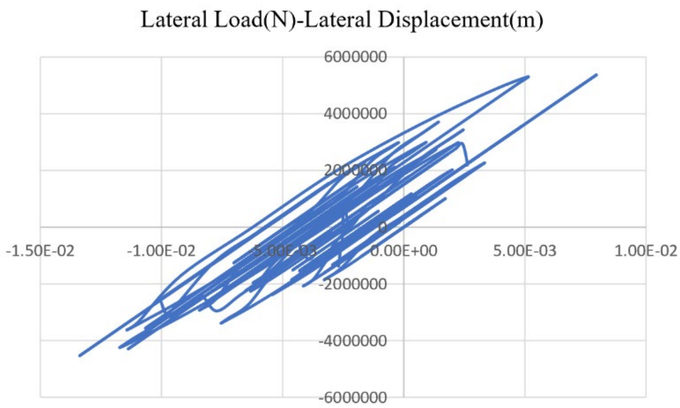


Figure 94: Seismic response of sand (PySimple1 material) beneath a 10-story subjected to Iwate ground motion

A steeper slope would indicate higher stiffness, while the area within the loops indicates the amount of energy dissipated, representing the damping capacity. The non-linear nature of the loops signifies that the sand exhibits non-linear behavior under seismic loading. The presence of hysteresis loops in the graph indicates that the sand material dissipates energy during seismic events, which is beneficial as it reduces the energy transmitted to the building structure.

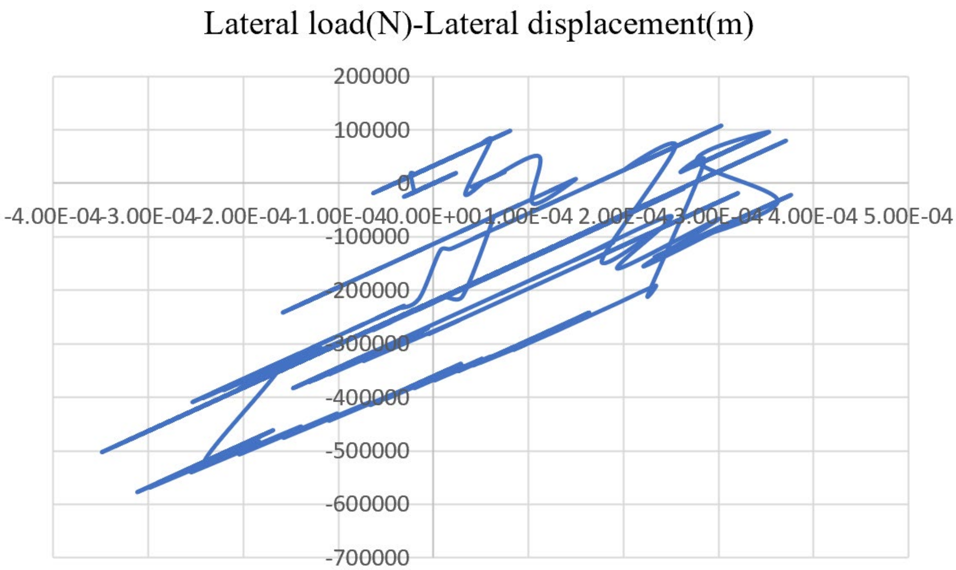


Figure 95: Seismic response of sand (PySimple1 material) beneath a 15-story subjected to Chi-Chi ground motion

Figures 96 to 98 illustrate the relationship between lateral load (N) and displacement (m) for sand modeled using the TzSimple1 material beneath 7-story, 10-story and 15-story structures when exposed to Kobe, Iwate and Chi- Chi ground motions. These graphs help in understanding the soil-structure interaction and the reaction of the earthquake of the foundation soil under this specific material model.

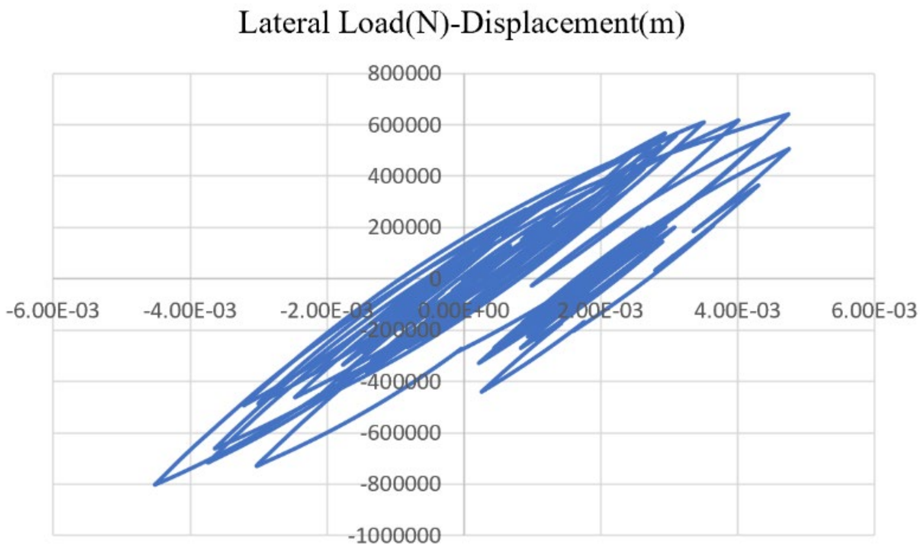


Figure 96: Seismic behavior of sand (TzSimple1 material) beneath a 7-story structure exposed to Kobe ground motion

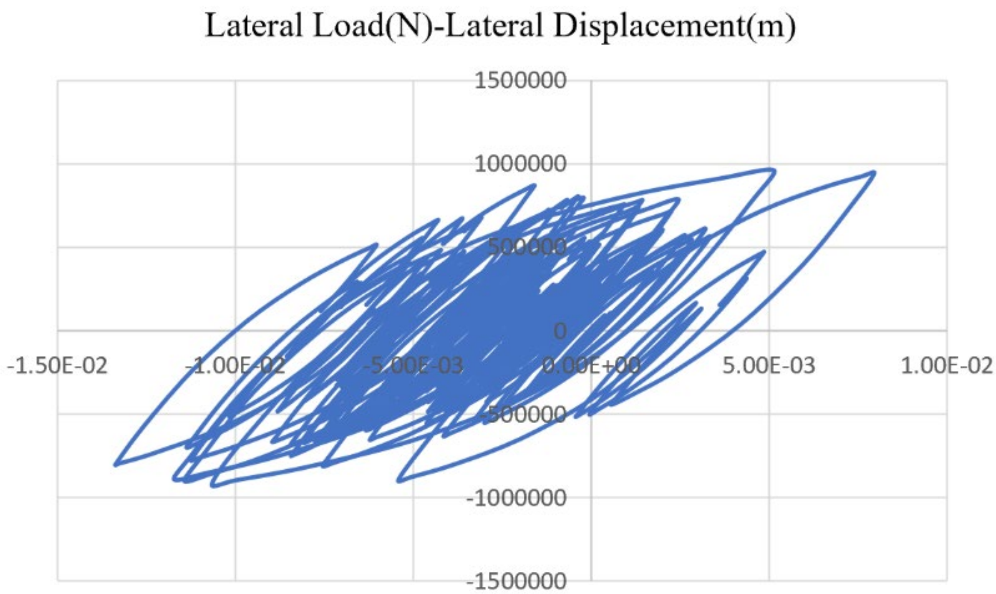


Figure 97: Seismic behavior of sand (TzSimple1 material) beneath a 10-story structure exposed to Iwate ground motion

In summary, figures, illustrate the complex interaction between the lateral load and displacement for sand modeled with TzSimple1 material. The cyclic loops highlight the soil's ability to dissipate energy and its non-linear behavior under seismic loading, which are key factors in ensuring the building's seismic resilience.

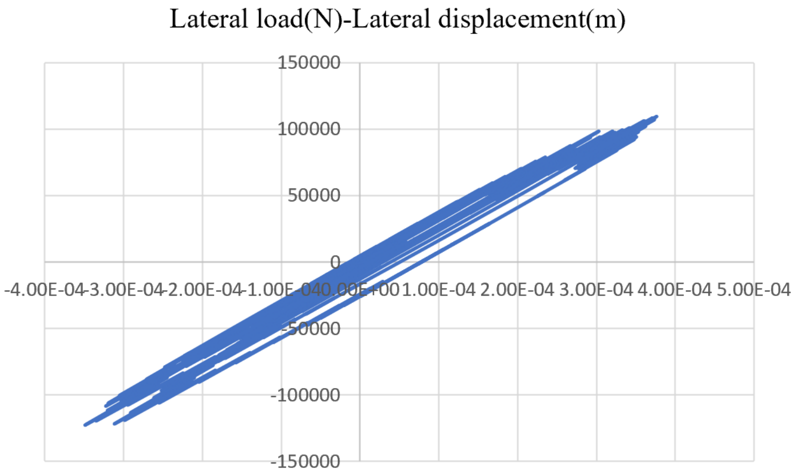


Figure 98: Seismic behavior of sand (TzSimple1 material) beneath a 15-story structure exposed to Chi- Chi ground motion

Figures 99 to 101 illustrate the connection between vertical force (N) and vertical movement (m) for sand modeled using the QzSimple1 material beneath 7-story, 10-story, and 15-story structures during the Kobe, Iwate, and Chi-Chi ground motions. These graphs provide insights into the vertical behavior and settlement characteristics of the soil foundation under seismic loading.

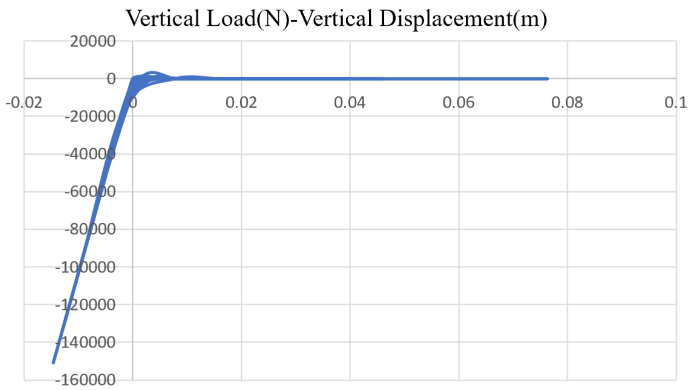


Figure 99: Seismic response of sand (QzSimple1 material) beneath a 7-story subjected to Kobe ground motion

The vertical displacement ranges from approximately -0.02 m to 0.08 m, showing the extent of vertical movement (settlement or uplift) experienced by the soil foundation beneath the buildings during seismic events. The graph initially displays a steep slope, indicating high stiffness during the early loading phase, followed by a plateau, which signifies stabilization in the soil's response to seismic loading.

The initial steep slope reflects the soil's high stiffness and resistance to vertical forces at the beginning of the loading phase. As the load increases, the soil undergoes initial compression. The plateau region indicates that after reaching a certain load, the soil material stabilizes, and additional load results in minimal additional displacement. This stabilization is crucial for foundation stability, as it limits excessive settlement.

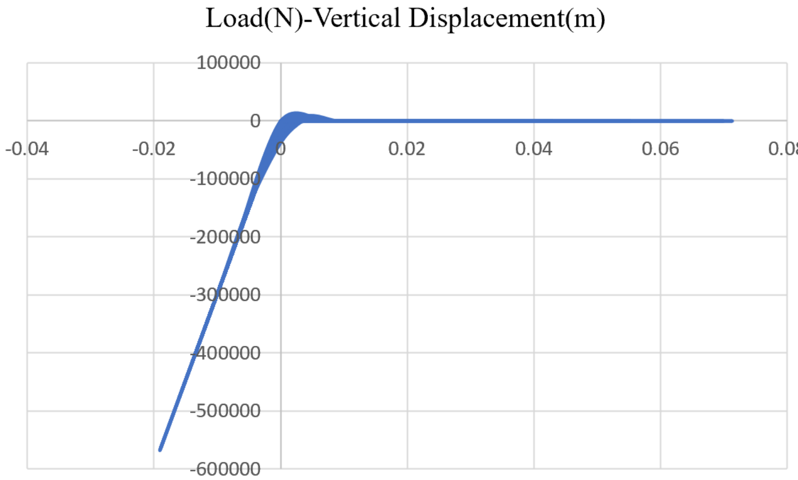


Figure 100: Seismic response of sand (QzSimple1 material) beneath a 10-story subjected to Iwate ground motion

The hysteresis loops near the origin suggest cyclic behavior, indicating some level of energy dissipation through minor vertical movements. However, this energy dissipation is less pronounced compared to lateral loading scenarios.

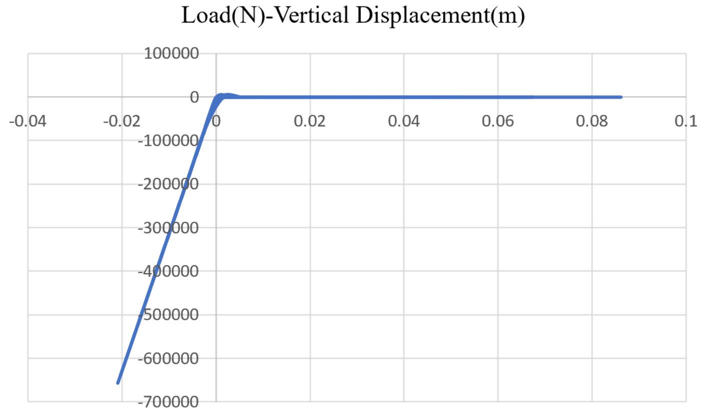


Figure 101: Seismic response of sand (QzSimple1 material) beneath a 15-story subjected to Chi-Chi ground motion

The plateau in the graph shows that the soil material reaches a state where further displacement is minimal despite increased load, highlighting the soil's ability to stabilize under continuous loading conditions. Overall, these figures underscore the importance of understanding the vertical load-displacement relationship for designing stable foundations in seismic regions. The initial high stiffness and subsequent stabilization of the soil foundation help limit excessive settlement, contributing to the overall stability of the structures. Figures 102 to 104 illustrate the vertical displacement over time for a structure subjected to the Kobe, Iwate, and Chi-Chi ground motions. The plot features two lines in different colors, representing the rocking movement on opposite sides of the structure.

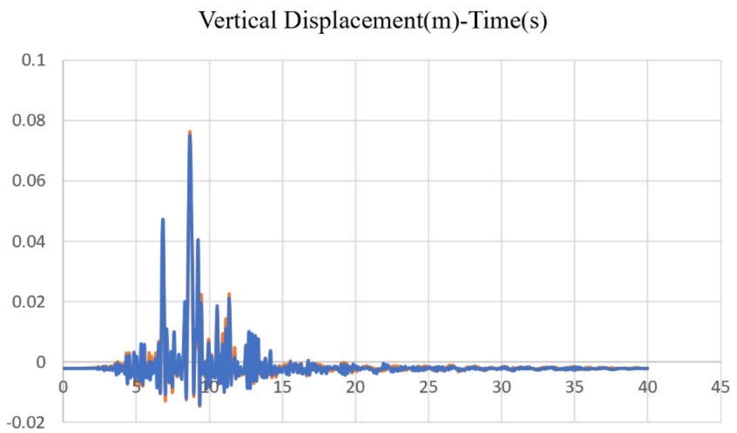


Figure 102: Rocking movement of 7-story structure subjected to Kobe ground motion on two opposite sides

In these plots, the vertical displacement reflects the cornerstone of the structure's dynamic reactivity to seismic activity. Rocking movement, indicated by the alternating vertical displacements on either side of the structure, is a crucial mechanism for energy dissipation during seismic events. However, when the building has been constructed on stiff sand soil, the rocking movement is not as pronounced compared to when it is supported by clay soil.

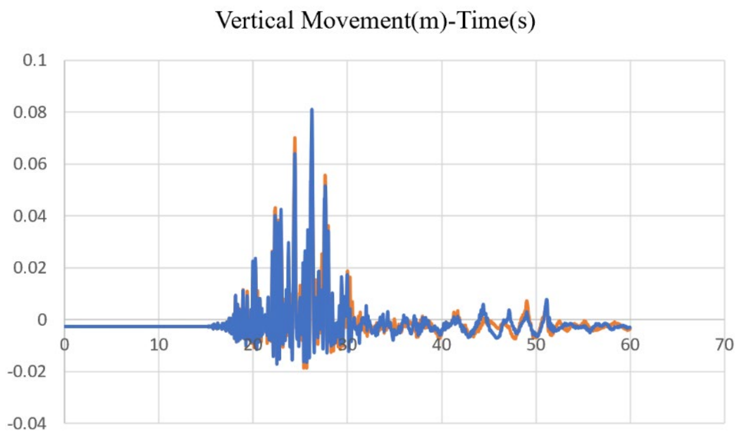


Figure 103: Rocking movement of 10-story structure subjected to Iwate ground motion on two opposite sides

Stiff sand provides a more rigid support base, which limits the extent of rocking motion. This reduced rocking movement results in less energy dissipation through this mechanism. In contrast, clay soil, being more compressible and less rigid, allows for greater rocking movement, enhancing energy dissipation through larger vertical displacements.

The plots demonstrate that, with stiff sand beneath the structure, the vertical displacement remains relatively low, indicating that the soil's rigidity restricts significant rocking motion. This behavior suggests that stiff sand effectively stabilizes the structure, but at the cost of reduced energy dissipation through rocking.

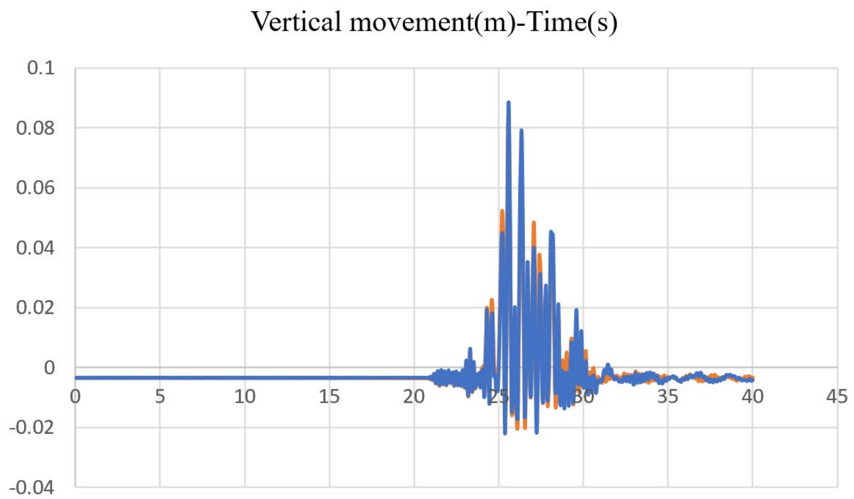


Figure 104: Rocking movement of 15-story structure subjected to Chi-Chi ground motion on two opposite sides

In summary, these figures emphasize the impact of soil type on the vertical displacement and rocking behavior of structures under seismic loading. While stiff sand limits rocking motion and maintains structural stability, it also results in lower energy dissipation through this mechanism compared to more flexible soils like clay. To capture the foundation's rotation during the earthquake, I used a zero-length spring, as explained in Chapter 3.

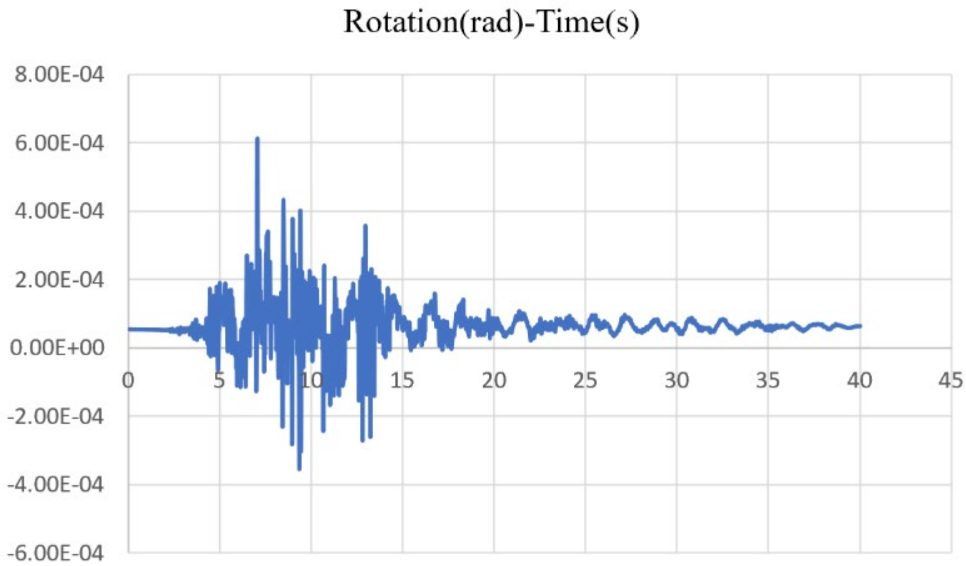


Figure 105: Rotation of foundation around X direction in response to Kobe ground motion

These plots (105, 106) illustrate how the foundation rotates around the coordinate axes in response to the Kobe and Iwate ground motions, highlighting the periods of maximum rotational movement and the eventual return to stability.

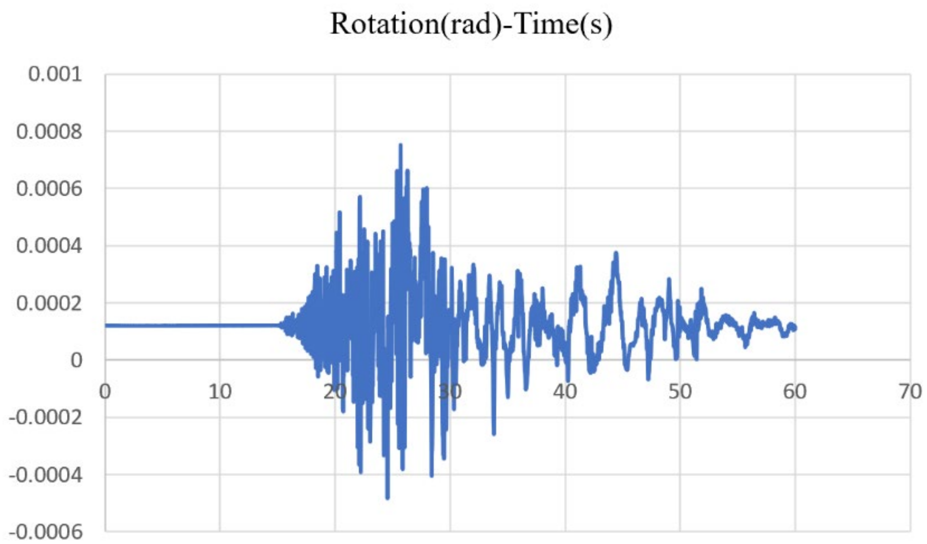


Figure 106: Rotation of foundation around Y direction in response to Iwate ground motion

Chapter 5

SUMMARY AND CONCLUSIONS

5.1 Summary

When a building is constructed on a shallow foundation, it can reduce earthquake energy through movement mechanisms like rocking, settling and sliding, thereby lowering the amount of forces acting on the superstructure. However, it is crucial to consider potential adverse effects on the foundation, such as excessive settlement, sliding, rocking, bearing failure, and significant movement requirements at the beam-column connections effective design necessitates a suitable modeling tool to comprehensively address these issues.

This study uses a model that represents the interaction between the foundation and the ground as a series of springs, dampers, and gaps. This model, called BNWF, accurately simulates how shallow foundations can bend, slide, and settle. Reinforced concrete walls are often used in earthquake-prone areas. Since these walls are expected to bend and flex during strong earthquakes, we need computer models that can accurately predict their behavior.

The proposed analytical model enhances the two-dimensional, two-node Multiple-Vertical-Line-Element Model (MVLEM), which has proven to be an efficient and accurate computational tool for the nonlinear analysis of planar, flexure-dominated reinforced concrete walls. This study offers a detailed set of equations to determine model parameters for a concentrated plastic hinge, enabling the simulation of monotonic and cyclic loading of RC beam columns.

Furthermore, this study investigates the behavior of elements that can dissipate earthquake energy and compares their response under two conditions: fixed foundation and soil-structure interaction (SSI) conditions in different soil types (type C and type D).

5.2 Major Conclusions

- **Impact of Elements on Earthquake Response**: The foundation, type of soil significantly influences the earthquake response and energy dissipation of a structure. Therefore, it is crucial to consider both soil and foundation in structural studies and design.
- **Effect of Soil-Structure Interaction (SSI) on Structural Period**: The structural period is influenced by both the soil type and the foundation dimensions. Under Soil-Structure Interaction (SSI) conditions, the period of the structure increases depending on these factors. For soil type D, the period ranges increased by approximately 1.3 to 1.5 times, varying with the number of stories and foundation dimensions. In contrast, for soil type C, the period ranges increased by about 1.03 to 1.07 times, also depending on the number of stories and foundation dimensions which leads to ignore the SSI condition for soil type of C.
- **Energy Dissipation in Structural Elements**: When structures are built on soil type D, without considering SSI, all earthquake energy in our study is dissipated by elements like beams, columns, and shear walls, leading to artificially high figures for moment, stress, and strain. In reality, soil dissipates some of the earthquake energy, which helps reduce the amount of stress, strain, and moment in structural elements. For example, In the 7-story building which is built on clay soil, the stress in the steel

reinforcement of the shear walls decreased from over 600 MN/m^2 to lower levels about 590MN/m^2 under the SSI condition, with strain reducing from approximately 0.0065 to around 0.0045. Similarly, moments and rotations in the beams and columns experienced significant reductions, such as a decrease from over 400KN.m to 380KN.m and rotation decreased from 0.0025 to less than 0.001. However, this phenomenon has not been observed in sandy soil, and the differences were minimal, so the SSI effects can be disregarded for soil type C.

- **Variability in Soil Response**: Different soils exhibit different responses during an earthquake. Clay soil dissipates more energy and allows for more significant settlement, sliding, and rocking movements of the structure compared to stiff soil. Consequently, internal forces, rotation, stress, and strain in elements such as beams, columns, and materials used in shear walls are significantly more affected by SSI conditions when the structure is built on clay soil compared to stiff soil so the for the stiff sand soil, the SSI condition can be ignored.

5.3 Scope of Future Work

This study focused solely on competent soils, like dense dry sand and medium clay. Future research could expand this model to assess the behavior of shallow foundations in less competent soils, such as liquefiable, unstable, or reduced-strength soils. This would offer a more comprehensive understanding of foundation behavior under various soil conditions and enhance the model's accuracy in predicting real-world scenarios.

Additionally, this study focused exclusively on reinforced concrete (RC) frames with shear walls. Future work could explore the performance of other structural systems, such as steel frames or hybrid RC-steel frames, to evaluate their behavior and energy dissipation mechanisms under seismic loading. Furthermore, the current research was conducted using numerical simulations. To validate and enhance the findings, future studies could incorporate experimental testing in a laboratory environment. Laboratory tests would provide empirical data to support the numerical models and could reveal practical challenges and nuances not captured in computer simulations. Combining numerical and experimental approaches would lead to more robust and reliable conclusions about the seismic performance of various structural systems and soil-foundation interactions.

REFERENCES

[1] S. ATC, "Evaluation and retrofit of concrete buildings, Rep," ATC-40, Applied Technology Council, Redwood City, California, 1996.

[2] P. M. Yesane, Y. Ghugal, and R. Wankhade, "Study on soil-structure interaction: a review," International journal of engineering research, vol. 5, no. 3, pp. 737-741, 2016.

[3] M. Lou, H. Wang, X. Chen, and Y. Zhai, "Structure–soil–structure interaction: Literature review," Soil dynamics and earthquake engineering, vol. 31, no. 12, pp. 1724-1731, 2011.

[4] A. K. Chopra and J. A. Gutierrez, "Earthquake response analysis of multistorey buildings including foundation interaction," Earthquake engineering & structural dynamics, vol. 3, no. 1, pp. 65-77, 1974.

[5] J. P. Wolf and P. Obernhuber, "Non-linear soil-structure-interaction analysis using dynamic stiffness or flexibility of soil in the time domain," Earthquake engineering & structural dynamics, vol. 13, no. 2, pp. 195-212, 1985.

[6] M. C. Genes and S. Kocak, "Dynamic soil–structure interaction analysis of layered unbounded media via a coupled finite element/boundary element/scaled boundary finite element model," International Journal for numerical methods in engineering, vol. 62, no. 6, pp. 798-823, 2005.

[7] M. C. Genes, "Dynamic analysis of large-scale SSI systems for layered unbounded media via a parallelized coupled finite-element/boundary-element/scaled boundary finite-element model," Engineering analysis with boundary elements, vol. 36, no. 5, pp. 845-857, 2012.

[8] P. Raychowdhury, Nonlinear Winkler-based shallow foundation model for performance assessment of seismically loaded structures. University of California, San Diego, 2008.

[9] G. W. Housner, "The behavior of inverted pendulum structures during earthquakes," Bulletin of the seismological society of America, vol. 53, no. 2, pp. 403-417, 1963.

[10] G. Pandey, D. Patel, V. K. Mourya, R. Kumar, and S. Kumar, "A review on soil-foundation-interaction models," Journal of rehabilitation in civil engineering, vol. 11, no. 3, pp. 158-179, 2023.

[11] A. K. Chopra and S. C.-S. Yim, "Simplified earthquake analysis of structures with foundation uplift," Journal of structural engineering, vol. 111, no. 4, pp. 906-930, 1985.

[12] D. K. Nakaki and G. C. Hart, Uplifting response of structures subjected to earthquake motions. EKEH, 1987.

[13] B. S. S. Council and A. T. Council, NEHRP guidelines for the seismic rehabilitation of buildings. Federal emergency management agency, 1997.

[14] J. Moehle and G. G. Deierlein, "A framework methodology for performance-based earthquake engineering," in 13th world conference on earthquake engineering, 2004, vol. 679: WCEE Vancouver, p. 12.

[15] D. G. Lignos and H. Krawinkler, "Development and utilization of structural component databases for performance-based earthquake engineering," Journal of structural engineering, vol. 139, no. 8, pp. 1382-1394, 2013.

[16] A. Ghobarah, "Performance-based design in earthquake engineering: state of development," Engineering structures, vol. 23, no. 8, pp. 878-884, 2001.

[17] F. E. FEMA 356, "Prestandard and commentary for the seismic rehabilitation of buildings," Federal emergency management agency: Washington, DC, USA, 2000.

[18] C. B. Haselton and P. E. E. R. Center, Beam-column element model calibrated for predicting flexural response leading to global collapse of RC frame buildings. Pacific earthquake engineering research center, 2008.

[19] L. F. Ibarra, Global collapse of frame structures under seismic excitations. Stanford University, 2004.

[20] R. Villaverde, "Methods to assess the seismic collapse capacity of building structures: State of the art," Journal of structural engineering, vol. 133, no. 1, pp. 57-66, 2007.

[21] H. Takizawa and P. Jennings, "Ultimate capacity of low-rise R/C buildings subjected to intense earthquake motions," In 6th world conference on earthquake engineering, 1977, vol. 3, pp. 79-84.

[22] C.-K. Sun, G. V. Berg, and R. D. Hanson, "Gravity effect on single-degree inelastic system," Journal of the engineering mechanics division, vol. 99, no. 1, pp. 183-200, 1973.

[23] D. Bernal, "Amplification factors for inelastic dynamic p–Δ effects in earthquake analysis," Earthquake engineering & structural dynamics, vol. 15, no. 5, pp. 635-651, 1987.

[24] G. A. MacRae, "P-Δ effects on single-degree-of-freedom structures in earthquakes," Earthquake spectra, vol. 10, no. 3, pp. 539-568, 1994.

[25] L. F. Ibarra, R. A. Medina, and H. Krawinkler, "Hysteretic models that incorporate strength and stiffness deterioration," Earthquake engineering & structural dynamics, vol. 34, no. 12, pp. 1489-1511, 2005.

[26] R. S. Thyagarajan, Modeling and analysis of hysteretic structural behavior. California institute of technology, 1990.

[27] S. A. Mahin and V. V. Bertero, "Prediction of nonlinear seismic building behavior," Journal of the technical councils of ASCE, vol. 104, no. 1, pp. 21-37, 1978.

[28] Y.-K. Wen, "Method for random vibration of hysteretic systems," Journal of the engineering mechanics division, vol. 102, no. 2, pp. 249-263, 1976.

[29] D. N. Biskinis and M. H. N. Fardis, "Deformations of concrete members at yielding and ultimate under monotonic or cyclic loading (including repaired and retrofitted members)," 2009.

[30] D. Kent and R. Park, "Cyclic load behaviour of reinforcing steel," Strain, vol. 9, no. 3, pp. 98-103, 1973.

[31] G. Chang and J. B. Mander, Seismic energy based fatigue damage analysis of bridge columns: Part I-Evaluation of seismic capacity. National center for earthquake engineering research, Buffalo, NY, 1994.

[32] J. B. Mander, M. J. Priestley, and R. Park, "Theoretical stress-strain model for confined concrete," Journal of structural engineering, vol. 114, no. 8, pp. 1804-1826, 1988.

[33] A. S. o. C. Engineers, "Minimum design loads and associated criteria for buildings and other structures," 2017: American society of civil engineers.

[34] A. CODE, "318-19 (22); Building code requirements for structural concrete and commentary," Amarican concrete institute: Indianapolis, IN, USA, 2022.

[35] C. Haselton, A. Whittaker, A. Hortacsu, J. Baker, J. Bray, and D. Grant, "Selecting and scaling earthquake ground motions for performing response-

history analyses," In proceedings of the 15th world conference on earthquake engineering, 2012: Earthquake engineering research institute Oakland, CA, USA, pp. 4207-4217.

[36] S. Gajan, T. C. Hutchinson, B. L. Kutter, P. Raychowdhury, J. A. Ugalde, and J. P. Stewart, Numerical models for analysis and performance-based design of shallow foundations subjected to seismic loading. Pacific earthquake engineering research center Berkeley, 2008.

[37] P. Raychowdhury and T. Hutchinson, "Material model parameters for shallow foundation numerical analysis," In geotechnical earthquake engineering and soil dynamics IV, 2008, pp. 1-10.

[38] S. Mazzoni, F. McKenna, M. H. Scott, and G. L. Fenves, "OpenSees command language manual," Pacific earthquake engineering research (PEER) center, vol. 264, no. 1, pp. 137-158, 2006.

[39] R. W. Boulanger, C. J. Curras, B. L. Kutter, D. W. Wilson, and A. Abghari, "Seismic soil-pile-structure interaction experiments and analyses," Journal of geotechnical and geoenvironmental engineering, vol. 125, no. 9, pp. 750-759, 1999.

[40] G. G. Meyerhof, "Some recent research on the bearing capacity of foundations," Canadian geotechnical journal, vol. 1, no. 1, pp. 16-26, 1963.

[41] B. M. Das and G. Ramana, Principles of soil dynamics. PWS-Kent publishing

company Boston, MA, 1993.

[42] Y. Cai, Q. Chen, Y. Zhou, S. Nimbalkar, and J. Yu, "Estimation of passive earth pressure against rigid retaining wall considering arching effect in cohesive-frictional backfill under translation mode," International journal of geomechanics, vol. 17, no. 4, p. 04016093, 2017.

[43] A. PEER, "72–1,(2010) Modeling and acceptance criteria for seismic design and analysis of tall buildings, PEER/ATC 72–1 report," Applied technology council, Redwood city, CA, 2010.

[44] K. Kolozvari, K. Orakcal, and J. W. Wallace, "New opensees models for simulating nonlinear flexural and coupled shear-flexural behavior of RC walls and columns," Computers & structures, vol. 196, pp. 246-262, 2018.

[45] K. I. Kolozvari, Analytical modeling of cyclic shear-flexure interaction in reinforced concrete structural walls. University of California, Los Angeles, 2013.

[46] K. Orakcal, J. W. Wallace, and J. P. Conte, "Flexural modeling of reinforced concrete walls-model attributes," Structural journal, vol. 101, no. 5, pp. 688-698, 2004.

[47] K. Orakcal and J. W. Wallace, "Flexural modeling of reinforced concrete walls-experimental verification," ACI materials journal, vol. 103, no. 2, p. 196, 2006.

[48] K. Kolozvari, K. Kalbasi, K. Orakcal, and J. Wallace, "Three-dimensional model for nonlinear analysis of slender flanged reinforced concrete walls," Engineering structures, vol. 236, p. 112105, 2021.

[49] K. Kolozvari, K. Orakcal, and J. Wallace, "Shear-flexure interaction modeling for reinforced concrete structural walls and columns under reversed cyclic loading," Pacific Earthquake Engineering Research Center, University of California, Berkeley, PEER report, vol. 12, 2015.

[50] A. S. o. C. Engineers, "Seismic evaluation and retrofit of existing buildings," 2014: American society of civil engineers.

[51] C. B. Haselton, A. B. Liel, S. C. Taylor-Lange, and G. G. Deierlein, "Calibration of model to simulate response of reinforced concrete beam-columns to collapse," ACI structural journal, vol. 113, no. 6, 2016.

[52] C. W. Harden, Numerical modeling of the nonlinear cyclic response of shallow foundations. Pacific earthquake engineering research center, 2005.

[53] L. F. Ibarra, "Global collapse of frame structures under seismic excitations," Doctor of philosophy, deparment of civil and environmental, engineering, stanford university, 2003.

[54] M. Menegotto, "Method of analysis for cyclically loaded RC plane frames including changes in geometry and non-elastic behavior of elements under combined normal force and bending," In Proc. of IABSE symposium on

resistance and ultimate deformability of structures acted on by well defined repeated loads, 1973, 1973.

[55] F. C. Filippou, E. P. Popov, and V. V. Bertero, "Effects of bond deterioration on hysteretic behavior of reinforced concrete joints," 1983.

[56] T. B. Panagiotakos and M. N. Fardis, "Deformations of reinforced concrete members at yielding and ultimate," Structural journal, vol. 98, no. 2, pp. 135-148, 2001.

APPENDICES

Appendix A: Input File for 10 Story

```
#Developed by Amir Amirbeygloo- amirbeygloo.ir@gmail.com
#Eastern Mediterranean University
import math
import os
import openseespy.opensees as ops
import sys
import ReadRecord
import numpy as np
import logging
log_directory = r"E:\Essay\essay-2-18-2024\CLAY-D-E\10-story\uplift-10\Mexico"
# Path for the error log file
log_file_path = os.path.join(log_directory, 'error_log.txt')
# Redirect stderr to the specified log file
sys.stderr = open(log_file_path, 'w')
ops.model('basic', '-ndm', 3, '-ndf', 6)
#units('N', 'm', 'kg', 's')
#creat model
#In put geometry of structure
NoF=10  #Num of Floor =7
NoSX=3   #Num of Span_X=3
NoSY=3   #Num of Span_Y=3
LX=5     #LenSpan_X=5
LY=5     #LenSpan_Y=5
H=3      #Hight=3
#geometry of beam floor1-3
h_Beam=0.5  #height of beam
b_Beam=0.5  #width of beam
#making the main nodes
nodes={}
nodes_coordinate={}
 new_nodes={}
new_nodes_coordinate={}
    for k in range(1,NoF+2):
   for i in range(1, NoSX+2):
       for j in range(1, NoSY+2):
           tag_node=100*(k-1)+10*(i-1)+j
           X=(i-1)*LX
           Y=(j-1)*LY
           Z=(k-1)*H
           nodes_coordinate[(X, Y, Z)] = tag_node
           nodes[tag_node] = (X, Y, Z)

            ops.node(tag_node, X, Y, Z)
           #print(f'Node({tag_node}, [3], {Y}, {Z})')
           #print(f'Node({tag_node}, [3], {Y}, {Z})')
 print('main nodes are provided')

# nodes_column={}

for k in range (1, 11):
   for i in range(1, NoSX+2):
       for j in range(1, NoSY+2):
           if (i==2 and j==1) or (i==3 and j==1) or (i==2 and j==4) or (i==3 and j==4) or (i==1 and
j==2) or (i==1 and j==3) or (i==4 and j==2) or (i==4 and j==3):
               continue
           tag_node=10000+100*(k-1)+10*(i-1)+j
           X=(i-1)*LX
```

```
        Y=(j-1)*LY
        Z=((k-1)*H)+(h_Beam/2)
        nodes[tag_node] = (X, Y, Z)
        nodes_coordinate[(X, Y, Z)] = tag_node
        ops.node(tag_node, X, Y, Z)
        #print(f'Node({tag_node}, [3], {Y}, {Z})')
        new_tag_node=tag_node+10000
        new_nodes[new_tag_node]=(X, Y, Z)
        new_nodes_coordinate[(X, Y, Z)]=new_tag_node
        ops.node(new_tag_node, X, Y, Z)
        #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 6)
        #print(f'Node({new_tag_node}, [3], {Y}, {Z})')

print('hing nodes for columns are provided')

for k in range (2, 12):
    for i in range(1, NoSX+2):
        for j in range(1, NoSY+2):
            if (i==2 and j==1) or (i==3 and j==1) or (i==2 and j==4) or (i==3 and j==4) or (i==1 and
j==2) or (i==1 and j==3) or (i==4 and j==2) or (i==4 and j==3):
                continue
            tag_node=30000+100*(k-1)+10*(i-1)+j
            X=(i-1)*LX
            Y=(j-1)*LY
            Z=((k-1)*H)-(h_Beam/2)
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            #print(f'Node({tag_node}, [3], {Y}, {Z})')
            new_tag_node=tag_node+30000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 6)
            #print(f'Node({new_tag_node}, [3], {Y}, {Z})'

print('hing nodes for columns are provided')

h_Column=0.5
b_Column=0.5

for k in range(2,NoF+2):
    for i in range(1, NoSX+1):
        for j in range(2, 4):
            tag_node=50000+100*(k-1)+10*(i-1)+j
            X=(i-1)*LX+(b_Column/2)
            Y=(j-1)*LY
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            #print(f'Node({tag_node}, [3], {Y}, {Z})')
            new_tag_node=tag_node+50000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)
            #print(f'Node({new_tag_node}, [3], {Y}, {Z})')
print('hing nodes for beams are provided')
```

```
for k in range(2,NoF+2):
    for i in range(2, 4):
        for j in range(1, NoSY+1):
            tag_node=70000+100*(k-1)+10*(i-1)+j
            X=(i-1)*LX
            Y=(j-1)*LY+(b_Column/2)
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            #print(f'Node({tag_node}, [3], {Y}, {Z})')
            new_tag_node=tag_node+70000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)

            #print(f'Node({tag_node}, [3], {Y}, {Z})'

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(2, 4):
        for j in range(2, NoSY+2):
            tag_node=90000+100*(k-1)+10*(i-1)+j
            X=(i-1)*LX
            Y=(j-1)*LY-(b_Column/2)
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            #print(f'Node({tag_node}, [3], {Y}, {Z})')
            new_tag_node=tag_node+90000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)
print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(1):
        for j in range(2, NoSY+2):
             if j==3 :
                 continue
             tag_node=110000+100*(k-1)+10*(i-1)+j
             X=0
             Y=(j-1)*LY-(b_Column/2)
             Z=(k-1)*H
             nodes[tag_node] = (X, Y, Z)
             nodes_coordinate[(X, Y, Z)] = tag_node
             ops.node(tag_node, X, Y, Z)
             #print(f'Node({tag_node}, [3], {Y}, {Z})')
             new_tag_node=tag_node+110000
             new_nodes[new_tag_node]=(X, Y, Z)
             new_nodes_coordinate[(X, Y, Z)]=new_tag_node
             ops.node(new_tag_node, X, Y, Z)
             #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)
             #print(f'Node({new_tag_node}, [3], {Y}, {Z})')
print('hing nodes for beams are provided')
```

```
for k in range(2,NoF+2):
    for i in range(1):
        for j in range(2, NoSY+2):
            if j==3 :
                continue
            tag_node=130000+100*(k-1)+10*(i-1)+j
            X=15
            Y=(j-1)*LY-(b_Column/2)
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            new_tag_node=tag_node+130000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(1):
        for j in range(1, NoSY+1):
            if j==2 :
                continue
            tag_node=150000+100*(k-1)+10*(i-1)+j
            X=15
            Y=(j-1)*LY+(b_Column/2)
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            new_tag_node=tag_node+150000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)
print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(1, NoSX+1):
            if i==2 :
                continue
            for j in range(1):
                tag_node=170000+100*(k-1)+10*(i-1)+j
                X=(i-1)*LX+(b_Column/2)
                Y=0
                Z=(k-1)*H
                nodes[tag_node] = (X, Y, Z)
                nodes_coordinate[(X, Y, Z)] = tag_node
                ops.node(tag_node, X, Y, Z)
                new_tag_node=tag_node+170000
                new_nodes[new_tag_node]=(X, Y, Z)
                new_nodes_coordinate[(X, Y, Z)]=new_tag_node
                ops.node(new_tag_node, X, Y, Z)
                #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
```

```
    for i in range(2, NoSX+2):
        for j in range(2, 4):
            tag_node=190000+100*(k-1)+10*(i-1)+j

            X=(i-1)*LX-(b_Column/2)
            Y=(j-1)*LY
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            new_tag_node=tag_node+190000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(2, NoSX+2):
            if i==3 :
                continue
            for j in range(1):
                tag_node=210000+100*(k-1)+10*(i-1)+j
                X=(i-1)*LX-(b_Column/2)
                Y=0
                Z=(k-1)*H
                nodes[tag_node] = (X, Y, Z)
                nodes_coordinate[(X, Y, Z)] = tag_node
                ops.node(tag_node, X, Y, Z)
                new_tag_node=tag_node+210000
                new_nodes[new_tag_node]=(X, Y, Z)
                new_nodes_coordinate[(X, Y, Z)]=new_tag_node
                ops.node(new_tag_node, X, Y, Z)
                #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)
print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(2, NoSX+2):
            if i==3 :
                continue
            for j in range(1):
                tag_node=230000+100*(k-1)+10*(i-1)+j
                X=(i-1)*LX-(b_Column/2)
                Y=15
                Z=(k-1)*H
                nodes[tag_node] = (X, Y, Z)
                nodes_coordinate[(X, Y, Z)] = tag_node
                ops.node(tag_node, X, Y, Z)
                new_tag_node=tag_node+230000
                new_nodes[new_tag_node]=(X, Y, Z)
                new_nodes_coordinate[(X, Y, Z)]=new_tag_node
                ops.node(new_tag_node, X, Y, Z)
                #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(1, NoSX+1):
            if i==2 :
```

```
                continue
            for j in range(1):
                tag_node=250000+100*(k-1)+10*(i-1)+j
                X=(i-1)*LX+(b_Column/2)
                Y=15
                Z=(k-1)*H
                nodes[tag_node] = (X, Y, Z)
                nodes_coordinate[(X, Y, Z)] = tag_node
                ops.node(tag_node, X, Y, Z)
                new_tag_node=tag_node+250000
                new_nodes[new_tag_node]=(X, Y, Z)
                new_nodes_coordinate[(X, Y, Z)]=new_tag_node
                ops.node(new_tag_node, X, Y, Z)
                #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 4, 6)

print('hing nodes for beams are provided')

for k in range(2,NoF+2):
    for i in range(1):
        for j in range(1, NoSY+1):
            if j==2 :
                continue
            tag_node=270000+100*(k-1)+10*(i-1)+j
            X=0
            Y=(j-1)*LY+(b_Column/2)
            Z=(k-1)*H
            nodes[tag_node] = (X, Y, Z)
            nodes_coordinate[(X, Y, Z)] = tag_node
            ops.node(tag_node, X, Y, Z)
            new_tag_node=tag_node+270000
            new_nodes[new_tag_node]=(X, Y, Z)
            new_nodes_coordinate[(X, Y, Z)]=new_tag_node
            ops.node(new_tag_node, X, Y, Z)
            #ops.equalDOF(tag_node, new_tag_node, 1, 2, 3, 5, 6)
print('hing nodes for beams are provided')



ops.uniaxialMaterial('Elastic', 15, 27789380.66e9)

#   Beam spring
#first floor
ele_tag=7000

My_Top_Beam_out=250074.9606182435

My_B0t_Beam_out=-346243.3099650788


K0_beam_out=67543633.5486111

as_Plus_out=0.005924931987441154

as_Neg_out=0.008203412519101836

Lamda_S_out=7.506813365977206

Lamda_C_out=7.506813365977206

Lamda_A_out=7.506813365977206
```

```
Lamda_K_out=7.506813365977206

c_S=1.0

c_C=1.0

c_A=1.0

c_K=1.0

theta_p_Plus_out= 0.08123548392185677

theta_p_Neg_out=0.06982133620480849

theta_pc_Plus_out=0.1

theta_pc_Neg_out=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_out=0.184937904823425

theta_u_Neg_out= 0.17494755301865356

D_Plus=1.0

D_Neg=1.0

nFac=10


ops.uniaxialMaterial('ModIMKPeakOriented', 20, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(2, 3):
            if Z1 == Z2 == (k-1) * H:
                if (Y1 == Y2 == 0) or (Y1 == Y2 == 15):
                    if (X1 == X2 == 0.25) or (X1 == X2 == 4.75) or (X1 == X2 == 10.25) or (X1 == X2 ==
14.75):
                        #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 15,
20, 15, '-dir',1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(2, 3):
            if Z1 == Z2 == (k-1) * H:
                if (X1 == X2 == 0) or (X1 == X2 == 15):
                    if (Y1 == Y2 == 0.25) or (Y1 == Y2 == 4.75) or (Y1 == Y2 == 10.25) or (Y1 == Y2 ==
14.75):
                         #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
```

```
                ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 20,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                ele_tag += 1



#second  floor
ele_tag=7016

My_Top_Beam_out=392495.5323857101

My_B0t_Beam_out=-486842.3792376529


K0_beam_out=67543633.5486111

as_Plus_out= 0.008874879200725093

as_Neg_out= 0.011008194868526024

Lamda_S_out=8.068914271735995

Lamda_C_out=8.068914271735995

Lamda_A_out=8.068914271735995

Lamda_K_out=8.068914271735995

c_S=1.0

c_C=1.0

c_A=1.0

c_K=1.0

theta_p_Plus_out= 0.08511991858281368

theta_p_Neg_out=0.07698776175713055

theta_pc_Plus_out=0.1

theta_pc_Neg_out=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_out=0.1909309108520293

theta_u_Neg_out=0.18419558214907428

D_Plus=1.0

D_Neg=1.0

nFac=10
```

```
ops.uniaxialMaterial('ModIMKPeakOriented', 21, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(3, 4):
            if Z1 == Z2 == (k-1) * H:
                if (Y1 == Y2 == 0) or (Y1 == Y2 == 15):
                    if (X1 == X2 == 0.25) or (X1 == X2 == 4.75) or (X1 == X2 == 10.25) or (X1 == X2 ==
14.75):
                        #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 15,
21, 15, '-dir',1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(3, 4):
            if Z1 == Z2 == (k-1) * H:
                if (X1 == X2 == 0) or (X1 == X2 == 15):
                    if (Y1 == Y2 == 0.25) or (Y1 == Y2 == 4.75) or (Y1 == Y2 == 10.25) or (Y1 == Y2 ==
14.75):
                            #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 21,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1



#third  floor
ele_tag=7032

My_Top_Beam_out= 486265.1696565817

My_B0t_Beam_out=-579708.5011715502


K0_beam_out=67543633.5486111

as_Plus_out= 0.008950616538115588

as_Neg_out= 0.010670615173891155

Lamda_S_out=10.00350095665368

Lamda_C_out=10.00350095665368

Lamda_A_out=10.00350095665368

Lamda_K_out=10.00350095665368

c_S=1.0

c_C=1.0
c_A=1.0
```

```
c_K=1.0

theta_p_Plus_out=0.10456326692250668

theta_p_Neg_out=0.09632689976304767

theta_pc_Plus_out=0.1

theta_pc_Neg_out=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_out=0.21176254158555272

theta_u_Neg_out= 0.20490962578347022

D_Plus=1.0

D_Neg=1.0

nFac=10

ops.uniaxialMaterial('ModIMKPeakOriented', 22, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(4, 5):
            if Z1 == Z2 == (k-1) * H:
                if (Y1 == Y2 == 0) or (Y1 == Y2 == 15):
                    if (X1 == X2 == 0.25) or (X1 == X2 == 4.75) or (X1 == X2 == 10.25) or (X1 == X2 ==
14.75):
                        #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 15,
22, 15, '-dir',1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(4, 5):
            if Z1 == Z2 == (k-1) * H:
                if (X1 == X2 == 0) or (X1 == X2 == 15):
                    if (Y1 == Y2 == 0.25) or (Y1 == Y2 == 4.75) or (Y1 == Y2 == 10.25) or (Y1 == Y2 ==
14.75):
                          #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 22,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1



#4th 5th, 6th, 7th floor
```

```
ele_tag=7048

My_Top_Beam_out= 439653.68591792067

My_B0t_Beam_out= -486708.3167766228


K0_beam_out= 67543633.5486111

as_Plus_out=0.008536711280769048

as_Neg_out=0.009450366302733082

Lamda_S_out= 9.648777164905113

Lamda_C_out= 9.648777164905113

Lamda_A_out= 9.648777164905113

Lamda_K_out= 9.648777164905113

c_S=1.0

c_C=1.0

c_A=1.0

c_K=1.0

theta_p_Plus_out= 0.09912405438795753

theta_p_Neg_out= 0.09453403309829342

theta_pc_Plus_out= 0.1

theta_pc_Neg_out= 0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_out=0.20563323464402863

theta_u_Neg_out= 0.2017398686627414

D_Plus= 1.0

D_Neg= 1.0

nFac=10

ops.uniaxialMaterial('ModIMKPeakOriented', 23, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(5, 9):
            if Z1 == Z2 == (k-1) * H:
```

```
            if (Y1 == Y2 == 0) or (Y1 == Y2 == 15):
              if (X1 == X2 == 0.25) or (X1 == X2 == 4.75) or (X1 == X2 == 10.25) or (X1 == X2 ==
14.75):
                #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 15,
23, 15, '-dir',1, 2, 3, 4, 5, 6)
                #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                ele_tag += 1
for tag_node1, (X1, Y1, Z1) in nodes.items():
  for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
    for k in range(5, 9):
      if Z1 == Z2 == (k-1) * H:
        if (X1 == X2 == 0) or (X1 == X2 == 15):
          if (Y1 == Y2 == 0.25) or (Y1 == Y2 == 4.75) or (Y1 == Y2 == 10.25) or (Y1 == Y2 ==
14.75):
                  #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 23,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                ele_tag += 1


#8th, 9th, 10th floor
ele_tag=7112

My_Top_Beam_out= 392704.4140506093

My_B0t_Beam_out= -440011.02358946844


K0_beam_out= 67543633.5486111

as_Plus_out=0.007789057977564333

as_Neg_out=0.008727356380221669

Lamda_S_out=9.419310552661296

Lamda_C_out=9.419310552661296

Lamda_A_out=9.419310552661296

Lamda_K_out=9.419310552661296

c_S=1.0

c_C=1.0

c_A=1.0

c_K=1.0

theta_p_Plus_out=0.09703754008599239

theta_p_Neg_out=0.09202824573474241

theta_pc_Plus_out=0.1
```

```
theta_pc_Neg_out=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_out= 0.20285162489930023

theta_u_Neg_out= 0.19854271646236166

D_Plus=1.0

D_Neg=1.0

nFac=10


ops.uniaxialMaterial('ModIMKPeakOriented', 24, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(9, 12):
            if Z1 == Z2 == (k-1) * H:
                if (Y1 == Y2 == 0) or (Y1 == Y2 == 15):
                    if (X1 == X2 == 0.25) or (X1 == X2 == 4.75) or (X1 == X2 == 10.25) or (X1 == X2 ==
14.75):
                        #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 15,
24, 15, '-dir',1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(9, 12):
            if Z1 == Z2 == (k-1) * H:
                if (X1 == X2 == 0) or (X1 == X2 == 15):
                    if (Y1 == Y2 == 0.25) or (Y1 == Y2 == 4.75) or (Y1 == Y2 == 10.25) or (Y1 == Y2 ==
14.75):
                            #print(f'node1: {tag_node1, X1, Y1, Z1}, node2: {new_tag_node2, X2, Y2, Z2}')
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 24,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1




#first floor

ele_tag=7160

My_Top_Beam_in=153216.07087046147

My_B0t_Beam_in= -251110.4756356856
```

```
K0_beam_in=67543633.5486111

as_Plus= 0.0036625759765066507

as_Neg= 0.006002707093892276

Lamda_S=7.154005945957951

Lamda_C=7.154005945957951

Lamda_A=7.154005945957951

Lamda_K=7.154005945957951


theta_p_Plus=0.08051496541779474

theta_p_Neg=0.0639800644753158

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.1

Res_Neg=0.1

theta_u_Plus=0.18278336678770613

theta_u_Neg=0.16769781642976112

D_Plus=1

D_Neg=1

nFac=10.0

ops.uniaxialMaterial('ModIMKPeakOriented', 26, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)



for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(2, 3):
            if Z1 == Z2 == (k-1) * H:
                for i in range(2, NoSX+1):
                    if ((X1 == X2 == (i-1) * LX and Y1 == Y2==0.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==4.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==5.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==9.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==10.25) or (X1 == X2 == (i-1) * LX and
Y1 == Y2==14.75)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 26,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
```

```
                    #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                    ele_tag += 1

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(2, 3):
            if Z1 == Z2 == (k-1) * H:
                for j in range(2, NoSY+1):
                    if ((X1==X2==0.25 and Y1==Y2==(j-1)*LY) or (X1==X2==4.75 and Y1==Y2==(j-
1)*LY)
                    or (X1==X2==5.25 and Y1==Y2==(j-1)*LY) or (X1==X2==9.75 and Y1==Y2==(j-
1)*LY)
                    or (X1==X2==10.25 and Y1==Y2==(j-1)*LY) or (X1==X2==14.75 and
Y1==Y2==(j-1)*LY)):
                    ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat', 15, 15, 15, 15,
26, 15, '-dir', 1, 2, 3, 4, 5, 6)
                    #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                    ele_tag += 1


#2th floor

ele_tag=7184
My_Top_Beam_in=250074.9606182435

My_B0t_Beam_in=-346243.3099650788


K0_beam_in=67543633.5486111

as_Plus=0.005924931987441154

as_Neg=0.008203412519101836

Lamda_S=7.506813365977206

Lamda_C=7.506813365977206

Lamda_A=7.506813365977206

Lamda_K=7.506813365977206



theta_p_Plus=0.08123548392185677

theta_p_Neg=0.06982133620480849

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus=0.184937904823425
```

```
theta_u_Neg=0.17494755301865356

D_Plus=1

D_Neg=1

nFac=10.0


ops.uniaxialMaterial('ModIMKPeakOriented', 27, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)



for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(3, 4):
            if Z1 == Z2 == (k-1) * H:
                for i in range(2, NoSX+1):
                    if ((X1 == X2 == (i-1) * LX and Y1 == Y2==0.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==4.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==5.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==9.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==10.25) or (X1 == X2 == (i-1) * LX and
Y1 == Y2==14.75)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 27,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(3, 4):
            if Z1 == Z2 == (k-1) * H:
                for j in range(2, NoSY+1):
                    if ((X1==X2==0.25 and Y1==Y2==(j-1)*LY) or (X1==X2==4.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==5.25 and Y1==Y2==(j-1)*LY) or (X1==X2==9.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==10.25 and Y1==Y2==(j-1)*LY) or (X1==X2==14.75 and
Y1==Y2==(j-1)*LY)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat', 15, 15, 15, 15,
27, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1



#3th,4,5,6,7floor

ele_tag=7208

My_Top_Beam_in=345303.37504443416
```

```
My_B0t_Beam_in=-440183.40684585145

K0_beam_in=67543633.5486111

as_Plus=0.007947471224822734

as_Neg=0.007969104681877904

Lamda_S=7.877019859545031

Lamda_C=7.877019859545031

Lamda_A=7.877019859545031

Lamda_K=7.877019859545031


theta_p_Plus=0.08362397187670247

theta_p_Neg=0.07468197251333325

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus=0.18873627272436

theta_u_Neg= 0.1798081893271783

D_Plus=1

D_Neg=1

nFac=10.0


ops.uniaxialMaterial('ModIMKPeakOriented', 28, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)


for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(4, 9):
            if Z1 == Z2 == (k-1) * H:
                for i in range(2, NoSX+1):
                    if ((X1 == X2 == (i-1) * LX and Y1 == Y2==0.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==4.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==5.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==9.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==10.25) or (X1 == X2 == (i-1) * LX and
Y1 == Y2==14.75)):
```

```
                ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 28,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                ele_tag += 1

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(4, 9):
            if Z1 == Z2 == (k-1) * H:
                for j in range(2, NoSY+1):
                    if ((X1==X2==0.25 and Y1==Y2==(j-1)*LY) or (X1==X2==4.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==5.25 and Y1==Y2==(j-1)*LY) or (X1==X2==9.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==10.25 and Y1==Y2==(j-1)*LY) or (X1==X2==14.75 and
Y1==Y2==(j-1)*LY)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat', 15, 15, 15, 15,
28, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1



#8, 9, 10 floor

ele_tag=7328

My_Top_Beam_in=250074.9606182435

My_B0t_Beam_in=-346243.3099650788


K0_beam_in=67543633.5486111

as_Plus=0.005924931987441154

as_Neg=0.008203412519101836

Lamda_S=7.506813365977206

Lamda_C=7.506813365977206

Lamda_A=7.506813365977206

Lamda_K=7.506813365977206


theta_p_Plus= 0.08123548392185677

theta_p_Neg= 0.06982133620480849

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0
```

```
Res_Neg=0.0

theta_u_Plus= 0.184937904823425

theta_u_Neg= 0.17494755301865356

D_Plus=1

D_Neg=1

nFac=10.0

ops.uniaxialMaterial('ModIMKPeakOriented', 29, K0_beam_out, as_Plus_out, as_Neg_out,
My_Top_Beam_out, My_B0t_Beam_out, Lamda_S_out, Lamda_C_out, Lamda_A_out,
Lamda_K_out, c_S, c_C, c_A, c_K, theta_p_Plus_out, theta_p_Neg_out, theta_pc_Plus_out,
theta_pc_Neg_out, Res_Pos, Res_Neg, theta_u_Plus_out, theta_u_Neg_out, D_Plus, D_Neg, nFac)


for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(9, 12):
            if Z1 == Z2 == (k-1) * H:
                for i in range(2, NoSX+1):
                    if ((X1 == X2 == (i-1) * LX and Y1 == Y2==0.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==4.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==5.25) or (X1 == X2 == (i-1) * LX and Y1
== Y2==9.75)
                        or (X1 == X2 == (i-1) * LX and Y1 == Y2==10.25) or (X1 == X2 == (i-1) * LX and
Y1 == Y2==14.75)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 29,
15, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        for k in range(9, 12):
            if Z1 == Z2 == (k-1) * H:
                for j in range(2, NoSY+1):
                    if ((X1==X2==0.25 and Y1==Y2==(j-1)*LY) or (X1==X2==4.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==5.25 and Y1==Y2==(j-1)*LY) or (X1==X2==9.75 and Y1==Y2==(j-
1)*LY)
                        or (X1==X2==10.25 and Y1==Y2==(j-1)*LY) or (X1==X2==14.75 and
Y1==Y2==(j-1)*LY)):
                        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat', 15, 15, 15, 15,
29, 15, '-dir', 1, 2, 3, 4, 5, 6)
                        #print(f'zerolength: {ele_tag}, node1: {tag_node1, X1, Y1, Z1}, node2:
{new_tag_node2, X2, Y2, Z2}')
                        ele_tag += 1
#Creating springs Hinges for columns
#uniaxialMaterial('ModIMKPeakOriented', matTag, K0, as_Plus, as_Neg, My_Plus, My_Neg,
Lamda_S, Lamda_C, Lamda_A, Lamda_K, c_S, c_C, c_A, c_K, theta_p_Plus, theta_p_Neg,
theta_pc_Plus, theta_pc_Neg, Res_Pos, Res_Neg, theta_u_Plus, theta_u_Neg, D_Plus, D_Neg)


#element('zeroLength', eleTag, *eleNodes, '-mat', *matTags, '-dir', *dirs, <'-doRayleigh', rFlag=0>, <'-
orient', *vecx, *vecyp>)
#for story 1  ,2,3,4
```

```
ele_tag=5000

My_Col1=705710.6148649589

My_Col1_Neg=-705710.6148649589

K01= 128766466.5017397

as_Plus=0.010228874471008785

as_Neg=0.010228874471008785

Lamda_S=5.741763013241428

Lamda_C=5.741763013241428

Lamda_A=5.741763013241428

Lamda_K=5.741763013241428


theta_p_Plus=0.06965292756825209

theta_p_Neg=0.06965292756825209

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus=0.17513347412697455

theta_u_Neg=0.17513347412697455

D_Plus=1

D_Neg=1

nFac=10

ops.uniaxialMaterial('ModIMKPeakOriented', 30, K01, as_Plus, as_Neg, My_Col1, My_Col1_Neg,
Lamda_S, Lamda_C, Lamda_A, Lamda_K, c_S, c_C, c_A, c_K, theta_p_Plus, theta_p_Neg,
theta_pc_Plus, theta_pc_Neg, Res_Pos, Res_Neg, theta_u_Plus, theta_u_Neg, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if (Z1==Z2==0.25 and X1==X2 and Y1==Y2) or (Z1==Z2==2.75 and X1==X2 and Y1==Y2 )
or (Z1==Z2==3.25 and X1==X2 and Y1==Y2) or (Z1==Z2==5.75 and X1==X2 and Y1==Y2) or
(Z1==Z2==6.25 and X1==X2 and Y1==Y2) or (Z1==Z2==8.75 and X1==X2 and Y1==Y2) or
(Z1==Z2==9.25 and X1==X2 and Y1==Y2) or (Z1==Z2==11.75 and X1==X2 and Y1==Y2):
            #print(f'node1:{tag_node1, X1, Y1, Z1}, node2:{new_tag_node2, X2, Y2, Z2} ')
```

```
        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 30, 30, 15,  '-dir',1, 2, 3, 4, 5, 6)
        #print(f'zerolength:{ele_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{new_tag_node2, X2, Y2, Z2}')
        ele_tag+=1



#for story5,6,7,8
My_Col2=519459.30618788296

My_Col2_Neg=-519459.30618788296

K02= 121578540.3875

as_Plus=0.0074440445849541705

as_Neg=0.0074440445849541705

Lamda_S=6.686938071870849

Lamda_C=6.686938071870849

Lamda_A=6.686938071870849

Lamda_K=6.686938071870849



theta_p_Plus=0.07461548954072746

theta_p_Neg=0.07461548954072746

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus=0.17888811285510714

theta_u_Neg=0.17888811285510714
D_Plus=1

D_Neg=1

nFac=10

ops.uniaxialMaterial('ModIMKPeakOriented', 31, K02, as_Plus, as_Neg, My_Col2, My_Col2_Neg, Lamda_S, Lamda_C, Lamda_A, Lamda_K, c_S, c_C, c_A, c_K, theta_p_Plus, theta_p_Neg, theta_pc_Plus, theta_pc_Neg, Res_Pos, Res_Neg, theta_u_Plus, theta_u_Neg, D_Plus, D_Neg, nFac)

for tag_node1, (X1, Y1, Z1) in nodes.items():
  for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
    if (Z1==Z2==12.25 and X1==X2 and Y1==Y2) or (Z1==Z2==14.75 and X1==X2 and Y1==Y2) or (Z1==Z2==15.25 and X1==X2 and Y1==Y2) or (Z1==Z2==17.75 and X1==X2 and Y1==Y2 ) or (Z1==Z2==18.25 and X1==X2 and Y1==Y2) or (Z1==Z2==20.75 and X1==X2 and Y1==Y2) or (Z1==Z2==21.25 and X1==X2 and Y1==Y2) or (Z1==Z2==23.75 and X1==X2 and Y1==Y2):
```

```
        ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat', 15, 15, 15, 31, 31, 15,  '-dir',1, 2, 3, 4, 5, 6 )
        #print(f'zerolength:{ele_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{new_tag_node2, X2, Y2, Z2}')
        ele_tag+=1

#for story9,10
My_Col3=333267.2810234917

My_Col3_Neg=-333267.2810234917

K03=121578540.3875

as_Plus=0.004491516132187603

as_Neg=0.004491516132187603

Lamda_S=7.688955589784392

Lamda_C=7.688955589784392

Lamda_A=7.688955589784392

Lamda_K=7.688955589784392



theta_p_Plus=0.07933889391615054

theta_p_Neg=0.07933889391615054

theta_pc_Plus=0.1

theta_pc_Neg=0.1

Res_Pos=0.0

Res_Neg=0.0

theta_u_Plus_3=0.18208006254641448

theta_u_Neg_3=0.18208006254641448

D_Plus=1

D_Neg=1

nFac=10

ops.uniaxialMaterial('ModIMKPeakOriented', 32, K03, as_Plus, as_Neg, My_Col3, My_Col3_Neg, Lamda_S, Lamda_C, Lamda_A, Lamda_K, c_S, c_C, c_A, c_K, theta_p_Plus, theta_p_Neg, theta_pc_Plus, theta_pc_Neg, Res_Pos, Res_Neg, theta_u_Plus, theta_u_Neg, D_Plus, D_Neg, nFac)


for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if (Z1==Z2==24.25 and X1==X2 and Y1==Y2) or (Z1==Z2==26.75 and X1==X2 and Y1==Y2) or (Z1==Z2==27.25 and X1==X2 and Y1==Y2) or (Z1==Z2==29.75 and X1==X2 and Y1==Y2):
            #print(f'node1:{tag_node1, X1, Y1, Z1}, node2:{new_tag_node2, X2, Y2, Z2} ')
```

```
            ops.element('zeroLength', ele_tag, tag_node1, new_tag_node2, '-mat',15, 15, 15, 32, 32, 15,  '-
dir',1, 2, 3, 4, 5, 6 )
            #print(f'zerolength:{ele_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{new_tag_node2, X2,
Y2, Z2}')
            ele_tag+=1

ops.section('Elastic', 1, 27789380.66e6, 0.25, 1.822e-3, 1.822e-3, 11578909.02e3, 8.805e-3)

ops.section('Elastic', 2, 27789380.66e3, 0.25, 1.822e-3, 1.822e-3, 11578909.02e3, 8.805e-3)#50*50-
20-floor1-7



#        *******creat column elements************

#geomTransf(transfType, transfTag, *transfArgs)
ops.geomTransf('PDelta', 1, 0, -1 , 0)

ops.geomTransf('Linear', 2, 0, -1, 0)

#element('elasticBeamColumn', eleTag, *eleNodes, secTag, transfTag, <'-mass', mass>, <'-cMass'> <'-
releasez', releaseCode>, <'-releasey', releaseCode>)


element_tag = 10  # Start element numbering
    #  section tag
   #  transformation tag
element_column={}
element_column_coordination={}

#paired_nodes = set()  # Correctly initialized set

for tag_node1, (X1, Y1, Z1) in nodes.items():
   for tag_node2, (X2, Y2, Z2) in nodes.items():
      if tag_node1 != tag_node2 and X1 == X2 and Y1 == Y2 and Z2-Z1==0.25:

            ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 1, 2, '-mass', 625)
            element_column[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
            element_column_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
            #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2,
Y2, Z2})')
            element_tag += 1

element_tag = 170
for new_tag_node1, (X1, Y1, Z1) in new_nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
       if new_tag_node1 != new_tag_node2 and X1 == X2 and Y1 == Y2 and Z2-Z1==2.5:

            ops.element('elasticBeamColumn', element_tag, new_tag_node1, new_tag_node2, 2, 1, '-
mass', 625)
            element_column[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
            element_column_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
            #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2,
Y2, Z2})')
            element_tag += 1

print('columns are provided')
```

```
ops.geomTransf('Linear', 3, 0, -1, 0)

#in direction y

ops.geomTransf('Linear', 4, -1, 0, 0)

#           *****section of beam*******
ops.section('Elastic', 3, 27789380.66e3, 0.25, 1.822e-3, 1.822e-3, 11578909.02e3, 8.805e-3)#50*50
ops.section('Elastic', 4, 27789380.66e6, 0.25, 1.822e-3, 1.822e-3, 11578909.02e3, 8.805e-3)


#section('Elastic', secTag, E_mod, A, Iz, Iy, G_mod, Jxx, alphaY=None, alphaZ=None)

#element('elasticBeamColumn', eleTag, *eleNodes, secTag, transfTag, <'-mass', mass>, <'-cMass'> <'-
releasez', releaseCode>, <'-releasey', releaseCode>)

#   *******************Creat Beam***************
#geomTransf('Linear', transfTag, *vecxz, '-jntOffset', *dI, *dJ) in X -direction

element_tag = 400

element_beam={}
element_beam_coordination={}
element_new_beam={}
element_new_beam_coordination={}
for tag_node1, (X1, Y1, Z1) in nodes.items():
   for tag_node2, (X2, Y2, Z2) in nodes.items():
      if tag_node1 != tag_node2 and Z1 == Z2:
         # Check for horizontal beams along X-axis
         if Y1 == Y2 and X2-X1==0.25:

              ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 3, '-mass',
1700)
              element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
              element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
              #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2,
Y2, Z2})')
              element_tag += 1

element_tag = 600
for new_tag_node1, (X1, Y1, Z1) in new_nodes.items():
   for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
      if new_tag_node1 != new_tag_node2 and Z1 == Z2:
         # Check for horizontal beams along X-axis
         if (Y1 == Y2 and X2-X1==4.5 and Y1 != 0) or (Y1 == Y2 and X2-X1==4.5 and Y1 != 15):

              ops.element('elasticBeamColumn', element_tag, new_tag_node1, new_tag_node2, 3, 3, '-
mass', 1700)
              element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
              element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
              element_new_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
              element_new_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
              #print(f'(element:{element_tag}, node1:{new_tag_node1, X1, Y1, Z1},
node2:{new_tag_node2, X2, Y2, Z2})')
              element_tag += 1

element_tag = 700

for tag_node1, (X1, Y1, Z1) in nodes.items():
   for tag_node2, (X2, Y2, Z2) in nodes.items():
```

```
        if tag_node1 != tag_node2 and Z1 == Z2:
            if X1 == X2 and Y2-Y1==0.25:

                ops.element('elasticBeamColumn', element_tag, tag_node2, tag_node1, 4, 4, '-mass',
1700)
                element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
                element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
                #print(f'(element:{element_tag}, node1:{new_tag_node1, X1, Y1, Z1},
node2:{new_tag_node2, X2, Y2, Z2})')
                element_tag += 1


element_tag = 900
for new_tag_node1, (X1, Y1, Z1) in new_nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if new_tag_node1 != new_tag_node2 and Z1 == Z2:
            # Check for horizontal beams along X-axis
            if X1 == X2 and Y2-Y1==4.5:

                ops.element('elasticBeamColumn', element_tag, new_tag_node2, new_tag_node1, 3, 4, '-
mass', 1700)
                element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
                element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
                element_new_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
                element_new_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
                #print(f'(element:{element_tag}, node1:{new_tag_node1, X1, Y1, Z1},
node2:{new_tag_node2, X2, Y2, Z2})')
                element_tag += 1


element_tag=1000

for k in range(2, NoF+1):
    X1=5
    Y1=0
    Z1=(((k-1)*H))
    node1=(X1, Y1, Z1)
    X2=10
    Y2=0
    Z2=((k-1)*H)
    node2=(X2, Y2, Z2)
    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
    tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
    ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 3, '-mass', 2350)
    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
    #print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')
    element_tag+=1

    X1=5
    Y1=15
    Z1=(k-1)*H
    node1=(X1, Y1, Z1)
    X2=10
    Y2=15
    Z2=((k-1)*H)
    node2=(X2, Y2, Z2)
    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
    tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
    ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 3, '-mass', 2350)
```

```
    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
    #print(f'(element:{element_tag}, node1:{tag_node1}, node2:{tag_node2})')
    element_tag+=1

    X1=0
    Y1=5
    Z1=((k-1)*H)
    node1=(X1, Y1, Z1)
    X2=0
    Y2=10
    Z2=((k-1)*H)
    node2=(X2, Y2, Z2)
    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
    tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
    ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 4, '-mass', 2350)
    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
    #print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')
    element_tag+=1

    X1=15
    Y1=5
    Z1=((k-1)*H)
    node1=(X1, Y1, Z1)
    X2=15
    Y2=10
    Z2=((k-1)*H)
    node2=(X2, Y2, Z2)
    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
    tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
    ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 4, '-mass', 2350)
    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
    #print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')
    element_tag+=1

element_tag = 1036
X1=5
Y1=0
Z1=30
node1=(X1, Y1, Z1)
X2=10
Y2=0
Z2=30
node2=(X2, Y2, Z2)
tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 3, '-mass', 1700)
element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
#print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')

element_tag = 1037
X1=5
Y1=15
Z1=30
node1=(X1, Y1, Z1)
X2=10
Y2=15
```

```
Z2=30
node2=(X2, Y2, Z2)
tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 3, '-mass', 1700)
element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
#print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')

element_tag=1038
X1=0
Y1=5
Z1=30
node1=(X1, Y1, Z1)
X2=0
Y2=10
Z2=30
node2=(X2, Y2, Z2)
tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 4, '-mass', 1700)
element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
#print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')

element_tag=1039
X1=15
Y1=5
Z1=30
node1=(X1, Y1, Z1)
X2=15
Y2=10
Z2=30
node2=(X2, Y2, Z2)
tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
tag_node2 =nodes_coordinate[(X2, Y2, Z2)]
ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 4, 4, '-mass', 1700)
element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
#print(f'(element:{element_tag}, node1:{X1, Y1, Z1}, node2:{X2, Y2, Z2})')

print('beams are provided')

#        *********Moddeling Shear wall*******
#Material for Modeling Concrete wall
#uniaxialMaterial('ConcreteCM', matTag, fpcc, epcc, Ec, rc, xcrn, ft, et, rt, xcrp, mon, '-GapClose', GapClose=0)

ops.uniaxialMaterial('ConcreteCM', 2000, -53000e3, -0.005, 34440000.66e3, 13, 1.03, 1822341.13 , 0.00008, 1.2, 10000, 1)

#uncinfuned concrete
ops.uniaxialMaterial('ConcreteCM', 1500, -34470e3, -0.002, 27789380.66e3, 15, 1.015, 1822341.13 , 0.00008, 1.2, 10000, 1)

fyYbp = 413000e3  # N/m²
bybp = 0.02  # dimensionless
```

```
fyYbn = 413000e3  # N/m²
bybn = 0.02  # dimensionless
Es = 200000000e3  # N/m²
R0 = 20.0  # dimensionless
a1 = 0.925  # dimensionless
a2 = 0.0015  # dimensionless
ops.uniaxialMaterial("SteelMPF", 3000, fyYbp, fyYbn, Es, bybp, bybn, R0, a1, a2)

Kcrack=0.1
Ashweb = 1.0  # m²
G = 11578909.02e3 # N/m² (converted from 1875 ksi)
GAs = Kcrack*G * Ashweb  # Shear stiffness in N/m

# Build shear material
ops.uniaxialMaterial("Elastic", 3500, GAs)

rho_web = 0.0025
m = 5
thick1 = 0.2
width = 1

element_tag = 3000
element_shearwall = {}

# Loop through floors
for k in range(1, NoF + 1):
    if k in range(1, 2):
        rho_bound1 = 0.056
        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
                    X1=(i-1)*LX
                    Y1=(j - 1) * LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1 = (X1, Y1, Z1)
                    X2=((i - 1) * LX)+LX
                    Y2=(j - 1) *LY
                    Z2=(k-1)*H
                    #print(X2, Y2, Z2)
                    node2 = (X2 , Y2, Z2)
                    X3=(i - 1) * LX + LX
                    Y3=(j - 1) * LY
                    Z3=((k-1)*H)+H
                    #print(X3,Y3, Z3)
                    node3 = (X3 , Y3, Z3)
                    X4=(i - 1) * LX
                    Y4=(j - 1) * LY
                    Z4=((k-1)*H)+H
                    #print(X4, Y4, Z4)
                    node4 = (X4, Y4, Z4)
                    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                    tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                    #print(tag_node2)
                    tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                    #print(tag_node3)
                    tag_node4=nodes_coordinate[(X4, Y4, Z4)]
```

```
            ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
            '-thick', thick1, thick1, thick1, thick1, thick1,
            '-width', width, width, width, width, width,
            '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
            '-matConcrete', 2000, 1500, 1500, 1500, 2000,
            '-matSteel', 3000, 3000, 3000, 3000, 3000,
            '-matShear', 3500, 3500, 3500, 3500, 3500,
             '-CoR', 0.4)
            element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
            #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
            element_tag += 1
            #print(tag_node4)

  elif k in range(2, 3):
    rho_bound1 = 0.045
    for i in range(1, NoSX + 2):
      for j in range(1, NoSY + 2):
        if (i == 2 and j == 1) or (i == 2 and j == 4):
            X1=(i-1)*LX
            Y1=(j - 1) * LY
            Z1=(k-1)*H
            #print(X1, Y1, Z1)
            node1 = (X1, Y1, Z1)
            X2=((i - 1) * LX)+LX
            Y2=(j - 1) *LY
            Z2=(k-1)*H
            #print(X2, Y2, Z2)
            node2 = (X2 , Y2, Z2)
            X3=(i - 1) * LX + LX
            Y3=(j - 1) * LY
            Z3=((k-1)*H)+H
            #print(X3,Y3, Z3)
            node3 = (X3 , Y3, Z3)
            X4=(i - 1) * LX
            Y4=(j - 1) * LY
            Z4=((k-1)*H)+H
            #print(X4, Y4, Z4)
            node4 = (X4, Y4, Z4)
            tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
            #print(tag_node1)nodes_coordinate[(X, Y, Z)]

            tag_node2=nodes_coordinate[(X2, Y2, Z2)]
            #print(tag_node2)
            tag_node3=nodes_coordinate[(X3, Y3, Z3)]
            #print(tag_node3)
            tag_node4=nodes_coordinate[(X4, Y4, Z4)]
            ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
            '-thick', thick1, thick1, thick1, thick1, thick1,
            '-width', width, width, width, width, width,
            '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
            '-matConcrete', 2000, 1500, 1500, 1500, 2000,
            '-matSteel', 3000, 3000, 3000, 3000, 3000,
            '-matShear', 3500, 3500, 3500, 3500, 3500,
             '-CoR', 0.4)
            element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
```

```
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(3, 4):
        rho_bound1 = 0.037
        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
                    X1=(i-1)*LX
                    Y1=(j - 1) * LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1 = (X1, Y1, Z1)
                    X2=((i - 1) * LX)+LX
                    Y2=(j - 1) *LY
                    Z2=(k-1)*H
                    #print(X2, Y2, Z2)
                    node2 = (X2 , Y2, Z2)
                    X3=(i - 1) * LX + LX
                    Y3=(j - 1) * LY
                    Z3=((k-1)*H)+H
                    #print(X3,Y3, Z3)
                    node3 = (X3 , Y3, Z3)
                    X4=(i - 1) * LX
                    Y4=(j - 1) * LY
                    Z4=((k-1)*H)+H
                    #print(X4, Y4, Z4)
                    node4 = (X4, Y4, Z4)
                    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                    tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                    #print(tag_node2)
                    tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                    #print(tag_node3)
                    tag_node4=nodes_coordinate[(X4, Y4, Z4)]
                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
                    '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                    #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                    element_tag += 1
                    #print(tag_node4)
    elif k in range(4, 5):

        rho_bound1 = 0.062


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
```

```
                X1=(i-1)*LX
                Y1=(j - 1) * LY
                Z1=(k-1)*H
                #print(X1, Y1, Z1)
                node1 = (X1, Y1, Z1)
                X2=((i - 1) * LX)+LX
                Y2=(j - 1) *LY
                Z2=(k-1)*H
                #print(X2, Y2, Z2)
                node2 = (X2 , Y2, Z2)
                X3=(i - 1) * LX + LX
                Y3=(j - 1) * LY
                Z3=((k-1)*H)+H
                #print(X3,Y3, Z3)
                node3 = (X3 , Y3, Z3)
                X4=(i - 1) * LX
                Y4=(j - 1) * LY
                Z4=((k-1)*H)+H
                #print(X4, Y4, Z4)
                node4 = (X4, Y4, Z4)
                tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                #print(tag_node2)
                tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                #print(tag_node3)
                tag_node4=nodes_coordinate[(X4, Y4, Z4)]
                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(5, 6):

        rho_bound1 = 0.053


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
                X1=(i-1)*LX
                Y1=(j - 1) * LY
                Z1=(k-1)*H
                #print(X1, Y1, Z1)
                node1 = (X1, Y1, Z1)
                X2=((i - 1) * LX)+LX
                Y2=(j - 1) *LY
                Z2=(k-1)*H
                #print(X2, Y2, Z2)
```

```
                    node2 = (X2 , Y2, Z2)
                    X3=(i - 1) * LX + LX
                    Y3=(j - 1) * LY
                    Z3=((k-1)*H)+H
                    #print(X3,Y3, Z3)
                    node3 = (X3 , Y3, Z3)
                    X4=(i - 1) * LX
                    Y4=(j - 1) * LY
                    Z4=((k-1)*H)+H
                    #print(X4, Y4, Z4)
                    node4 = (X4, Y4, Z4)
                    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                    tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                    #print(tag_node2)
                    tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                    #print(tag_node3)
                    tag_node4=nodes_coordinate[(X4, Y4, Z4)]
                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                    '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                    #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                    element_tag += 1
                    #print(tag_node4)
    elif k in range(6, 7):

        rho_bound1 = 0.032


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
                    X1=(i-1)*LX
                    Y1=(j - 1) * LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1 = (X1, Y1, Z1)
                    X2=((i - 1) * LX)+LX
                    Y2=(j - 1) *LY
                    Z2=(k-1)*H
                    #print(X2, Y2, Z2)
                    node2 = (X2 , Y2, Z2)
                    X3=(i - 1) * LX + LX
                    Y3=(j - 1) * LY
                    Z3=((k-1)*H)+H
                    #print(X3,Y3, Z3)
                    node3 = (X3 , Y3, Z3)
                    X4=(i - 1) * LX
                    Y4=(j - 1) * LY
                    Z4=((k-1)*H)+H
```

```
                #print(X4, Y4, Z4)
                node4 = (X4, Y4, Z4)
                tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                #print(tag_node2)
                tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                #print(tag_node3)
                tag_node4=nodes_coordinate[(X4, Y4, Z4)]
                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(7, NoF+1):

        rho_bound1 = 0.019


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 2 and j == 1) or (i == 2 and j == 4):
                    X1=(i-1)*LX
                    Y1=(j - 1) * LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1 = (X1, Y1, Z1)
                    X2=((i - 1) * LX)+LX
                    Y2=(j - 1) *LY
                    Z2=(k-1)*H
                    #print(X2, Y2, Z2)
                    node2 = (X2 , Y2, Z2)
                    X3=(i - 1) * LX + LX
                    Y3=(j - 1) * LY
                    Z3=((k-1)*H)+H
                    #print(X3,Y3, Z3)
                    node3 = (X3 , Y3, Z3)
                    X4=(i - 1) * LX
                    Y4=(j - 1) * LY
                    Z4=((k-1)*H)+H
                    #print(X4, Y4, Z4)
                    node4 = (X4, Y4, Z4)
                    tag_node1 =nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)nodes_coordinate[(X, Y, Z)]

                    tag_node2=nodes_coordinate[(X2, Y2, Z2)]
                    #print(tag_node2)
                    tag_node3=nodes_coordinate[(X3, Y3, Z3)]
                    #print(tag_node3)
```

```
                tag_node4=nodes_coordinate[(X4, Y4, Z4)]
                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)


element_tag=3020
# Loop through floors
for k in range(1, NoF + 1):
    if k in range(1, 2):
        rho_bound1 = 0.056
        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
                    Z2=(k-1)*H
                    node2=(X2, Y2, Z2)
                    X3= (i-1)*LX
                    Y3=(j-1)*LY-LY
                    Z3=((k-1)*H)+H
                    node3=(X3, Y3, Z3)
                    X4=(i-1)*LX
                    Y4=(j-1)*LY
                    Z4=((k-1)*H)+H
                    node4=(X4, Y4, Z4)
                    tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)
                    tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                    tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                    tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
                    '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
```

```
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)

    elif k in range(2, 3):
        rho_bound1 = 0.045
        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
                    Z2=(k-1)*H
                    node2=(X2, Y2, Z2)
                    X3= (i-1)*LX
                    Y3=(j-1)*LY-LY
                    Z3=((k-1)*H)+H
                    node3=(X3, Y3, Z3)
                    X4=(i-1)*LX
                    Y4=(j-1)*LY
                    Z4=((k-1)*H)+H
                    node4=(X4, Y4, Z4)
                    tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)
                    tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                    tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                    tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
                    '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                    #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                    element_tag += 1
                    #print(tag_node4)
    elif k in range(3, 4):
        rho_bound1 = 0.037
        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
```

```
                Z2=(k-1)*H
                node2=(X2, Y2, Z2)
                X3= (i-1)*LX
                Y3=(j-1)*LY-LY
                Z3=((k-1)*H)+H
                node3=(X3, Y3, Z3)
                X4=(i-1)*LX
                Y4=(j-1)*LY
                Z4=((k-1)*H)+H
                node4=(X4, Y4, Z4)
                tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                #print(tag_node1)
                tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_bound1, rho_web, rho_bound1, rho_bound1,
                '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(4, 5):

        rho_bound1 = 0.062


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
                    Z2=(k-1)*H
                    node2=(X2, Y2, Z2)
                    X3= (i-1)*LX
                    Y3=(j-1)*LY-LY
                    Z3=((k-1)*H)+H
                    node3=(X3, Y3, Z3)
                    X4=(i-1)*LX
                    Y4=(j-1)*LY
                    Z4=((k-1)*H)+H
                    node4=(X4, Y4, Z4)
                    tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)
                    tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                    tag_node3= nodes_coordinate[(X3, Y3, Z3)]
```

```
                tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                '-matConcrete', 2000, 1500, 1500, 1500, 2000,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(5, 6):

        rho_bound1 = 0.053


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
                    Z2=(k-1)*H
                    node2=(X2, Y2, Z2)
                    X3= (i-1)*LX
                    Y3=(j-1)*LY-LY
                    Z3=((k-1)*H)+H
                    node3=(X3, Y3, Z3)
                    X4=(i-1)*LX
                    Y4=(j-1)*LY
                    Z4=((k-1)*H)+H
                    node4=(X4, Y4, Z4)
                    tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)
                    tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                    tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                    tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                    '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
```

```
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
    elif k in range(6, 7):

        rho_bound1 = 0.032


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
                    Z1=(k-1)*H
                    #print(X1, Y1, Z1)
                    node1=(X1, Y1, Z1)
                    X2= (i-1)*LX
                    Y2=(j-1)*LY-LY
                    Z2=(k-1)*H
                    node2=(X2, Y2, Z2)
                    X3= (i-1)*LX
                    Y3=(j-1)*LY-LY
                    Z3=((k-1)*H)+H
                    node3=(X3, Y3, Z3)
                    X4=(i-1)*LX
                    Y4=(j-1)*LY
                    Z4=((k-1)*H)+H
                    node4=(X4, Y4, Z4)
                    tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                    #print(tag_node1)
                    tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                    tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                    tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                    ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                    '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3500, 3500, 3500, 3500, 3500,
                     '-CoR', 0.4)
                    element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                    #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                    element_tag += 1
                    #print(tag_node4)
    elif k in range(7, NoF+1):

        rho_bound1 = 0.019


        for i in range(1, NoSX + 2):
            for j in range(1, NoSY + 2):
                if (i == 1 and j == 3) or (i == 4 and j == 3):
                    X1=(i-1)*LX
                    Y1=(j-1)*LY
```

```
                Z1=(k-1)*H
                #print(X1, Y1, Z1)
                node1=(X1, Y1, Z1)
                X2= (i-1)*LX
                Y2=(j-1)*LY-LY
                Z2=(k-1)*H
                node2=(X2, Y2, Z2)
                X3= (i-1)*LX
                Y3=(j-1)*LY-LY
                Z3=((k-1)*H)+H
                node3=(X3, Y3, Z3)
                X4=(i-1)*LX
                Y4=(j-1)*LY
                Z4=((k-1)*H)+H
                node4=(X4, Y4, Z4)
                tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
                #print(tag_node1)
                tag_node2= nodes_coordinate[(X2, Y2, Z2)]
                tag_node3= nodes_coordinate[(X3, Y3, Z3)]
                tag_node4= nodes_coordinate[(X4, Y4, Z4)]

                ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4,
m,
                '-thick', thick1, thick1, thick1, thick1, thick1,
                '-width', width, width, width, width, width,
                '-rho', rho_bound1, rho_web, rho_web, rho_web, rho_bound1,
                '-matConcrete', 1500, 1500, 1500, 1500, 1500,
                '-matSteel', 3000, 3000, 3000, 3000, 3000,
                '-matShear', 3500, 3500, 3500, 3500, 3500,
                 '-CoR', 0.4)
                element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4,
Z4))]=element_tag
                #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
                element_tag += 1
                #print(tag_node4)
print('shear walls are provided')

ops.node(130, 7.5, 7.5, 3)
ops.node(230, 7.5, 7.5, 6)
ops.node(330, 7.5, 7.5, 9)
ops.node(430, 7.5, 7.5, 12)
ops.node(530, 7.5, 7.5, 15)
ops.node(630, 7.5, 7.5, 18)
ops.node(730, 7.5, 7.5, 21)
ops.node(830, 7.5, 7.5, 24)
ops.node(930, 7.5, 7.5, 27)
ops.node(1030, 7.5, 7.5, 30)
ops.fix(130, 0, 0, 1, 1, 1, 0)
ops.fix(230, 0, 0, 1, 1, 1, 0)
ops.fix(330, 0, 0, 1, 1, 1, 0)
ops.fix(430, 0, 0, 1, 1, 1, 0)
ops.fix(530, 0, 0, 1, 1, 1, 0)
ops.fix(630, 0, 0, 1, 1, 1, 0)
ops.fix(730, 0, 0, 1, 1, 1, 0)
ops.fix(830, 0, 0, 1, 1, 1, 0)
ops.fix(930, 0, 0, 1, 1, 1, 0)
ops.fix(1030, 0, 0, 1, 1, 1, 0)
```

```
ops.rigidDiaphragm(3, 130, 101, 102, 103, 104, 111, 112, 113, 114, 121, 122, 123, 124, 131, 132,
133, 134)

ops.rigidDiaphragm(3, 230, 201, 202, 203, 204, 211, 212, 213, 214, 221, 222, 223, 224, 231, 232,
233, 234)

ops.rigidDiaphragm(3, 330, 301, 302, 303, 304, 311, 312, 313, 314, 321, 322, 323, 324, 331, 332,
333, 334)
ops.rigidDiaphragm(3, 430, 401, 402, 403, 404, 411, 412, 413, 414, 421, 422, 423, 424, 431, 432,
433, 434)

ops.rigidDiaphragm(3, 530, 501, 502, 503, 504, 511, 512, 513, 514, 521, 522, 523, 524, 531, 532,
533, 534)

ops.rigidDiaphragm(3, 630, 601, 602, 603, 604, 611, 612, 613, 614, 621, 622, 623, 624, 631, 632,
633, 634)

ops.rigidDiaphragm(3, 730, 701, 702, 703, 704, 711, 712, 713, 714, 721, 722, 723, 724, 731, 732,
733, 734)
ops.rigidDiaphragm(3, 830, 801, 802, 803, 804, 811, 812, 813, 814, 821, 822, 823, 824, 831, 832,
833, 834)
ops.rigidDiaphragm(3, 930, 901, 902, 903, 904, 911, 912, 913, 914, 921, 922, 923, 924, 931, 932,
933, 934)
ops.rigidDiaphragm(3, 1030, 1001, 1002, 1003, 1004, 1011, 1012, 1013, 1014, 1021, 1022, 1023,
1024, 1031, 1032, 1033, 1034)

print('floors fix')


# for tag_node, (X, Y, Z) in nodes.items():
#     if Z==0:
#         ops.fix(tag_node, 1, 1, 1, 1, 1, 1)
#         print(f'node:{tag_node}')

# ops.fix(1, 1, 1, 1, 1, 1, 1)
# ops.fix(2, 1, 1, 1, 1, 1, 1)
# ops.fix(3, 1, 1, 1, 1, 1, 1)
# ops.fix(4, 1, 1, 1, 1, 1, 1)
# ops.fix(11, 1, 1, 1, 1, 1, 1)
# ops.fix(12, 1, 1, 1, 1, 1, 1)
# ops.fix(13, 1, 1, 1, 1, 1, 1)
# ops.fix(14, 1, 1, 1, 1, 1, 1)
# ops.fix(21, 1, 1, 1, 1, 1, 1)
# ops.fix(22, 1, 1, 1, 1, 1, 1)
# ops.fix(23, 1, 1, 1, 1, 1, 1)
# ops.fix(24, 1, 1, 1, 1, 1, 1)
# ops.fix(31, 1, 1, 1, 1, 1, 1)
# ops.fix(32, 1, 1, 1, 1, 1, 1)
# ops.fix(33, 1, 1, 1, 1, 1, 1)
# ops.fix(34, 1, 1, 1, 1, 1, 1)


# print('foundation is fixed')

# Set constants
PI = 3.143

#----------------------------------------------
# Assign Soil Properties and Footing Dimension
#----------------------------------------------
```

```
soilType = 1  #---soiltype clay 1, sand 2
cd = 0.0
#---ratio of maximum drag to ultimate resistance
phi = 0.0001
#---friction angle (in deg; cohesionless soil; but given a very small value to avoid solution failure)
gamma = 16200.0  #---unit wt (N/m^3)
beta = 0.0     #---angle of load applied

c = 80.0 * 10**3     #---cohesion

Gmax = 4 * 10**6  #---provided by Christine

neu = 0.3        #---Poisson's ratio

crad = 0.05
#---radiation damping (default value=5%)
damping = 0.05  #---rayleigh damping (default value=5%)

tp = 0.0     #---uplift capacity (default value=10%)




#----------------------------------------------
# Assign Soil Properties and Footing Dimension
#----------------------------------------------
soilType = 2 #---soiltype clay 1, sand 2
cd = 0.0
#---ratio of maximum drag to ultimate resistance
phi = 42
#---friction angle (in deg; cohesionless soil; but given a very small value to avoid solution failure)
gamma = 16200.0  #---unit wt (N/m^3)
beta = 0.0     #---angle of load applied

c = 0    #---cohesion

Gmax = 26 * 10**6  #---provided by Christine

neu = 0.32        #---Poisson's ratio

crad = 0.05
#---radiation damping (default value=5%)
damping = 0.05  #---rayleigh damping (default value=5%)

tp = 0.0     #---uplift capacity (default value=10%)




stripL = 17.4  # length, width, and height of the strip footing in meters
stripB = 2.4
stripH = 1.5
Df = 1.5  #---depth of embedment
```

```
# Assign FEM mesh properties
#-------------------------------
ndive = 8  # no of divisions at end region; should always be even
ndivm = 30
ratioe = 20.0  # End length ratio (Lend/L)


L = stripL
Le = 1.2
Lm = L - 2 * Le
ratiom = (Lm / L) * 100.0
le = Le / ndive
lm = Lm / ndivm
nom = ndivm - 1
noe = ndive + 1
not_footing = nom + 2 * noe
elet = not_footing - 1

# Nodes at the footing level starts from 1
# Nodes at the spring level starts from 1001

# End portions



tag_node=650000
for i in range(-7, 2):

    X = round((i-1) * le, 2)
    Y=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650009
for i in range(2, 32):
    #if i==5 or i==15:
        #continue
    X = round( (i-1) * lm, 2)
    Y=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
```

```
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650039
for i in range(2, 10):

    X = round(15+(i-1) * le, 2)
    Y=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650047
for i in range(-7, 2):

    X = round((i-1) * le, 2)
    Y=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650056
for i in range(2, 32):
    #if i==5 or i==15:
        #continue
    X = round( (i-1) * lm, 2)
    Y=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
```

```
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650086
for i in range(2, 10):

    X = round(15+(i-1) * le, 2)
    Y=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650095
for i in range(-7, 2):

    X = round((i-1) * le, 2)
    Y=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650104
for i in range(2, 32):

    X = round((i-1) * lm, 2)
    Y=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
```

```
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650134
for i in range(2, 10):

    X = round(15+(i-1) * le, 2)
    Y=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650142
for i in range(-7, 2):

    X = round((i-1) * le, 2)
    Y=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650151
for i in range(2, 32):
    #if i==5 or i==15:
        #continue
    X = round((i-1) * lm, 2)
    Y=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
```

```
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650181
for i in range(2, 10):

    X = round(15+(i-1) * le, 2)
    Y=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650189
for i in range(-7, 1):

    Y = round((i-1) * le, 2)
    X=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650197
for i in range(2, 31):
    if i==11 or i==21:
        continue
    Y = round( (i-1) * lm, 2)
    X=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
```

```
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650224
for i in range(2, 10):

    Y = round(15+(i-1) * le, 2)
    X=0
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650232
for i in range(-7, 1):


    Y = round((i-1) * le, 2)
    X=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650240
for i in range(2, 31):
    if i==11 or i==21:
        continue
    Y = round( (i-1) * lm, 2)
    X=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
```

```
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650267
for i in range(2, 10):

    Y = round(15+(i-1) * le, 2)
    X=5
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')


tag_node=650276
for i in range(-7, 1):


    Y = round((i-1) * le, 2)
    X=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650284
for i in range(2, 31):
    if i==11 or i==21:
        continue

    Y = round((i-1) * lm, 2)
    X=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
```

```
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')

tag_node=650311
for i in range(2, 10):

    Y = round(15+(i-1) * le, 2)
    X=10
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')

tag_node=650319
for i in range(-7, 1):

    Y = round((i-1) * le, 2)
    X=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650327
for i in range(2, 31):
    if i==11 or i==21:
        continue

    Y = round((i-1) * lm, 2)
    X=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
```

```
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})')



tag_node=650354
for i in range(2, 10):

    Y = round(15+(i-1) * le, 2)
    X=15
    Z=-0.75
    nodes_coordinate[(X, Y, Z)] = tag_node
    nodes[tag_node] = (X, Y, Z)

    ops.node(tag_node, X, Y, Z)
    new_tag_node=tag_node+10000
    new_nodes[new_tag_node]=(X, Y, Z)
    new_nodes_coordinate[(X, Y, Z)]=new_tag_node
    ops.node(new_tag_node, X, Y, Z)
    #print(f'Node({tag_node}, [3], {Y}, {Z})')
    tag_node+=1
    #print(f'Node({tag_node}, [3], {Y}, {Z})'
ops.section('Elastic', 10, 27789380.66e9, 0.16e3, 746.66e-3, 746.66e-3, 11578909.02e6, 360666.6e-5)



element_tag = 250
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for tag_node2, (X2, Y2, Z2) in nodes.items():
        if tag_node1 != tag_node2 and X1 == X2 and Y1 == Y2 and round(Z2-Z1, 2)==0.75:


            ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 10, 2, '-mass', 625)
            element_column[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
            element_column_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
            #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2, Y2,
Z2})')
            element_tag += 1


Kcrack=1
Ashweb = 1  # m²
G = 11578909.02e3 # N/m² (converted from 1875 ksi)
GAs = Kcrack*G * Ashweb  # Shear stiffness in N/m

# Build shear material
ops.uniaxialMaterial("Elastic", 3600, GAs)


element_tag =3040
for k in range(0, 1):

    for i in range(1, NoSX + 2):
        for j in range(1, NoSY + 2):
            if (i == 2 and j == 1) or (i == 2 and j == 4):
                X1 = (i - 1) * LX
                Y1 = (j - 1) * LY
```

```
            Z1 = (k - 1) * H/4
            node1 = (X1, Y1, Z1)

            X2 = ((i - 1) * LX) + LX
            Y2 = (j - 1) * LY
            Z2 = (k - 1) * H/4
            node2 = (X2, Y2, Z2)

            X3 = (i - 1) * LX + LX
            Y3 = (j - 1) * LY
            Z3 = ((k - 1) * H/4) + H/4
            node3 = (X3, Y3, Z3)

            X4 = (i - 1) * LX
            Y4 = (j - 1) * LY
            Z4 = ((k - 1) * H/4) + H/4
            node4 = (X4, Y4, Z4)

            tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
            tag_node2 = nodes_coordinate[(X2, Y2, Z2)]
            tag_node3 = nodes_coordinate[(X3, Y3, Z3)]
            tag_node4 = nodes_coordinate[(X4, Y4, Z4)]
            rho_bound1 = 0.1

            ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4, m,
                    '-thick', thick1, thick1, thick1, thick1, thick1,
                    '-width', width, width, width, width, width,
                    '-rho', rho_bound1, rho_bound1, rho_bound1, rho_bound1, rho_bound1,
                    '-matConcrete', 2000, 2000, 2000, 2000, 2000,
                    '-matSteel', 3000, 3000, 3000, 3000, 3000,
                    '-matShear', 3600, 3600, 3600, 3600, 3600,
                    '-CoR', 0.4)

            element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4, Z4))] =
element_tag

            #print(f'element: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}, node3:
{(X3, Y3, Z3)}, node4: {(X4, Y4, Z4)}')

            element_tag += 1


element_tag=3042
for k in range(0, 1):
   for i in range(1, NoSX + 2):
      for j in range(1, NoSY + 2):
         if (i == 1 and j == 3) or (i == 4 and j == 3):
            X1=(i-1)*LX
            Y1=(j-1)*LY
            Z1=(k-1)*H/4
            #print(X1, Y1, Z1)
            node1=(X1, Y1, Z1)
            X2= (i-1)*LX
            Y2=(j-1)*LY-LY
            Z2=(k-1)*H/4
            node2=(X2, Y2, Z2)
            X3= (i-1)*LX
            Y3=(j-1)*LY-LY
            Z3=((k-1)*H/4)+H/4
            node3=(X3, Y3, Z3)
```

```
            X4=(i-1)*LX
            Y4=(j-1)*LY
            Z4=((k-1)*H/4)+H/4
            node4=(X4, Y4, Z4)
            tag_node1 = nodes_coordinate[(X1, Y1, Z1)]
            #print(tag_node1)
            tag_node2= nodes_coordinate[(X2, Y2, Z2)]
            tag_node3= nodes_coordinate[(X3, Y3, Z3)]
            tag_node4= nodes_coordinate[(X4, Y4, Z4)]
            rho_bound1 = 0.1
            ops.element('MVLEM_3D', element_tag, tag_node1, tag_node2, tag_node3, tag_node4, m,
'-thick', thick1, thick1, thick1, thick1, thick1, '-width', width, width, width, width, width, '-rho',
rho_bound1, rho_bound1, rho_bound1, rho_bound1, rho_bound1, '-matConcrete', 2000, 2000, 2000,
2000, 2000, '-matSteel', 3000, 3000, 3000, 3000, 3000, '-matShear', 3600, 3600, 3600, 3600, 3600, '-
CoR', 0.4)
            element_shearwall[((X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4, Z4))]=element_tag
            #print(f'element:{element_tag}, node1:{(X1, Y1, Z1)}, node2:{X2, Y2, Z2}, node3:{X3,
Y3, Z3}, node4:{X4, Y4, Z4}')
            element_tag += 1


for new_tag_node1, (X1, Y1, Z1) in new_nodes.items():
   if Z1==-0.75:
       ops.fix(new_tag_node1, 1, 1, 1, 1, 1, 1)
       #print(f'Node({new_tag_node1}, {X1}, {Y1}, {Z1})')


# Constants
PI = 3.143
radphi = PI * phi / 180.0
theta = PI / 4.0 + radphi / 2.0
Nphi1 = math.tan(theta)**2
Nq = Nphi1 * math.exp(PI * math.tan(radphi))
Nc = 5.14
Ngamma = (Nq - 1.0) * math.tan(1.4 * radphi)

print(f"Nc = {Nc}, Nq = {Nq}, Ngamma = {Ngamma}")

B = stripB
L = stripL
H1 = stripH

# Shape factors
Fcs = 1 + 0.2 * (B / L) #* Nphi1
Fqs = 1 + 0.1 * (B / L) #* Nphi1
Fgammas = Fqs

# Depth factors
Fcd = 1.0 + 0.2 * (Df / B) #* math.sqrt(Nphi1)
Fqd = 1.0 #+ 0.1 * (Df / B) * math.sqrt(Nphi1)
Fgammad = Fqd

# Inclination factors
Fci = (1 - beta / 90)**2
Fqi = Fci
Fgammai = (1 - beta / radphi)**2

# Ultimate bearing capacity
q1 = c * Nc * Fcs * Fcd * Fci
q2 = gamma * Df * Nq * Fqs * Fqd * Fqi
```

```
q3 = 0.5 * gamma * B * Ngamma * Fgammas * Fgammad * Fgammai
qu = q1 + q2 + q3
Qult = (qu * L * B)
# FSv = Qult / w
print(Qult)

# Bearing capacity per unit length
qult = Qult / stripL
q1mid = qult * lm  # Capacity of each mid-spring
  # Capacity of each extreme end spring
q2end = qult * le  # Capacity of other end region springs

# Sliding Capacity
Qf = stripB * stripL * c+ stripH* stripL*c*2

Qfs = (stripB * stripH * c*2)+(stripB * stripL *c)
# print(f"kxp: {kxp}")
# print(f"KX: {KX}")

G = Gmax


#for sand



# Constants
PI = 3.143
radphi = PI * phi / 180.0
theta = PI / 4.0 + radphi / 2.0
Nphi1 = math.tan(theta)**2
Nq = Nphi1 * math.exp(PI * math.tan(radphi))
Nc = 5.14
Ngamma = (Nq - 1.0) * math.tan(1.4 * radphi)

print(f"Nc = {Nc}, Nq = {Nq}, Ngamma = {Ngamma}")

B = stripB
L = stripL
H1 = stripH

# Shape factors
Fcs = 1 + 0.2 * (B / L) * Nphi1
Fqs = 1 + 0.1 * (B / L) * Nphi1
Fgammas = Fqs

# Depth factors
Fcd = 1.0 + 0.2 * (Df / B) * math.sqrt(Nphi1)
Fqd = 1.0 + 0.1 * (Df / B) * math.sqrt(Nphi1)
Fgammad = Fqd

# Inclination factors
Fci = (1 - beta / 90)**2
Fqi = Fci
Fgammai = (1 - beta / radphi)**2

# Ultimate bearing capacity
q1 = c * Nc * Fcs * Fcd * Fci
q2 = gamma * Df * Nq * Fqs * Fqd * Fqi
q3 = 0.5 * gamma * B * Ngamma * Fgammas * Fgammad * Fgammai
```

```
qu = q1 + q2 + q3
Qult = (qu * L * B)
# FSv = Qult / w
print(Qult)

# Bearing capacity per unit length
qult = Qult / stripL
q1mid = qult * lm  # Capacity of each mid-spring
  # Capacity of each extreme end spring
q2end = qult * le  # Capacity of other end region springs

# Sliding Capacity

Wg=1200000


Qf = Wg * math.tan(0.7 * radphi)

Qfs = Wg * math.tan(0.7 * radphi)
# print(f"kxp: {kxp}")
# print(f"KX: {KX}")

G = Gmax



ex=(1 + 0.21 * (Df / B)**0.5)*((1 + 1.6 * (0.5*Df * Df * (B + L)) / (B * L ** 2)) ** 0.4)
print(ex)
kx = ex*(( G * B / (2. - neu)) * (1.2 + 3.4 * (L**0.65 / B**0.65)))


## For horizontal soil springs toward the shortside
w=3.4*((L/B)**0.65)
u=(0.4*(L/B))+0.8
ey=ex
Ky=((G*B)/(2-neu))*(w+u)*ey

print(f'{kx}, {Ky}')

# For vertical soil springs
a=0.8+1.55*((L/B)**0.75)

ez=(1 + (1 / 21) * (Df / B) * (2 + 2.6 * (B / L)))*((1 + 0.32 * (0.5*Df * (B + L)) / (B * L)) ** (2 / 3))
print(ez)
kz=ez*(G*B*a/(1-neu))

ratioK = 9.0
kmid = kz / (Lm + 2 * Le* ratioK)
kend = kmid * ratioK
kzm = kmid * lm  # Mid region intensity
kze1 = kend * le  # Stiffness of other end springs

#long side

exx=(1 + 2.5 * (0.5*Df / B))*(1 + (2 * 0.5*Df / B) * (0.5*Df / Df) ** -0.2 * math.sqrt(B / L))
print(exx)
kxx= exx * G * (B**3) * (0.4 * (L/B) + 0.1) / (1 - neu)


kxxm=(kxx/L)*lm
```

```
kxxe2=(kxx/L)*le

#short side
eyy=(1 + 1.4 * (0.5*Df / L) ** 0.6)*(1.5 + 3.7 * (Df*0.5 / L) ** 1.9 * (0.5*Df / Df) ** -0.6)
print(eyy)

kyy=eyy*G*(B**3)*((0.47*(L/B)**2.4)+0.034)/(1-neu)


kyym=(kyy/L)*lm

kyye2=(kyy/L)*le

ops.uniaxialMaterial('Elastic', 16, kxxm)

ops.uniaxialMaterial('Elastic', 17, kxxe2)

ops.uniaxialMaterial('Elastic', 39, kyym)

ops.uniaxialMaterial('Elastic', 40, kyye2)


print(f"kz={kz}, kx={kx}, ky={Ky}, kxx={kxx}, kyy={kyy}")

print(stripL)

Kp=1
#Pult=gamma*(Df)*Kp* stripL+(2*c*Kp**0.5)* stripL
Pult=(gamma*((Df)**2)*Kp* L*0.5)*2#+(gamma*((Df)**2)*Kp* stripB*0.5)
#it has two side
print(Pult)

Psult=((gamma*((Df)**2)*Kp* B*0.5))*2#+(gamma*((Df)**2)*Kp* stripL*0.5)
# Define variables for QzSimple1, PySimple1, and TzSimple1 materials based on soil type

   # Qz Properties
zmid = 0.525 * q1mid / kzm

zend = 0.525 * q2end / kze1


# Py Properties
# Py Properties
y50 = 8.0 * Pult / kx

# Tz Properties
zt50 = 0.708 * Qf / kx

# Py Properties toward short
ys50 = 8.0 * Psult / Ky

# # Tz Properties
zst50 = 0.708 * Qfs / Ky


#for sand


Kp=11
```

```
   # Qz Properties
zmid = 1.39 * q1mid / kzm
z1mid = 1.39 * q1mid / kz1m
zend = 1.39 * q2end / kze1
z1end = 1.39 * q2end / kz1end

   # Qz Properties


# Py Properties
# Py Properties
y50 = 0.542 * Pult / kx

# Tz Properties
zt50 = 2.05 * Qf / kx

# Py Properties toward short
ys50 = 0.542 * Psult / Ky

# # Tz Properties
zst50 = 2.05 * Qfs / Ky



# Define QzSimple1 materials
ops.uniaxialMaterial("QzSimple1", 116, soilType, q1mid, zmid, tp, crad)

ops.uniaxialMaterial("QzSimple1", 118, soilType, q2end, zend, tp, crad)

# Define PySimple1 material toward longside
ops.uniaxialMaterial("PySimple1", 119, soilType, Pult, y50, cd, crad)

# Define PySimple1 material toward shortside
ops.uniaxialMaterial("PySimple1", 120, soilType, Psult, ys50, cd, crad)

# Define TzSimple1 material in longside
ops.uniaxialMaterial("TzSimple1", 121, soilType, Qf, zt50, crad)

#Define TzSimple1 material in shortside
ops.uniaxialMaterial("TzSimple1", 122, soilType, Qfs, zst50, crad)

# For clarity, print material parameters (optional)
print(f"zend={zend}, zmid={zmid}, y50={y50}, zt50={zt50}")

ops.section('Elastic', 5, 27789380.66e3, 3.6, 1.728, 0.675, 11578909.02e3, 165350692.3e-8)

element_tag = 3200


for tag_node1, (X1, Y1, Z1) in nodes.items():
   for tag_node2, (X2, Y2, Z2) in nodes.items():
      if tag_node1 != tag_node2 and Z1 == Z2==-0.75:
         # Check for horizontal beams along X-axis
         if Y1 == Y2 and (round(X2 - X1, 2) == 0.15 or round(X2 - X1, 2) == 0.5 ):

               ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 5, 3, '-mass',
8750)
```

```
                    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
                    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
                    #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2,
Y2, Z2})')
                    element_tag += 1

element_tag = 4000


for tag_node1, (X1, Y1, Z1) in nodes.items():
    for tag_node2, (X2, Y2, Z2) in nodes.items():
        if tag_node1 != tag_node2 and Z1 == Z2==-0.75:
            # Check for horizontal beams along X-axis
            if X1 == X2 and (round(Y2 - Y1, 2) == 0.15 or round(Y2 - Y1, 2) == 0.5 ):

                    ops.element('elasticBeamColumn', element_tag, tag_node1, tag_node2, 5, 4, '-mass',
8750)
                    element_beam[((X1, Y1, Z1), (X2, Y2, Z2))] = element_tag
                    element_beam_coordination[element_tag]=((X1, Y1, Z1), (X2, Y2, Z2))
                    #print(f'(element:{element_tag}, node1:{tag_node1, X1, Y1, Z1}, node2:{tag_node2, X2,
Y2, Z2})')
                    element_tag += 1


#vertical spring in end region
element_tag = 55000
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and X1 == X2 and ((-1.2 <= X1 <= 0 or 15 <= X1 <= 16.2) and (Y1 == Y2
== 0 or  Y1 == Y2 == 15)):
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  118, 17, 40, 15, '-
dir',  3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2:
({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1

element_tag = 55036
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and Y1 == Y2 and ((-1.2 <= Y1 <= 0 or 15 <= Y1 <= 16.2) and (X1 == X2
== 0 or  X1 == X2 == 15)):
            if (X1==0 and Y1==0) or (X1==15 and Y1==0) or (X1==0 and Y1==15) or (X1==15 and
Y1==15):
                continue
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  118, 40, 17, 15, '-
dir',   3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2:
({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1


#vertical spring for middle

element_tag = 55068
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and X1 == X2 and ((0 < X1 < 15 ) and (Y1 == Y2 == 0 or Y1 == Y2 ==
15)):
            if (X1==5 and Y1==0) or (X1==10 and Y1==0):
                continue
```

```
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  116, 16, 39, 15, '-dir',   3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2: ({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1


element_tag = 55124
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and Y1 == Y2 and ((0 < Y1 < 15) and (X1 == X2 == 0 or X1 == X2 == 15)):
            if  (X1==0 and Y1==5) or (X1==0 and Y1==10):
                continue
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  116, 39, 16, 15, '-dir', 3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2: ({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1


element_tag = 55180
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and X1 == X2 and ((0 <= X1 <15) and ( Y1 == Y2 == 5 or Y1 == Y2 == 10 )):
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  116, 16, 39, 15, '-dir',  3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2: ({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1


element_tag = 55240
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and X1 == X2 and ((-1.2 <= X1 <0 or 15<X1<=16.2) and ( Y1 == Y2 == 5 or Y1 == Y2 == 10 )):
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  118, 17, 40, 15, '-dir',  3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2: ({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1

element_tag = 55272
for tag_node1, (X1, Y1, Z1) in nodes.items():
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and Y1 == Y2 and ((0 <= Y1 < 15 ) and ( X1 == X2 == 5 or X1 == X2 == 10 )):
            if (X1==5 and Y1==5) or (X1==10 and Y1==5) or (X1==5 and Y1==10) or (X1==10 and Y1==10):
                continue
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  116, 39, 16, 15, '-dir',   3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2: ({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1

element_tag = 55328
for tag_node1, (X1, Y1, Z1) in nodes.items():
```

```
    for new_tag_node2, (X2, Y2, Z2) in new_nodes.items():
        if Z1 == Z2 == -0.75 and Y1 == Y2 and ((-1.2 <= Y1 < 0 or 15<Y1<=16.2) and ( X1 == X2 ==
5 or X1 == X2 == 10 )):
            if (X1==5 and Y1==5) or (X1==10 and Y1==5) or (X1==5 and Y1==10) or (X1==10 and
Y1==10):
                continue
            ops.element('zeroLength', element_tag, new_tag_node2, tag_node1,  '-mat',  118, 40, 17, 15, '-
dir',   3, 4, 5, 6)
            #print(f'zerolength: {element_tag}, node1: ({tag_node1}, {X1}, {Y1}, {Z1}), node2:
({new_tag_node2}, {X2}, {Y2}, {Z2})')
            element_tag += 1

# Nodes created for horizontal springs in X direction
ops.node(8001, -1.2, 0.0, -0.75)
ops.node(8002, -1.2, 0.0, -0.75)
new_nodes[8001]=(-1.2, 0.0, -0.75)
new_nodes_coordinate[(-1.2, 0.0, -0.75)]=8001
new_nodes[8002]=(-1.2, 0.0, -0.75)
new_nodes_coordinate[(-1.2, 0.0, -0.75)]=8002
ops.fix(8001, 1, 1, 1, 1, 1, 1)
ops.fix(8002, 1, 1, 1, 1, 1, 1)

ops.node(8003, -1.2, 5.0, -0.75)
ops.node(8004, -1.2, 5.0, -0.75)
new_nodes[8003]=( -1.2, 5.0, -0.75)
new_nodes_coordinate[( -1.2, 5.0, -0.75)]=8003
new_nodes[8004]=( -1.2, 5.0, -0.75)
new_nodes_coordinate[( -1.2, 5.0, -0.75)]=8004
ops.fix(8003, 1, 1, 1, 1, 1, 1)
ops.fix(8004, 1, 1, 1, 1, 1, 1)

ops.node(8005, -1.2, 10.0, -0.75)
ops.node(8006, -1.2, 10.0, -0.75)
new_nodes[8005]=( -1.2, 10.0, -0.75)
new_nodes_coordinate[( -1.2, 10.0, -0.75)]=8005
new_nodes[8006]=( -1.2, 10.0, -0.75)
new_nodes_coordinate[( -1.2, 10.0, -0.75)]=8006
ops.fix(8005, 1, 1, 1, 1, 1, 1)
ops.fix(8006, 1, 1, 1, 1, 1, 1)


ops.node(8007, -1.2, 15.0, -0.75)
ops.node(8008, -1.2, 15.0, -0.75)
new_nodes[8007]=( -1.2, 15.0, -0.75)
new_nodes_coordinate[( -1.2, 15.0, -0.75)]=8007
new_nodes[8008]=( -1.2, 15.0, -0.75)
new_nodes_coordinate[( -1.2, 15.0, -0.75)]=8008
ops.fix(8007, 1, 1, 1, 1, 1, 1)
ops.fix(8008, 1, 1, 1, 1, 1, 1)

ops.node(8011, 5.0, -1.2, -0.75)
ops.node(8012, 5.0, -1.2, -0.75)
new_nodes[8011]=( 5.0, -1.2, -0.75)
new_nodes_coordinate[( 5.0, -1.2, -0.75)]=8011
new_nodes[8012]=( 5.0, -1.2, -0.75)
new_nodes_coordinate[( 5.0, -1.2, -0.75)]=8012
ops.fix(8011, 1, 1, 1, 1, 1, 1)
ops.fix(8012, 1, 1, 1, 1, 1, 1)

ops.node(8013, 10.0, -1.2, -0.75)
```

```
ops.node(8014, 10.0, -1.2, -0.75)
new_nodes[8013]=( 10.0, -1.2, -0.75)
new_nodes_coordinate[( 10.0, -1.2, -0.75)]=8013
new_nodes[8014]=( 10.0, -1.2, -0.75)
new_nodes_coordinate[( 10.0, -1.2, -0.75)]=8014
ops.fix(8013, 1, 1, 1, 1, 1, 1)
ops.fix(8014, 1, 1, 1, 1, 1, 1)


ops.node(8015, 15.0, -1.2, -0.75)
ops.node(8016, 15.0, -1.2, -0.75)
new_nodes[8015]=( 15.0, -1.2, -0.75)
new_nodes_coordinate[( 15.0, -1.2, -0.75)]=8015
new_nodes[8016]=( 15.0, -1.2, -0.75)
new_nodes_coordinate[( 15.0, -1.2, -0.75)]=8016
ops.fix(8015, 1, 1, 1, 1, 1, 1)
ops.fix(8016, 1, 1, 1, 1, 1, 1)

ops.node(8009, 0.0, -1.2, -0.75)
ops.node(8010, 0.0, -1.2, -0.75)
new_nodes[8009]=(0.0, -1.2, -0.75)
new_nodes_coordinate[(0.0, -1.2, -0.75)]=8009
new_nodes[8010]=(0.0, -1.2, -0.75)
new_nodes_coordinate[(0.0, -1.2, -0.75)]=8010
ops.fix(8009, 1, 1, 1, 1, 1, 1)
ops.fix(8010, 1, 1, 1, 1, 1, 1)


ops.element('zeroLength', 55464,  8001, 650000, '-mat', 119, 120,  15, '-dir', 1, 2,  6)
ops.element('zeroLength', 55465, 8002, 650000, '-mat', 121, 122,  15, '-dir', 1, 2,   6)

ops.element('zeroLength', 55466, 8009,  650189, '-mat',  120, 119,  15, '-dir', 1,  2,  6)
ops.element('zeroLength', 55467, 8010,  650189, '-mat',  122, 121,  15, '-dir', 1,  2,  6)


ops.element('zeroLength', 55468, 8003, 650047,  '-mat', 119,  120,  15, '-dir', 1, 2,   6)
ops.element('zeroLength', 55469, 8004, 650047, '-mat', 121,  122,  15, '-dir', 1, 2,   6)


ops.element('zeroLength', 55470, 8005, 650095,  '-mat',  119,   120,  15, '-dir', 1, 2,  6)
ops.element('zeroLength', 55471, 8006, 650095, '-mat', 121, 122,    15, '-dir', 1, 2,    6)



ops.element('zeroLength', 55472,  8007, 650142, '-mat', 119, 120,    15, '-dir', 1, 2,    6)
ops.element('zeroLength', 55473,  8008, 650142, '-mat', 121, 122,    15, '-dir', 1, 2,    6)

# Nodes created for horizontal springs in Y direction

ops.element('zeroLength', 55474, 8011, 650232,  '-mat',   120,  119,  15, '-dir', 1,  2,  6)
ops.element('zeroLength', 55475, 8012, 650232, '-mat',   122, 121,  15, '-dir', 1, 2,  6)


ops.element('zeroLength', 55476, 8013, 650276,  '-mat',  120,  119,  15, '-dir', 1,  2,  6)
ops.element('zeroLength', 55477, 8014, 650276, '-mat',   122,  121,  15, '-dir', 1,  2,  6)

ops.element('zeroLength', 55478,  8015, 650319, '-mat',    120,   119,  15, '-dir', 1,   2,  6)
ops.element('zeroLength', 55479,  8016, 650319, '-mat',   122, 121,  15, '-dir', 1,  2,  6)
```

```
eigenvalues = ops.eigen(6)
ops.modalProperties('-print', 'ModalReport-7story-fix.txt', '-return')

ops.timeSeries('Linear', 1)
ops.pattern('Plain', 1, 1)

for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
    if Z1 == Z2 and Z1 != 0 and Z1 != 30:

        if ((X1, Y1, Z1) == (5, 0, Z1) and (X2, Y2, Z2) == (10, 0, Z2)) or \
            ((X1, Y1, Z1) == (5, 15, Z1) and (X2, Y2, Z2) == (10, 15, Z2)):
            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -25000, 0, 0)
            #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')


ops.timeSeries('Linear', 2)
ops.pattern('Plain', 2, 2)

for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
    if Z1 == Z2 and Z1 ==30:

        if ((X1, Y1, Z1) == (5, 0, Z1) and (X2, Y2, Z2) == (10, 0, Z2)) or \
            ((X1, Y1, Z1) == (5, 15, Z1) and (X2, Y2, Z2) == (10, 15, Z2)):
            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)
            #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')

ops.timeSeries('Linear', 3)
ops.pattern('Plain', 3, 3)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
    if Z1 == Z2 and Z1 != 0 and Z1 != 30:
        if ((X1, Y1, Z1) == (0, 5, Z1) and (X2, Y2, Z2) == (0, 10, Z2)) or \
        ((X1, Y1, Z1) == (15, 5, Z1) and (X2, Y2, Z2) == (15, 10, Z2)):

            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -25000, 0, 0)
            #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')
            #print(f'element:{element_tag}')
ops.timeSeries('Linear', 4)
ops.pattern('Plain', 4, 4)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
    if Z1 == Z2 and Z1 ==30:
        if ((X1, Y1, Z1) == (0, 5, Z1) and (X2, Y2, Z2) == (0, 10, Z2)) or \
        ((X1, Y1, Z1) == (15, 5, Z1) and (X2, Y2, Z2) == (15, 10, Z2)):

            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)

ops.timeSeries('Linear', 5)
ops.pattern('Plain', 5, 5)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_new_beam.items():
    if (X2 - X1 == 4.5) and (Y1 == Y2) and (Z1 == Z2) and (Z1 != 0):
        ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)
        #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')


ops.timeSeries('Linear', 6)
ops.pattern('Plain', 6, 6)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_new_beam.items():
    if (X2==X1) and (Y2-Y1==4.5) and  (Z1==Z2) and (Z1 != 0):
            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)
            #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')
```

```
ops.timeSeries('Linear', 7)
ops.pattern('Plain', 7, 7)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_new_beam.items():
   if (X2 - X1 == 0.25) and (Y1 == Y2) and (Z1 == Z2) and (Z1 != 0):
      ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)


ops.timeSeries('Linear', 8)
ops.pattern('Plain', 8, 8)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_new_beam.items():
   if (X2==X1) and (Y2-Y1==0.25) and  (Z1==Z2) and (Z1 != 0):
         ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -18800, 0, 0)


ops.timeSeries('Linear', 9)
ops.pattern('Plain', 9, 9)

for k in range (1, NoF+1):
   for i in range(1, NoSX + 2):
      for j in range(1, NoSY + 2):
         if (i==2 and j==1) or (i==3 and j==1) or (i==2 and j==4) or (i==3 and j==4) or (i==1 and
j==2) or (i==1 and j==3) or (i==4 and j==2) or (i==4 and j==3):
            continue
         X1=(i-1)*LX
         Y1=(j-1)*LY
         Z1=(k-1)*H
         node1 = (X1, Y1, Z1)
         tag_node1 = nodes_coordinate[(X1, Y1, Z1)]

         ops.load(tag_node1, 0, 0, -18350, 0, 0, 0)
         #print(f'node:{tag_node1}')

ops.timeSeries('Linear', 10)
ops.pattern('Plain', 10, 10)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
   if tag_node1 != tag_node2 and Z1 == Z2==-0.75:
      # Check for horizontal beams along X-axis
      if Y1 == Y2 and (round(X2 - X1, 2) == 0.15 or round(X2 - X1, 2) == 0.5 ):
         ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -95000, 0, 0)
         #print(f'eleLoad: {element_tag}, node1: {(X1, Y1, Z1)}, node2: {(X2, Y2, Z2)}')

ops.timeSeries('Linear', 11)
ops.pattern('Plain', 11, 11)
for ((X1, Y1, Z1), (X2, Y2, Z2)), element_tag in element_beam.items():
      if tag_node1 != tag_node2 and Z1 == Z2==-0.75:
         # Check for horizontal beams along X-axis
         if X1 == X2 and (round(Y2 - Y1, 2) == 0.15 or round(Y2 - Y1, 2) == 0.5 ):
            ops.eleLoad('-ele', element_tag, '-type', '-beamUniform', -95000, 0, 0)






print("loading complete")


# Clear any existing model data
```

```
ops.wipeAnalysis()

# Recorder to capture node displacements
ops.recorder('Node', '-file', '701.out', '-time', '-node', 1001, '-dof', 1, 2, 3, 'disp')

ops.constraints('Transformation')
ops.numberer('RCM')
ops.system('SparseGeneral')  # Use UmfPack solver
ops.test('EnergyIncr', 1e-3, 10)
ops.algorithm('NewtonLineSearch')
ops.integrator('LoadControl', 0.1)  # Load control with smaller increments
  # Convergence test
  # Newton-Raphson algorithm
ops.analysis('Static')  # Static analysis

# Perform gravity analysis
result = ops.analyze(10)
if result != 0:
    print(f'Gravity analysis failed with error code: {result}')
    # Print node displacements for debugging


print("Gravity Analysis Completed")

ops.loadConst('-time', 0)

# Clear the analysis setup
ops.wipeAnalysis()

frequencies = [(val**0.5) for val in eigenvalues]

# Number of modes to consider for damping
nModeDamp = 6

# Damping ratio
zetaDamp = 0.05 # 5%

# Calculate natural frequencies (omega) for the modes
omegaI = 2 * np.pi * frequencies[0]
omegaJ = 2 * np.pi * frequencies[nModeDamp-1]

# Calculate Rayleigh damping coefficients
alphaM = 2 * zetaDamp * omegaI * omegaJ / (omegaI + omegaJ)
betaK = 0.0
betaK0 = 0
betaKc = 2 * zetaDamp / (omegaI + omegaJ)

# Print calculated values for debugging
print(f"omegaI: {omegaI}, omegaJ: {omegaJ}")
print(f"alphaM: {alphaM}, betaK: {betaK}, betaK0: {betaK0}, betaKc: {betaKc}")

# Apply Rayleigh damping
ops.rayleigh(alphaM, betaK, betaK0, betaKc)


# Ground motion records
recordX = 'RSN8161_SIERRA.MEX_E12090'
recordY = 'RSN8161_SIERRA.MEX_E12360'
recordZ = 'RSN8161_SIERRA.MEX_E12-UP'
```

```
# Perform the conversion from SMD record to OpenSees record for each direction
dtx, nPtsx = ReadRecord.ReadRecord(recordX + '.at2', recordX + '.dat')
dty, nPtsy = ReadRecord.ReadRecord(recordY + '.at2', recordY + '.dat')
dtz, nPtsz = ReadRecord.ReadRecord(recordZ + '.at2', recordZ + '.dat')

# Define time series for ground motions
ops.timeSeries('Path', 12, '-filePath', recordX + '.dat', '-dt', dtx, '-factor', 35.98)
ops.timeSeries('Path', 13, '-filePath', recordY + '.dat', '-dt', dty, '-factor', 35.98)
ops.timeSeries('Path', 14, '-filePath', recordZ + '.dat', '-dt', dtz, '-factor', 21.67)

# Define a multi-support excitation pattern
ops.pattern('MultipleSupport', 12)

# Define ground motions for each direction
ops.groundMotion(1, 'Plain', '-accel', 12)
ops.groundMotion(2, 'Plain', '-accel', 13)
ops.groundMotion(3, 'Plain', '-accel', 14)

# Apply the ground motions to the base nodes
# Apply the ground motions to the base nodes
for new_tag_node, (X, Y, Z) in new_nodes.items():
    if Z == -0.75:
        ops.imposedMotion(new_tag_node, 1, 1)  # Apply X direction ground motion
        ops.imposedMotion(new_tag_node, 2, 2)  # Apply Y direction ground motion
        ops.imposedMotion(new_tag_node, 3, 3)  # Apply Z direction ground motion


dtInput = max(dtx, dty, dtz)  # Use the maximum dt from the records
nSteps = max(nPtsx, nPtsy, nPtsz)  # Use the maximum number of steps from the records
Tmax = dtInput * nSteps  # Total time of analysis


nSteps=12000

# Convergence Test
Tol = 1.e-3 # tolerance
maxNumIter = 1000  # maximum number of iterations
printFlag = 0  # print convergence information flag
TestType = 'NormDispIncr'  # test type

# Algorithm
algorithmType = 'KrylovNewton'

# Newmark-integrator parameters
NewmarkGamma = 0.5  # gamma
NewmarkBeta = 0.25  # beta

ops.constraints('Transformation')
ops.numberer('RCM')
ops.system('UmfPack')
ops.test(TestType, Tol, maxNumIter, printFlag)
ops.algorithm(algorithmType)
ops.integrator('Newmark', NewmarkGamma, NewmarkBeta)
ops.analysis('Transient')

DtAnalysis = 0.005  # Time step for analysis, make sure this is defined

for step in range(nSteps):
    ok = ops.analyze(1, DtAnalysis)
    if ok != 0:
```

```
        print(f"Analysis failed at step [27]")
        break
    currentTime = ops.getTime()
    print(f"Step: [27], Time: {currentTime}")

if ok == 0:
    print("Ground Motion Done.")
else:
    print("Ground Motion Analysis Failed.")

# Get current analysis time
currentTime = ops.getTime()
print("The current time is:", currentTime)
```

Appendix B: Calculation My

```
import math

def CalculateMy(B, H, cover, P, fpc, fy, Es, nBarBot, DBarBot, nBarTop, DBarTop, nBarInt,
DBarInt, DBarSh, cUnitTomm, cUnitToMPa):

    # Assumptions
    epsilonCU = 0.003

    # Convert units to N and mm
    cUnitToN = cUnitToMPa * (cUnitTomm ** 2)

    DBarTop *= cUnitTomm
    DBarBot *= cUnitTomm
    DBarInt *= cUnitTomm
    DBarSh *= cUnitTomm
    B *= cUnitTomm
    H *= cUnitTomm
    cover *= cUnitTomm

    Es *= cUnitToMPa
    fpc *= cUnitToMPa
    fy *= cUnitToMPa
    P *= cUnitToN

    pi = math.pi
    d = H - cover - DBarSh - DBarBot / 2.0
    dp = cover + DBarSh + DBarTop / 2.0

    # Steel yield strain
    epsilonSy = fy / Es
    # Depth of compression block at balanced (mm)
    Cb = epsilonCU * d / (epsilonCU + epsilonSy)
    # Tension steel, without intermediate bars for now (mm^2)
    Ast = nBarBot * (DBarBot ** 2) * pi / 4.0
    Asc = nBarTop * (DBarTop ** 2) * pi / 4.0
    # Tension steel ratio, without intermediate bars for now
    rhost = Ast / (B * d)
    rhosc = Asc / (B * d)
    # Intermediate steel (mm^2)
    # Intermediate steel ratio
    Asw = nBarInt * pi / 4.0 * (DBarInt ** 2)
    rhosw = Asw / (B * d)
    rho = rhosw + rhost + rhosc
    # Depth of pressure block factor
    if fpc <= 30:
        beta1 = 0.85
    else:
        beta1 = max(0.85 - 0.05 / 7.0 * (fpc - 30), 0.65)

    # Depth of compression block at balanced (mm)
    C = (Ast * fy - Asc * fy + P) / (0.85 * B * beta1 * fpc)

    # Compute terms in Fardis Equation
    Ec = 4700.0 * (fpc ** 0.5)
    n = Es / Ec

    # Compute AF and BF
    if C <= Cb:
```

```
        # Tension Controlled
        AF = rhosw + rhosc + rhost + (P / (d * B * fy))
        BF = rhost + rhosc * (dp / d) + 0.5 * rhosw * (1.0 + dp / d) + P / (d * fy * B)
    else:
        # Compression Controlled
        AF = rhosw + rhosc + rhost - (P / (1.8 * n * d * B * fpc))
        BF = rhost + rhosc * (dp / d) + 0.5 * rhosw * (1.0 + dp / d)

    # Compression zone depth (normalized by d) at yield (mm)
    ky = ( ( (n * AF) ** 2 + 2.0 * n * BF ) ** 0.5 ) - n * AF

    if C <= Cb:
        phiy = fy / (Es * (1.0 - ky) * d)
    else:
        phiy = 1.8 * fpc / (Ec * ky * d)

    Term1 = Ec * (ky ** 2) / 2.0 * (0.5 * (1.0 + dp / d) - ky / 3.0)
    Term2 = Es / 2.0 * ((1.0 - ky) * rhost + (ky - dp / d) * rhosc + rhosw / 6.0 * (1.0 - dp / d)) * (1.0 -
dp / d)
    MyP = B * (d ** 3) * phiy * (Term1 + Term2)

    # Convert from N.mm to our units
    MyP = MyP / cUnitToN / cUnitTomm

    # Calculate MyN
    d = H - cover - DBarSh - DBarTop / 2.0
    dp = cover + DBarSh + DBarBot / 2.0

    # Depth of compression block at balanced (mm)
    Cb = epsilonCU * d / (epsilonCU + epsilonSy)
    # Tension steel, without intermediate bars for now (mm^2)
    Ast = nBarTop * (DBarTop ** 2) * pi / 4.0
    Asc = nBarBot * (DBarBot ** 2) * pi / 4.0
    # Tension steel ratio, without intermediate bars for now
    rhost = Ast / (B * d)
    rhosc = Asc / (B * d)
    rho = rhosw + rhost + rhosc

    # Depth of compression block at balanced (mm)
    C = (Ast * fy - Asc * fy + P) / (0.85 * B * beta1 * fpc)

    # Compute AF and BF
    if C <= Cb:
        # Tension Controlled
        AF = rhosw + rhosc + rhost + (P / (d * B * fy))
        BF = rhost + rhosc * (dp / d) + 0.5 * rhosw * (1.0 + dp / d) + P / (d * fy * B)
    else:
        # Compression Controlled
        AF = rhosw + rhosc + rhost - (P / (1.8 * n * d * B * fpc))
        BF = rhost + rhosc * (dp / d) + 0.5 * rhosw * (1.0 + dp / d)

    # Compression zone depth (normalized by d) at yield (mm)
    ky = ( ( (n * AF) ** 2 + 2.0 * n * BF ) ** 0.5 ) - n * AF

    if C <= Cb:
        phiy = fy / (Es * (1.0 - ky) * d)
    else:
        phiy = 1.8 * fpc / (Ec * ky * d)

    Term1 = Ec * (ky ** 2) / 2.0 * (0.5 * (1.0 + dp / d) - ky / 3.0)
```

```
    Term2 = Es / 2.0 * ((1.0 - ky) * rhost + (ky - dp / d) * rhosc + rhosw / 6.0 * (1.0 - dp / d)) * (1.0 -
dp / d)
    MyN = B * (d ** 3) * phiy * (Term1 + Term2)

    # Convert from N.mm to our units
    MyN = MyN / cUnitToN / cUnitTomm

    return MyP, -MyN

# Example usage
B = 0.5
H = 0.5
cover = 0.05
P =200000
fpc = 34470e3
fy = 413000e3
Es = 200000000e3
nBarBot = 4
DBarBot = 0.02
nBarTop = 4
DBarTop = 0.02
nBarInt = 0
DBarInt = 0.02
DBarSh = 0.012
cUnitTomm = 1000
cUnitToMPa = 1e-6

MyP, MyN = CalculateMy(B, H, cover, P, fpc, fy, Es, nBarBot, DBarBot, nBarTop, DBarTop,
nBarInt, DBarInt, DBarSh, cUnitTomm, cUnitToMPa)

print(f"MyP: {MyP}, MyN: {MyN}")
```

Appendix C: Calculation Properties of RC Hinges

```
import math
from openseespy.opensees import uniaxialMaterial

def ComputeHingeRC(matTag, B, H, L, cover, P, fpc, Ec, fyBot, fyTop, MyP, MyN, nBarBot,
DBarBot, nBarTop, DBarTop, nBarInt, DBarInt, nBarSh, DBarSh, SStirrup, nFac, cUnitToMPa):

    stiffnessType = 1  # 1: 40%  2: yield
    asl = 1            # 1: include bond-slip  0: don't include bond-slip
    epsilonCU = 0.003

    A = H * B
    Ig = B * H**3 / 12.0

    if stiffnessType == 1:
        EIstfoEIg = min(max(0.77 * (0.1 + P / fpc / A)**0.80 * (L / 2.0 / H)**0.43, 0.35), 0.8)
        crackFac = EIstfoEIg
    elif stiffnessType == 2:
        EIyoEIg = min(max(0.30 * (0.1 + P / fpc / A)**0.80 * (L / 2.0 / H)**0.72, 0.2), 0.6)
        crackFac = EIyoEIg

    pi = math.pi
    nu = P / (fpc * A)
    sn = (SStirrup / DBarBot) * math.sqrt(fyBot * cUnitToMPa / 100.0)

    Ash = nBarSh * (DBarSh**2) * pi / 4.0
    AsBot = nBarBot * (DBarBot**2) * pi / 4.0
    AsTop = nBarTop * (DBarTop**2) * pi / 4.0
    AsInt = nBarInt * (DBarInt**2) * pi / 4.0
    rhosh = Ash / SStirrup / B
    d = H - (cover + DBarSh + (DBarBot / 2.0))
    AsTotal = AsBot + AsTop + AsInt
    rhoTotal = AsTotal / B / d

    thetaCapPl = 0.12 * (1 + 0.55 * asl) * (0.16**nu) * (0.02 + 40 * rhosh)**0.43 * (0.54**(0.01 * fpc
* cUnitToMPa)) * (0.66**(0.1 * sn)) * (2.27**(10.0 * rhoTotal))

    rhoBot = AsBot / B / d
    rhoTop = AsTop / B / d
    compValue = max(0.01, rhoTop * fyTop / fpc)
    tensileValue = max(0.01, rhoBot * fyBot / fpc)
    nonSymmetricFacP = (compValue / tensileValue)**0.225
    nonSymmetricFacN = nonSymmetricFacP**-1.0
    thetaCapPlP = nonSymmetricFacP * thetaCapPl
    thetaCapPlN = nonSymmetricFacN * thetaCapPl

    thetaPc = min(0.76 * (0.031**nu) * (0.02 + 40.0 * rhosh)**1.02, 0.1)

    McoMy = 1.13

    lambdaPrime = 170.7 * (0.27**nu) * (0.1**(SStirrup / d))
    lambdaVal = lambdaPrime * thetaCapPl

    K0 = 6.0 * Ec * Ig * EIstfoEIg / L
    thetaYP = MyP / K0
    thetaYN = MyN / K0
    asP = (McoMy - 1.0) / (thetaCapPlP / thetaYP)
    asN = (McoMy - 1.0) / (thetaCapPlP / thetaYN)
    LambdaS = lambdaVal
```

```
   LambdaC = lambdaVal
    LambdaA = lambdaVal
    LambdaK = lambdaVal
    cS = 1.0
    cC = 1.0
    cA = 1.0
    cK = 1.0
    thetaPP = thetaCapPlP
    thetaPN = thetaCapPlN
    thetaPcP = thetaPc
    thetaPcN = thetaPc
    ResP = 0.0
    ResN = 0.0
    thetaUP = thetaYP + thetaPP + thetaPcP
    thetaUN = -1*thetaYN + thetaPN + thetaPcN
    DP = 1.0
    DN = 1.0

    uniaxialMaterial('ModIMKPeakOriented', matTag, K0, asP, asN, MyP, MyN, LambdaS, LambdaC,
LambdaA, LambdaK, cS, cC, cA, cK, thetaPP, thetaPN, thetaPcP, thetaPcN, ResP, ResN, thetaUP,
thetaUN, DP, DN, nFac)
    print(f"Crack Factor: {crackFac}")
    print(f"K0: {K0}")
    print(f"thetaYP: {thetaYP}")
    print(f"thetaYN: {thetaYN}")
    print(f"asP: {asP}")
    print(f"asN: {asN}")
    print(f"LambdaS: {LambdaS}")
    print(f"LambdaC: {LambdaC}")
    print(f"LambdaA: {LambdaA}")
    print(f"LambdaK: {LambdaK}")
    print(f"thetaPP: {thetaPP}")
    print(f"thetaPN: {thetaPN}")
    print(f"thetaPcP: {thetaPcP}")
    print(f"thetaPcN: {thetaPcN}")
    print(f"thetaUP: {thetaUP}")
    print(f"thetaUN: {thetaUN}")
    return crackFac
    #print(K0, asP, asN, MyP, MyN, LambdaS, LambdaC, LambdaA, LambdaK, cS, cC, cA, cK,
thetaPP, thetaPN, thetaPcP, thetaPcN, ResP, ResN, thetaUP, thetaUN, DP, DN, nFac)

# Example usage
matTag = 1
B = 0.500
H = 0.500
L = 2.50
cover =0.050
P = 200000
fpc = 34.47e6
Ec =  27789380.66e3
fyBot = 413000e3
fyTop = 413000e3
MyP= 241292.54176923467
MyN = -241292.54176923467

nBarBot = 4
DBarBot = 0.020
anBarTop = 4
DBarTop = 0.020
nBarInt =0
```

```
DBarInt = 0.020
nBarSh = 3
DBarSh = 0.012
SStirrup = 0.100
nFac = 10
cUnitToMPa = 1e-6


crackFac = ComputeHingeRC(matTag, B, H, L, cover, P, fpc, Ec, fyBot, fyTop, MyP, MyN, nBarBot,
DBarBot, nBarTop, DBarTop, nBarInt, DBarInt, nBarSh, DBarSh, SStirrup, nFac, cUnitToMPa)
```

Appendix D: 3-D Models of Structures

In figures 107, 108, and 109, 3-D models of 7-story, 10-story and 15-story respectively, are provided.

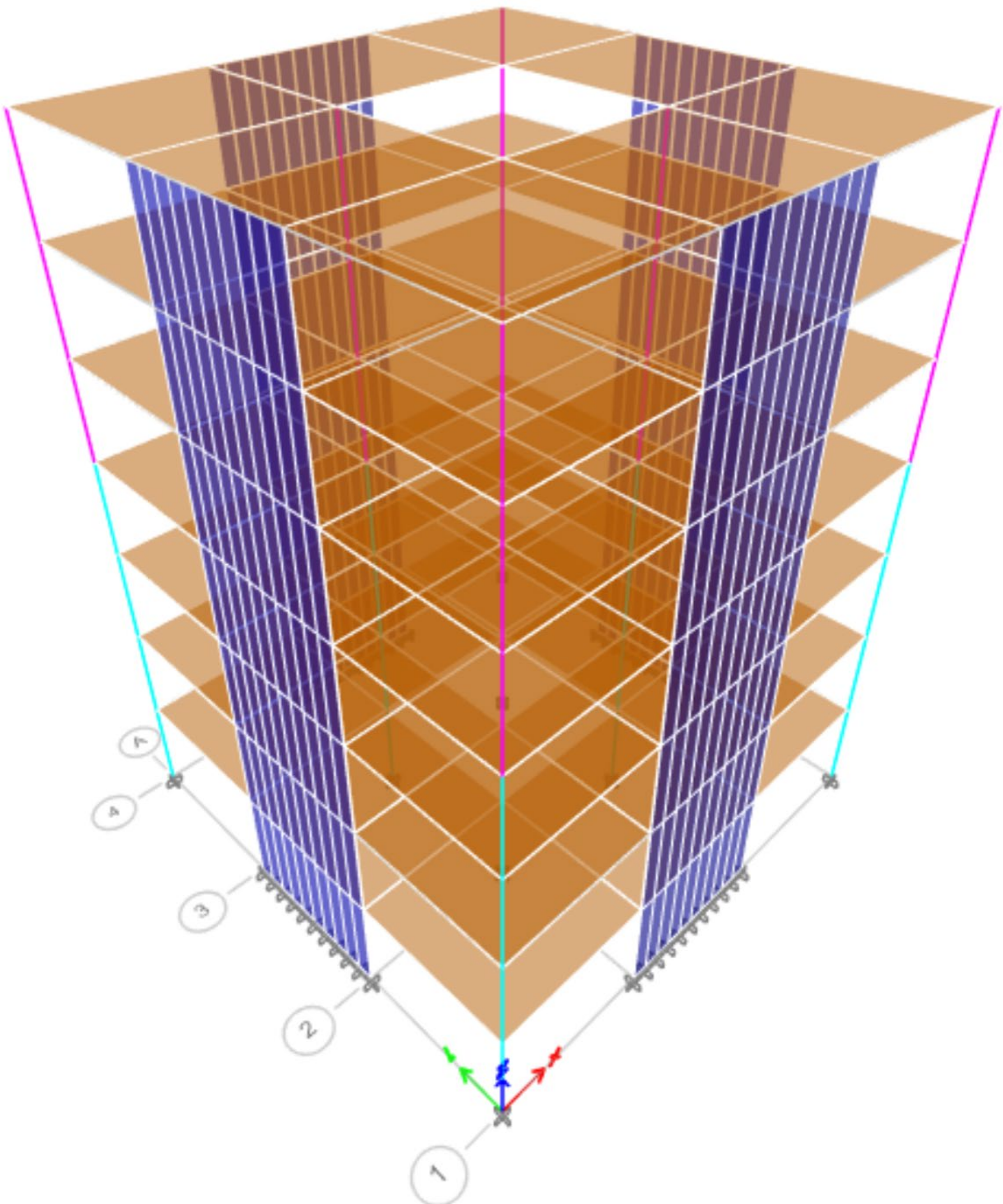


Figure 107: 3-D model of 7-story

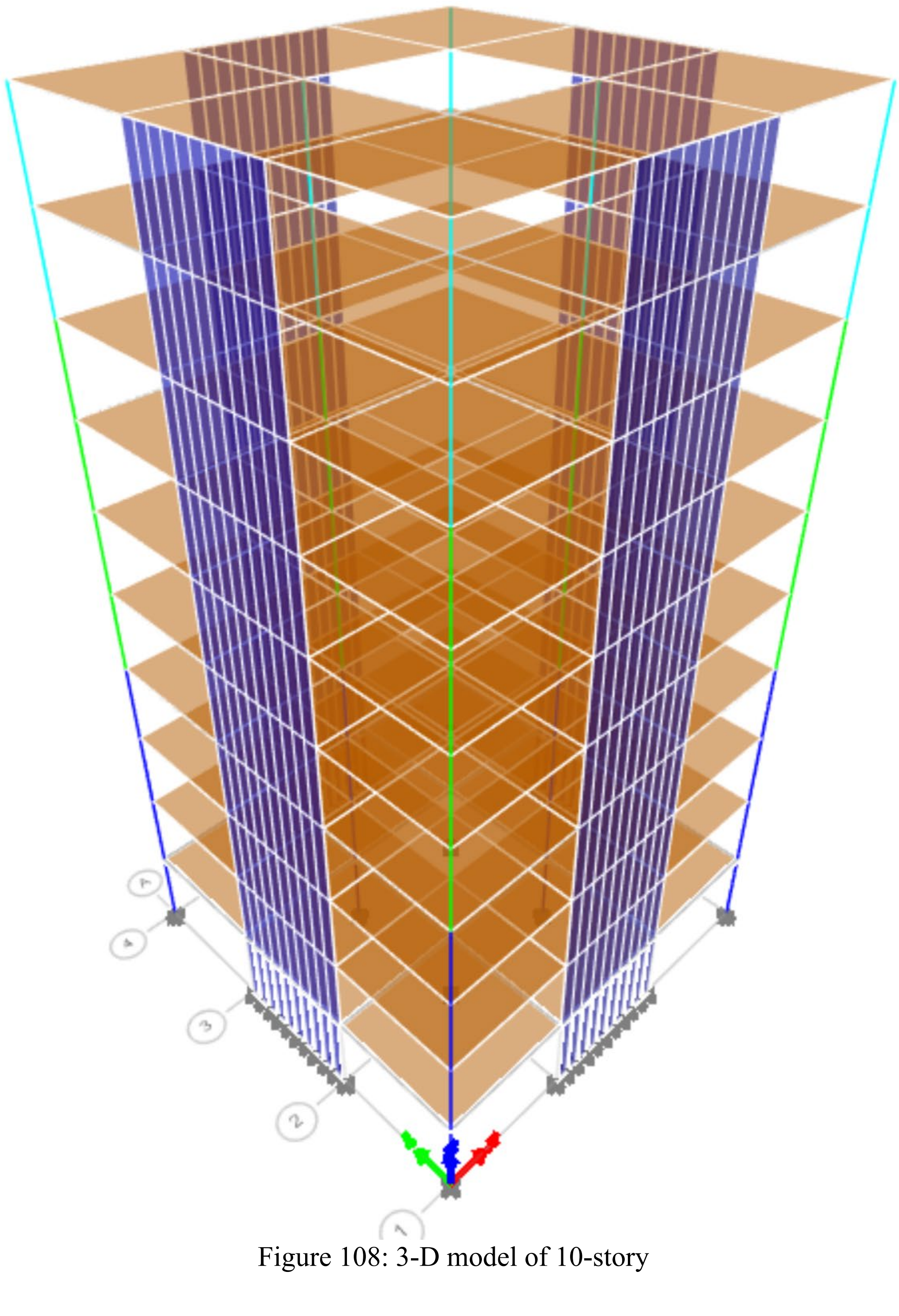


Figure 108: 3-D model of 10-story

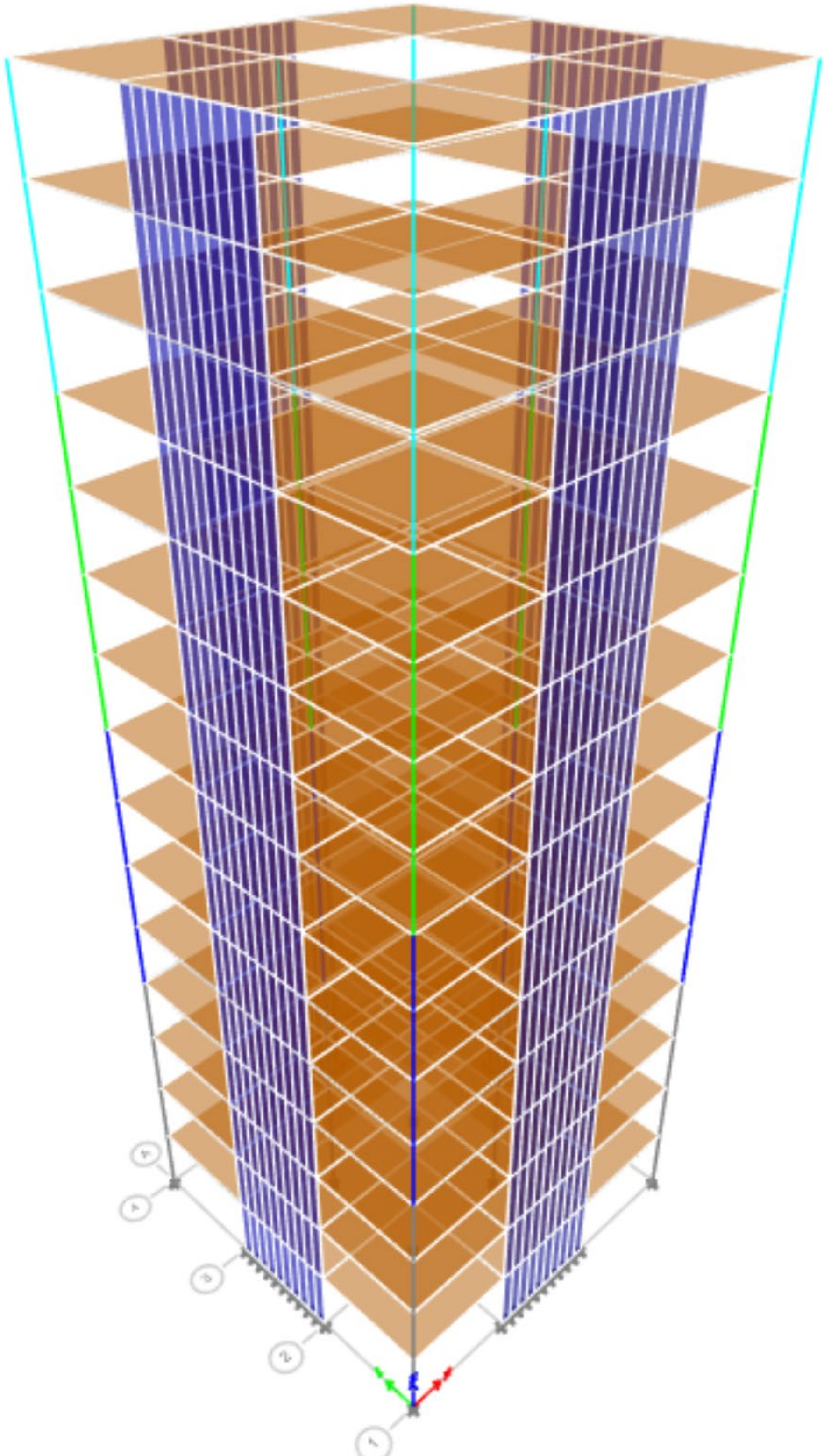


Figure 109: 3-D model of 15-story