

Addis Ababa University
Addis Ababa Institute of Technology
School of Civil and Environmental Engineering



**Software Development for Capacity Design of Earthquake
Resistant Reinforced Concrete Plane Frames (RCPF)**

by

Yonathan Bekele Adamu

Sponsor: Ethiopian Roads Authority [ERA]

January 29, 2021

A Thesis submitted at Addis Ababa Institute of Technology
in partial fulfillment of the requirements for
the Degree of Master of Science in Structural Engineering

The undersigned have examined the thesis entitled ‘**Software Development for Capacity Design of Earthquake Resistant Reinforced Concrete Plane Frames**’ presented by **Yonathan Bekele**, a candidate for the degree of Master of Science and hereby certify that it is worthy of acceptance.

Dr.-Ing. Bedilu Habte

Advisor

Signature

Date

Dr.-Ing. Adil Zekaria

Internal Examiner

Signature

Date

Dr.-Ing. Girma Zerayohannes

External Examiner

Signature

Date

Dr.-Ing. Mebruk Mohammed

Chair Person

Signature

Date

UNDERTAKING

I certify that the research work titled “**Software Development for Capacity Design of Earthquake Resistant Reinforced Concrete Plane Frames**” is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources, it has been properly acknowledged /referred.

Yonathan Bekele

Abstract

Experiences from the response of structures during earthquakes and the nature of past earthquakes themselves have taught designers that ductility is one of the important elements in seismic resistant design. When a structure is said to be ductile it denotes that the structure possesses the ability to dissipate energy through plastic deformations. Designing buildings to resist earthquake induced forces elastically results in an uneconomical construction. On top of that, the earthquake which the structure is designed to withstand may not occur during its lifetime. But engineering is about economy as much as safety. Generally speaking, structural design is finding a balance between these two key elements and designing economical structures without compromising safety. Nonetheless, in the process of significantly dissipating energy it may be necessary to accept some damages in predetermined locations. These locations are expected to possess the required ductility and rotation capacities to exhibit the desired behaviour. Here is where the strong-column-weak-beam concept comes into play which, as the name implies, bases its philosophy on providing columns with flexural capacities that are greater than that of beams. This is to avoid soft storey failure that can affect the stability of a structure; but the failure of beams usually has a localized effect. Shear capacities of members are also to be greater than their flexural capacities so that shear failure, a brittle failure, may not precede the ductile and more preferable flexural failure. The capacity design method is based on these concepts.

In this research, software for the implementation of the capacity design process of reinforced concrete plane frames is presented. Design rules for capacity design were based on those described in the 2015 Ethiopian Building Code (Specifically ES EN 1990, ES EN 1992 and ES EN 1998). The software generally performs analysis, regular design and capacity design. It automates the reanalysis and redesign of failed members after increasing their sections using the preset increment value. It also automatically converts continuous reinforcement areas to discrete number of bars (or bar center-to-center spacings in case of shear reinforcing) for the program determined bar diameter.

Keywords: Capacity Based Design, Earthquake Resistant Design, Reinforced Concrete Frame Design, Overstrength Factor, Ductile Frames, Capacity Design Software.

ACKNOWLEDGMENTS

Even though it may seem stating the obvious, I would like to begin by expressing my heartfelt gratitude to the Almighty God for all His provisions throughout this research and even throughout my stay in this institute. And also for His love and mercy which is the reason of our existence.

Secondly, I would like to thank my advisor, Dr.-Ing. Bedilu Habte, for meticulously guiding me in my attempt to reach my objectives and present a worthy paper. I would also like to acknowledge Mr. Elias Yilma (a lecturer in AAiT) for making his work available as input to my research and himself for clarifications on the same. Special thanks to Ethiopian Roads Authority's leadership for making sponsorship funds available for my master's program in general and this research in particular.

Last but definitely not least, my parents, who have always been there for me and believed in me even when I myself did not.

Contents

1	Introduction	1
1.1	General Background	1
1.2	Motivation	2
1.3	Statement of the Problem	3
1.4	Scope and Limitations	3
1.5	Objectives	3
1.5.1	General Objectives	3
1.5.2	Specific Objectives	4
1.6	Thesis Organization	4
1.6.1	Introduction	4
1.6.2	Literature Review	4
1.6.3	Research Methods, Materials and Procedure	4
1.6.4	Implementation	5
1.6.5	Results and Discussion	5
1.6.6	Conclusions	5
2	Literature Review	6
2.1	Introduction	6
2.2	Earthquake Resistant Design	8
2.3	Capacity Based Design	14
2.3.1	Definition	14
2.3.2	Capacity Design for RCPFs	18
2.4	Computer Programming	22
2.4.1	History[25]	22
2.4.2	Programming in Java	23
2.4.3	Why Java?	24
3	Research Methods, Materials and Procedures	26
3.1	General	26
3.2	Methods and Materials	26
3.2.1	Literature	26

3.2.2	Software and Utilities	26
3.3	Software Design and Result Verification	27
3.3.1	Design Assumptions	27
3.3.2	Design Procedure	27
3.3.3	HELIX[27]	27
3.3.3.1	Main Interface and Grid Definition	28
3.3.3.2	Ribbon Functions	28
3.3.3.3	Sub-menus	29
3.3.4	Earthquake Load Computation and Analysis	32
3.3.5	Beam Design[29]	35
3.3.5.1	Design for Flexure: Singly reinforced rectangular sections with moment redistribution ($\mu_{sd,s} \leq \mu_{sd,s}^*$)	36
3.3.5.2	Design for Flexure: Doubly reinforced rectangular sec- tions with moment redistribution ($\mu_{sd,s} > \mu_{sd,s}^*$)	38
3.3.6	Column Design[29]	39
3.3.6.1	Column Classification (Short or Slender)	40
3.3.6.2	Design of Short Columns ($\lambda < \lambda_{lim}$)	43
3.3.6.3	Design of Slender Columns ($\lambda \geq \lambda_{lim}$)	44
3.3.6.4	Interaction Curve Development	46
3.3.7	Design for Shear[29]	48
3.3.8	Capacity Design[5]	49
3.3.8.1	Primary Beams	51
3.3.8.2	Primary Columns	54
3.3.9	Output	57
3.3.10	Testing and Verification	57
4	Implementation	58
4.1	General Program Structure	58
4.2	Earthquake Load and Analysis	59
4.2.1	Earthquake Load Computation	59
4.2.2	Earthquake Analysis	59
4.3	Member Design Implementation	60
4.3.1	Assumptions	60
4.3.2	Beam Design	60
4.3.3	Column Design	60
4.4	Capacity Design Implementation	60
4.5	Modification to Helix's Interface	61
4.5.1	Modified Ribbon and New Functions	62
4.5.2	Modified and New Sub-menus	63

4.5.3	The Results Output Dialogs and Tables	64
4.6	Main Program Flowchart	65
4.7	Algorithm Listings	67
4.7.1	Main Program	68
4.7.2	Earthquake Load Computation	69
4.7.3	Member Design	74
4.7.4	Capacity Design	78
5	Results and Discussion	79
5.1	General	79
5.2	Result Comparison	80
5.2.1	Frame A (3 Storey x 4 Bay)	80
5.2.1.1	Section Comparison	81
5.2.1.2	Reinforcement Comparison	81
5.2.2	Frame B (5 Storey x 5 Bay)	83
5.2.2.1	Section Comparison	83
5.2.2.2	Reinforcement Comparison	84
5.2.3	Frame C (10 Storey x 4 Bay)	86
5.2.3.1	Section Comparison	86
5.2.3.2	Reinforcement Comparison	87
6	Conclusions	89
6.1	General	89
6.2	Application of Results	89
6.3	Future Use	90
A	Program Listings	94
A.1	Earthquake Load Computation and Distribution	94
A.2	Member Design	103
A.3	Capacity Design Check	121
A.4	Main Program	125

List of Algorithms

4.1	Main Program	68
4.2	Calculate Fundamental Period of Structure, T_1	69
4.3	Calculate Design Ground Acceleration, α_g	70
4.5	Compute Base Shear Force, F_b	71
4.4	Calculate S_d	72
4.6	Storey Distribution of Horizontal Seismic Force	73
4.7	Beam Design for Flexure	74
4.8	Determine Slenderness and Slenderness Limit of Column	75
4.9	Column Design	76
4.10	Design for Shear	77
4.11	Capacity Design	78

List of Tables

2.1	Different Earthquake Resistant Design Underlying Concepts[16]	12
2.2	Mandatory and Optional Earthquake Resistant Design Methods [16] . . .	13
3.1	Main Ribbon Functions	29
3.2	Values of Parameters Describing the Recommended Type 1 Elastic Response Spectra[5]	32
3.3	Values of Parameters Describing the Recommended Type 2 Elastic Response Spectra[5]	33
3.4	Importance Classes and Importance Factor[5]	33
3.5	Bedrock Acceleration Ratio[5]	34
3.6	Basic Behaviour Factor Computation, q_o [5]	34
4.1	Modified Main Ribbon Functions	62
5.1	Input Data for Each Frame	79
5.2	Material, Section and Loading Input Data (Same for all frames)	80

List of Figures

2.1	Chain with Brittle and Ductile Links[9]	16
2.2	Plastic Mechanisms in Multistorey Buildings [21]	18
2.3	Comparison of Energy Dissipating Mechanisms[9]	19
2.4	Capacity Design Flow Chart	21
3.1	Main Canvas Components	28
3.2	Sub-Menus 1	31
3.3	Sub-Menus 2	31
3.4	Beam Design Flow Chart	36
3.5	Column Design Flow Chart	40
3.6	Effective Column Height Calculation[29]	41
3.7	Section Analysis	46
3.8	Capacity Design Values of Moments for Shear Force Calculation in Beams (a)[5]	53
3.9	Capacity Design Values of Moments for Shear Force Calculation in Beams (b)[10]	54
3.10	Capacity Design Values of Moments for Shear Force Calculation in Columns (a)[5]	56
3.11	Capacity Design Values of Moments for Shear Force Calculation in Columns (b)[10]	56
4.1	Capacity Design Program for Frames	58
4.2	Modified Sub-Menus	63
4.3	Earthquake Data Display Dialog	64
4.4	Story Forces Output Table	64
4.5	Member Result Information Dialog	65
4.6	Design Output Table	65
4.7	Main Program Flowchart	66
5.1	Frame A	80
5.2	Column Section Comparison - Frame A	81
5.3	Beam Depth Comparison - Frame A	81

5.4	Column Reinforcement Comparison- Frame A	81
5.5	Beam Reinforcement Comparison - Frame A	82
5.6	Frame B	83
5.7	Column Section Comparison - Frame B	83
5.8	Beam Depth Comparison - Frame B	84
5.9	Column Reinforcement Comparison- Frame B	84
5.10	Beam Reinforcement Comparison - Frame B	85
5.11	Frame C	86
5.12	Column Section Comparison - Frame C	86
5.13	Beam Depth Comparison - Frame C	87
5.14	Column Reinforcement Comparison- Frame C	87
5.15	Beam Reinforcement Comparison - Frame C	88

Chapter 1

Introduction

1.1 General Background

Civil engineering structures are designed to resist static and dynamic loads. Dynamic loads are more difficult to estimate than the static ones. This difficulty comes from their unpredictable and sophisticated nature which may lead to misconceptions and ultimately become the cause of disaster, particularly in case of earthquake. Conventional civil engineering structures are designed on the basis of strength and stiffness criteria. Strength is related to ultimate limit state, which ensures that the forces developed in the structure remain in elastic range[1]. Stiffness is related to serviceability limit state which assures that the structural displacements remain within the permissible limits[1]. In case of earthquake forces the demand is for ductility. Ductility is an essential attribute of a structure that must respond to strong ground motions. Ductility is the ability of the structure to undergo distortion or deformation without damage or failure which results in dissipation of energy. The larger the capacity of the structure to deform plastically without collapse, the more the resulting ductility and the energy dissipation. This brings reduction in effective earthquake forces[1].

The reinforcing details required to ensure adequate ductility of hinging regions in a laterally loaded structure tend to be tedious to design and expensive to place. This complexity can be reduced if the structure is designed so that only a few cross sections form hinges under the seismic loads, while the rest of the structure has enough reinforcement to remain elastic.[2]

Designers of structures that may be subjected to earthquakes, therefore, are faced with a choice. It is either providing adequate stiffness and strength to limit the response of structures to the elastic range or providing lower-strength, with presumably lower initial costs, that have the ability to withstand large inelastic deformations while maintaining their load-carrying capability[3]. The later method of designing reinforced concrete buildings is called the capacity design method.

In this method, the right building components need to be made ductile so that seismic inertia forces generated at floor levels could be transferred through the various beams and columns to the ground. If plastic hinges form in columns, the stability of the structure may be compromised. As a result, the design of ductile moment-resisting frames attempts to force the structure to respond in which the plastic hinges induced by the seismic forces form at the ends of the beams[2]. This is because failure of a column can affect the stability of the whole building, but the failure of a beam has localized effect. Therefore, it is better to make beams to be the ductile weak links than columns[4]. Hence, the capacity design method is also known as the strong-column-weak-beam design method. The hinging regions are detailed to maintain their shear capacity while the plastic hinges undergo flexural yielding in both positive and negative bending[2]. The procedures involved to achieve this are time consuming as much as they are repetitive compared to the conventional design method. Nevertheless, the Ethiopian Building Code states that; “the design of earthquake resistant concrete buildings shall provide the structure with adequate capacity to dissipate energy without substantial reduction of its overall resistance against horizontal and vertical loading.”[5]. Since one of the ways to achieve this is through the capacity design method, the need for simpler ways to engage in it arises. Here the development of software comes into play.

The goal of this thesis is to bring a contribution to the design of earthquake resistant reinforced concrete frames; specifically using the capacity design philosophy.

1.2 Motivation

It is evident that the design of economic and attractive structures which can successfully withstand forces induced by a severe ground motion is a challenge demanding the best in structural engineering. In addition, as an upcoming engineer specializing in structural engineering, earthquake engineering has always been the one particular field that the researcher is most attracted to. A number of components such as safety, design approach, economy, building codes and the like play significant roles in earthquake resistant structural design. Assessing this complex issue means dealing with these different components. Combined with the fact that not too many young structural engineers these days acknowledge or pay attention to this subject, the researcher has decided to contribute whatever little it might be in his own capacity. Furthermore, the statement made by Newmark and Rosenblueth says it all: “Earthquakes systematically bring out the mistakes made in design and construction-even the most minute mistakes; it is this aspect of earthquake engineering that makes it challenging and fascinating, and gives it an educational value far beyond its immediate objectives.”[6].

1.3 Statement of the Problem

The new Ethiopian Building Code gives an elaborate and rigorous approach to earthquake resistant design. Capacity design, the underlying concept of this code for designing ductile structures (those belonging to either ductility class medium or high), imposes stringent requirements for the fulfillment of its cause. Though this will result in structures that behave favorably to minimize damages from earthquakes, it might be an arduous task to undertake. The reasons to why, among many, could be the lengthy and iterative procedures involved in earthquake force determination, analysis, design, reanalysis, redesign... of the structure under consideration. Thus, the development of capacity based seismic design software would undoubtedly help in encouraging the application of the code's procedures and ultimately improve the commonly used design approach.

1.4 Scope and Limitations

This thesis covers the development of software for the capacity design of earthquake resistant reinforced concrete plane frames. It does not address:

- Any other structural element section apart from rectangular ones.
- Three dimensional analysis or design. Input data processed shall be uni-axial moments, shear forces and axial forces only.
- Inclined and curved members. All members are assumed to have either horizontal or vertical alignments.
- Irregular frames. All frames are assumed to be regular as per the Ethiopian building code.
- Detailing of members. The software is to design members not detail them.
- Joint design nor joint detailing.

1.5 Objectives

1.5.1 General Objectives

The general objective of this research is to develop software for capacity design of two-dimensional earthquake resistant reinforced concrete frames which shall serve as an auxiliary design tool.

1.5.2 Specific Objectives

- To study the basics of capacity design philosophy that could be represented by algorithms.
- To develop, according to the Ethiopian Building Code (ES EN), a capacity design procedure of two dimensional reinforced concrete frames that will serve as the basis for the formulation of an algorithm.
- To find a simplified way of representing capacity design procedures using varying algorithm parameters.
- To solve the problem for finding the right programming approach and tool for the work.
- To develop an integrated software architecture for testing and using the design procedure developed.

1.6 Thesis Organization

1.6.1 Introduction

The need for developing software to facilitate structural design works, especially when it comes to earthquake analysis and design, is presented in this first chapter. This introductory chapter also includes the motivation behind this research and gives the problem statement followed by its scope and limitations. Finally, it addresses general and specific objectives.

1.6.2 Literature Review

Thorough researching of the subject matter involved is important so as any paper can base its results on profound and reliable facts. This second chapter is dedicated to performing that. It provides a brief summary of the major thoughts underlying earthquake analysis, earthquake design, capacity design and java programming.

1.6.3 Research Methods, Materials and Procedure

This section, the third chapter, focuses on how the paper intends to tackle the problem stated in the first chapter. It elaborates each method and tool required to accomplish that with specific procedures laid out clearly.

1.6.4 Implementation

Here the methods that have been specified in third chapter shall be engaged towards achieving the objectives stated in the first chapter. It explains how the algorithms and the program were developed and shows how they can be implemented.

1.6.5 Results and Discussion

Results obtained from execution of the program developed in the fourth chapter will be presented in tabular and graphical tools. They will, at the same time, be compared with results from other such software to validate them.

1.6.6 Conclusions

The observations from the fifth chapter are expected to lead to certain conclusions and the conclusions in turn to recommendations. Therefore, this final chapter shall attempt to summarize significant findings of the research along with future areas of further study.

Chapter 2

Literature Review

2.1 Introduction

Almost any type of structural system may be subjected to one form or another of dynamic loading during its lifetime. The dynamic loadings that act on structural systems may result from a wide range of input mechanisms. One important class of loading involves vehicular systems such as ships, airplanes, automobiles, etc., for which the dynamic loading is a result of the motion of the vehicle. The other basic class concerns fixed civil engineering structures such as bridges, buildings, dams, etc., to which the dynamic loading is applied externally. Of the many sources of external load that must be considered in the design of fixed structures, the most important by far in terms of its potential for disastrous consequences is the earthquake.[6]

Of the 500,000 or so detectable earthquakes that occur on Planet Earth each year, people will “feel” about 100,000 of them and about 100 will cause damage. Although most earthquakes are moderate in size and destructive potential, a severe earthquake occasionally strikes a community that is not adequately prepared and thousands of lives and billions of dollars in economic investment are lost.[7]

Complete or partial structural collapse is the major cause of fatalities from earthquakes worldwide; earthquakes themselves seldom kill people, collapsing buildings do. Earthquake energy causes structures, not sufficiently designed to resist earthquakes, to move laterally. Interest in constructing buildings to provide greater resistance to earthquakes arose in association with the scientific and professional development of engineering, especially from the late 1800s and early 1900s, in response to large earthquake damages that occurred in Japan, Italy, and California. For instance, the earthquake near San Francisco, in April 1906 (magnitude $M = 7.8$ on the Richter scale, 3,000 fatalities) destroyed structures in an area 350 miles long by 70 miles wide, and was the most expensive natural disaster in U.S. history until hurricane Andrew in 1992, with \$500 million in damages (equivalent to \$10 billion in 2004 dollars).[8]

Earthquakes may result from a number of natural and human-induced phenomena, including meteoric impact, volcanic activity, underground nuclear explosion, and rock stress changes induced by the filling of large human-made reservoirs. However, the vast majority of damaging earthquakes originate at, or adjacent to, the boundaries of crustal tectonic plates, due to relative deformations at the boundaries.[9]

The basic concept embodied in plate tectonics was recognized several centuries ago by persons working with world maps who noted that the outline of the west coast of Africa is a good match for the east coast of South America and suggested that the two continents might at one time have been joined together. Although this fact was a matter of great curiosity, it was not until much later that a Frenchman, Antonio Snider Pellegrini made the first attempt to develop the concept of substantial continental movement; in a book published in 1858 he hypothesized that the Atlantic Ocean had been formed by the tearing apart and separation of the two continents. The author supported this postulation of “continental drift” (as it came to be known) by presenting extensive evidence that the geological structure on the South American coast was very similar to that on the African coast and also by identifying similarities between the ancient plant and animal life in the two regions as indicated by their fossil specimens.[6]

Plate tectonics theory visualizes the earth as consisting of a viscous, molten magma core with a number of lower-density rock plates floating on it. The exposed surfaces of the plates form the continents and the bottoms of the oceans. As time goes by, the plates move relative to each other, breaking apart in some areas and jamming together in others. Where the plates are moving apart, this movement causes cracks (or rifts) to form, generally in the ocean beds. In some cases, molten magma flows out of these rifts. The regions where the plates are either moving into each other or are sliding adjacent to each other are referred to as fault zones. Compression and shear stresses are generated in the plates and strain energy builds up at the edges of the plates. At some point in time, the stresses and strain energy at a locked fault surpass the limiting resistance to rupture or slip along the fault. Once started, energy is released rapidly, causing intense vibrations to propagate out from the fault. Three main types of stress waves travel through the rock layers: primary (compression) waves, secondary (shear) waves, and surface waves—each at different speeds. As a result, the effects of these seismic waves and local soil conditions will lead to different ground motions at various sites. Earthquakes may involve regions of slip and/or offsets along surface faults.[2]

Despite the clear preponderance of earthquakes associated with plate boundaries, earthquakes can occur, within plates, at considerable distance from boundaries with devastating effects. The largest earthquake in the contiguous 48 states of the United States in recorded history did not occur in California but in New Madrid, Illinois, in 1811. Frequently, the lack of recent earthquake activity in an area results in a false sense of security and a tendency to ignore seismic effects in building design. Although

the annual risk of significant earthquake activity may be low in intraplate regions, the consequences can be disastrous, and should be assessed, particularly for important or hazardous structures.[9]

2.2 Earthquake Resistant Design

Earthquake resistant design can be considered as the art of balancing the seismic capacity of structures with the expected seismic demand to which they may be subjected. In this sense, earthquake resistant design is the mitigation of seismic risk which may be defined as the possibility of losses (human, social or economic) due to the effects of future earthquakes. Seismic risk is often considered as the convolution of seismic hazard, exposure and vulnerability. Exposure refers to the people, buildings, infrastructure, commercial and industrial facilities located in an area where earthquake effects may be felt; exposure is usually determined by planners and investors, although in some cases avoidance for major geo-hazards may lead to relocation of new infrastructure. Vulnerability is the susceptibility of structures to earthquake effects and is generally defined by the expected degree of damage that would result under different levels of seismic demand; this is the component of the risk equation that can be controlled by engineering design. Seismic hazards are the potentially damaging effects of earthquakes at a particular location, which may include surface rupture, tsunami run-up, liquefaction and landslides, although the most important cause of damage on a global scale is earthquake-induced ground shaking. In the context of probabilistic seismic hazard analysis (PSHA), seismic hazard actually refers to the probability of exceeding a specific level of ground shaking within a given time.[10]

As experienced by structures, earthquakes consist of random horizontal and vertical movements of the earth's surface. As the ground moves, inertia tends to keep structures in place resulting in the imposition of displacements and forces that can have catastrophic results. The purpose of seismic design is to proportion structures so that they withstand the displacements and the forces induced by the ground motion.[3]

These earthquake ground motions impart vertical and horizontal accelerations to the base of a structure. If the structure was completely rigid, forces of magnitude $F = ma$ would be generated in it, where a is the horizontal acceleration and m is the mass of the structure. Because real structures are not rigid, the actual forces generated will differ from this value depending on the period of the building and the dominant periods of the earthquake ground motions. The determination of the seismic force is made more complicated because recorded earthquake ground motions contain a wide range of frequencies and maximum values of base acceleration[2]. The assessment of ground-shaking hazard due to future earthquakes invariably involves three steps: the development of a seismicity model for the location and size (and, if appropriate, the frequency) of future earthquakes

in the region; the development of a ground-motion model for the prediction of expected levels of shaking at a given site as a result of any of these earthquake scenarios; and the integration of these two models into a model for the expected levels of shaking at the site of interest[10].

Earthquake forces are generated by the inertia of buildings as they dynamically respond to ground motion. The dynamic nature of the response makes earthquake loadings markedly different from other building loads. The dynamic characteristics of buildings are fundamental to their structural response and thus the earthquake induced actions are able to be mitigated by design.[11]

If resources were unlimited, seismic protection would be achieved by simply providing as much earthquake resistance as possible to structures. In practice, it is not feasible to reduce seismic vulnerability to an absolute minimum because the costs would be prohibitive and certainly not justified since they would be for protection against a loading case that may not even occur during the useful life of the structure. Seismic design therefore seeks to balance the investment in provision of seismic resistance against the level of damage, loss or disruption that earthquake loading could impose.[10]

In general terms, the primary objective of earthquake resistant design is to prevent building collapse during earthquakes thus minimizing the risk of death or injury to people in or around those buildings. Because damaging earthquakes are rare, economics dictate that damage to buildings is expected and acceptable provided collapse is avoided[11]. The challenge to the engineer is to design the structure so that the damage is controlled to an acceptable degree[12].

During a seismic excitation, structures are caused to oscillate in response to the forced motion of the foundations. The affected structure needs to be able to resist the resulting horizontal load, and also dissipate the imparted kinetic energy over successive deformation cycles. It would be uneconomical to design the structure to withstand a major earthquake elastically, and the normal approach is to provide it with sufficient strength and ductility to withstand such an event by responding inelastically, provided that the critical regions and the connections between members are designed specially to ensure adequate ductility[13]. The design of structures to perform satisfactorily under expected seismic conditions requires that realistic earthquake loadings be specified and that the structural components be proportioned to resist these and other combined loadings within the limits of certain design requirements[6].

The concept of dynamic considerations of buildings is one which sometimes generates unease and uncertainty within the designer. Although this is understandable, and a common characteristic of any new challenge, it is usually misplaced. Effective earthquake design methodologies can be, and usually are, easily simplified without detracting from the effectiveness of the design. Indeed the high level of uncertainty relating to the ground motion generated by earthquakes seldom justifies the often used complex analysis

techniques nor the high level of design sophistication often employed. A good earthquake engineering design is one where the designer takes control of the building by dictating how the building is to respond. This can be achieved by selection of the preferred response mode, selecting zones where inelastic deformations are acceptable and suppressing the development of undesirable response modes which could lead to building collapse.[11]

The incorporation of seismic design procedures in building design was first adopted in a general sense in the 1920s and 1930s, when the importance of inertial loadings of buildings began to be appreciated. In the absence of reliable measurements of ground accelerations and as a consequence of lack of detailed knowledge of the dynamic response of structures, the magnitude of seismic inertial forces could not be estimated with any reliability. Typically, design for lateral forces corresponding to about 10% of the building weight was adopted. Since elastic design to permissible stress levels was invariably used, actual building strengths for lateral forces were generally somewhat larger.[9]

By the 1960s accelerograms giving detailed information on the ground acceleration occurring in earthquakes were becoming more generally available. The advent of strength design philosophies and development of sophisticated computer-based analytical procedures facilitated a much closer examination of the seismic response of multi-degree-of-freedom structures. It quickly became apparent that in many cases, seismic design to existing lateral force levels specified in codes was inadequate to ensure that the structural strength provided was not exceeded by the demands of strong ground shaking. At the same time, observations of building responses in actual earthquakes indicated that this lack of strength did not always result in failure, or even necessarily in severe damage. Provided that the structural strength could be maintained without excessive degradation as inelastic deformations developed, the structures could survive the earthquake, and frequently could be repaired economically. However, when inelastic deformation resulted in severe reduction in strength, as, for example, often occurs in conjunction with shear failure of concrete or masonry elements, severe damage or collapse was common.[9]

Most recently, it has become accepted that seismic design should encourage structural forms that are more likely to possess ductility than those that do not. Generally, this relates to aspects of structural regularity and careful choice of the location, often termed plastic hinges, where inelastic deformations may occur. In conjunction with the careful selection of structural configuration, required strengths for undesirable inelastic deformation modes are deliberately amplified in comparison with those for desired inelastic modes. Thus for concrete and masonry structures, the required shear strength must exceed the required flexural strength to ensure that inelastic shear deformations, associated with large deterioration of stiffness and strength, which could lead to failure, cannot occur. These simple concepts are the bases for the capacity design philosophy.[9]

Procedures for the analysis and design of structures to resist the effects of earthquake ground motions are in a continuous state of development. Over the years various analysis

techniques have been developed to put into account the effect of earthquake accelerations on buildings. Main features of seismic method of analyses are described as follows;

- **Equivalent Lateral Force Method**

If a structure can reasonably be assumed to be dominated by a single (normally the fundamental) mode then a simple static analysis procedure, equivalent lateral force method, can be used that involves only minimal consideration of the dynamic behaviour[10]. The equivalent lateral force method is the simplest method of analysis and requires less computational effort because the forces depend on the code based fundamental period of structures with some empirical modifier. The design base shear shall first be computed as a whole, and then be distributed along the height of buildings based on simple formulae appropriate for buildings with regular distribution of mass and stiffness. The design lateral force obtained at each floor level shall be distributed to individual lateral load resisting elements depending upon floor diaphragm action[1]. The design lateral force or design base shear and the distribution are given by some empirical formulae presented in the Ethiopian Building Code. Typically, the equivalent lateral force method is permitted for regular buildings up to about 20 stories. Sometimes it can be used for irregular buildings if special attention is given to the types of irregularities[2].

- **Response Spectrum Analysis**

A response spectrum is a plot of the peak or steady-state response (displacement, velocity or acceleration) of a series of oscillators of varying natural frequency, that are forced into motion by the same base vibration or shock[14]. This method is applicable for those structures where modes other than the fundamental one affect significantly the response of the structure. In this method the response of multi-degree of freedom system is expressed as the superposition of modal responses, each modal response being determined from the spectral analysis of single-degree of freedom (SDOF) system, which is then combined to compute the total response[1]. So the response spectrum shows the peak response of an SDOF structure to a particular earthquake, as a function of the natural period and damping ratio of the structure[10]. Significant seismic damage may occur if the building response is “in tune” with components of the ground motion (resonance), which may be identified from the response spectrum[14]. A key advantage of this approach is that earthquakes that look quite different when represented in the time domain may actually contain similar frequency contents and so result in broadly similar response spectra. This makes the response spectrum a useful design tool for dealing with a future earthquake whose precise nature is unknown. To create a design spectrum, it is normal to compute spectra for several different earthquakes, then

envelope and smooth them, resulting in a single curve that encapsulates the dynamic characteristics of a large number of possible earthquake accelerograms[10].

- **Elastic Time History Analysis**

A linear analysis, the time history analysis, overcomes all disadvantages of modal response spectrum provided non-linear behavior is not involved. Time history analysis is a step-by-step analysis of the dynamic response of a structure to a specified loading that may vary with time. Time history analysis is used to determine the seismic response of a structure under dynamic loading of representative earthquake. The method requires greater computational efforts for calculating the response at discrete times. One interesting advantage of this is that the relative signs of response quantities are preserved in the response histories[1]. A full time history will give the response of a structure over time during and after the application of a load[15].

As in any other engineering work, design follows analysis and the results obtained from any of the above methods will serve as design inputs. Earthquake resistant design approaches are classified based on three criteria; strength, stiffness and ductility.

Table 2.1: Different Earthquake Resistant Design Underlying Concepts[16]

Design Method	Underlying Concept
Lateral Strength Based	Minimum lateral strength to resist seismic loads
Displacement Based	Takes into account the anticipated earthquake induced displacements
Capacity Based	Strong-Column-Weak-Beam Philosophy
Energy Based	Assumes the total energy input is resisted by kinetic energy, the elastic strain energy and energy dissipated through plastic deformations and damping

- **Lateral Strength Based Design**

This is the most common seismic design approach adopted nowadays. It is based on providing the structure with the minimum lateral strength to resist seismic loads, assuming that the structure will behave adequately in the non-linear range. For this reason only some simple construction detail rules are needed to be satisfied[1].

- **Displacement Based Design**

In this method the structure is designed to possess adequate ductility so that it can dissipate energy by yielding and survive the shock[1]. It takes into account the anticipated earthquake induced displacements for the design event(s). Furthermore, an emphasis is directed to achieving a realistic and realistically conservative

estimate of the displacements[17]. The displacement base design uses the fact that displacement correlates better with damage than force levels. This method operates directly with deformation quantities hence gives better insight on the expected performance of the structures. The displacement based design approach has been adopted by the seismic codes of many countries[1].

- **Capacity Based Design**

In this design approach the structures are designed in such a way so that plastic hinges can form only in predetermined positions and in predetermined sequences[18]. The concept of this method is to avoid brittle mode of failure. This is achieved by designing the brittle modes of failure to have higher strength than ductile modes[1]. The subsequent subsection gives an in depth discussion of this method.

- **Energy Based Design**

This is the most promising and futuristic approach of earthquake resistant design. One of the reasons being the fact that energy is a relatively stable parameter when compared with other response parameters[19]. In this approach it is assumed that the total energy input is collectively resisted by kinetic energy, the elastic strain energy and energy dissipated through plastic deformations and damping[1]. The earthquake input energy transmitted to the structure is estimated from ground motion and then distributed to various structural elements and damping devices to be dissipated[20].

Table 2.2: Mandatory and Optional Earthquake Resistant Design Methods [16]

Design Approach	Mandatory For	Optional For
Lateral Strength Based	Normal buildings in low seismic zones	-
Capacity Based	Normal buildings in moderate/high seismic zones	Normal buildings in low seismic zones
Displacement and Energy Based	Critical and lifeline buildings	Normal buildings in moderate/high seismic zones

As cited in Chapter 1, the Ethiopian Building Code states that; “the design of earthquake resistant concrete buildings shall provide the structure with adequate capacity to dissipate energy without substantial reduction of its overall resistance against horizontal and vertical loading.”[5]. This research focuses on the capacity design method because it is one of the means to fulfill the aforementioned code requirement.

2.3 Capacity Based Design

2.3.1 Definition

Capacity design is a concept or a method of designing flexural capacities of critical member sections of a building structure based on a hypothetical behavior of the structure in responding to seismic actions. This hypothetical behavior is reflected by the assumptions that the seismic action is of a static equivalent nature increasing gradually until the structure reaches its state of near collapse and that plastic hinging occurs simultaneously at predetermined locations to form a collapse mechanism simulating ductile behavior. The actual behavior of a building structure during a strong earthquake is far from that described above, with seismic actions having a vibratory character and plastic hinging occurring rather randomly. However, by applying the capacity design concept in the design of the flexural members of the structure, it is believed that the structure will possess adequate seismic resistance, as has been proven in many strong earthquakes in the past[4].

Capacity design is a design method in which elements of a structural system are chosen, suitably designed and detailed for energy dissipation under severe deformations while all other structural elements are provided with sufficient overstrength so that the chosen means of energy dissipation can be maintained. The critical regions of these members, often termed as plastic hinges, are detailed for inelastic flexural action, and shear failure is inhibited by a suitable strength differential. All other structural elements are then protected against actions that could cause failure, by providing them with strength greater than that of the potential plastic hinge regions[9].

A feature in the capacity design concept is the ductility level of the structure, expressed by the displacement ductility factor or briefly ductility factor. This is the ratio of the lateral displacement of the structure due to the design earthquake at near collapse and that at the point of first yielding[1].

The capacity design philosophy for earthquake resistant design involves the following procedures[11]:

- a) Clearly define plastic hinge regions within the structure and the structural mechanism which is to be employed within those regions. These are usually flexural mechanisms located within the end zones of beam elements, the base region of cantilever shear walls, the link beams of coupled shear walls and the mid-span region of eccentrically braced frames.
- b) Detail the capacity of the elements within the plastic hinge zones so that their dependable strength matches, as closely as possible, the combined gravity and earthquake loads derived from the conventional structural analysis.

c) Detail these zones so that other, less desirable post-elastic mechanisms are suppressed. When the flexural behavior is chosen as the preferred post-elastic mechanism, the probable overstrength of the section within the plastic hinge zone (including the strength contribution which may be made to the section by secondary components such as slab reinforcing) needs to be considered, with the dependable strength of other, less desirable failure mechanisms having a dependable strength well in excess of this. The following are considered unreliable (and thus undesirable) failure mechanisms with the recommended precautionary measures being indicated within brackets;

- the loss of anchorage (by avoiding bar laps within the plastic hinge zones);
- shear failure within the element (by the provision of closely spaced transverse reinforcing stirrups within the plastic hinge zone so as to assure diagonal shear crack development within the concrete is inhibited.);
- buckling of the flexural compression steel (by closely spacing well anchored lateral ties) and
- loss of axial load carrying capacity (by properly confining concrete column cores and avoiding buckling of main column reinforcing steel by using closely spaced, well anchored lateral ties).

d) The resulting structure has now effectively been tuned by the design process so that the post-elastic deformations will only occur within the well detailed plastic hinge zones of the structure, and that all other potential failure mechanisms have been suppressed, regardless of the intensity of ground shaking.

The adoption of the capacity design philosophy will ensure that the principle objective of earthquake resistant design, namely avoidance of building collapse under severe earthquake attack, is satisfied. It has been reasonably argued that this is indeed the only method which can assure compliance with this fundamental performance objective, particularly, considering the uncertainties of the random character of earthquake induced ground motion, the large influence that local site effects will have on this motion, and the ongoing rather crude engineering modeling methods available to simulate the actual post-elastic dynamic response of the building to that motion[11].

The process requires the margin of strength necessary for non-yielding elements to ensure that their behavior remains elastic. Such forces are associated with the development of the overstrength of potential plastic hinges. The strategy leads to the establishment of a suitable strength or capacity hierarchy between components of the total system. The chain (Professor Paulay's "ductile chain"[10]) shown below in Figure 2.1 best illustrates the principle of capacity design. The idea is that the ductile link yields at a load that is well below the failure load of the brittle links. Although most building structures are

somewhat less straightforward than the chain used in Professor Paulay's example, one of the great strengths of the capacity design principle is that it relies on simple static analysis to ensure good performance, and is not dependent on the vagaries of a complex dynamic calculation[10].

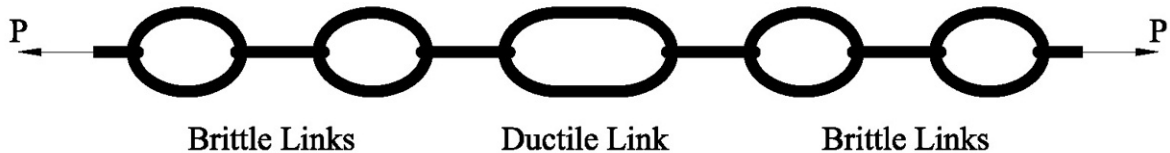


Figure 2.1: Chain with Brittle and Ductile Links[9]

If the brittle links are designed to have the same nominal strength as the ductile link, the randomness of strength variation between all links, including the ductile link, would imply a high probability that failure would occur in a brittle link and the chain would have no ductility. Failure of the brittle links is prevented by increasing their strength in excess of the maximum feasible strength of the weak link.

$$P_{brittle} = \phi_o * P_{ductile} \quad (2.1)$$

where ϕ_o is the overstrength factor > 1

The three important examples of capacity design are[10]:

- Ensuring that columns are stronger than beams in moment frames,
- Ensuring that concrete beams are stronger in shear than in flexure and
- Ensuring that steel braces buckle before columns.

Plastic mechanisms in reinforced concrete structures must rely on flexure as the source of energy dissipation. Therefore, with few exceptions, mechanisms associated with inelastic deformations originating from shear, transfer of bond between reinforcement and concrete, sliding and instability of members, must be definitively suppressed. The choice of the designer involves thus the selection of plastic hinges in beams or columns that enable a kinematically admissible complete mechanism to be developed in the given structural system. An important aim in this selection is that for a given global or system displacement ductility demand, μ_Δ , the associated curvature ductility at plastic hinges remain within proven limits.[21]

The above considerations are illustrated in a series of diagrams in Figure 2.2 which show desirable or acceptable mechanisms and those that are to be avoided. The same

ultimate displacement, Δ_u , has been assumed for all the example systems shown. The advantages of strong-column-weak-beam mechanisms in ductile multistorey frames are well established. When columns are provided with sufficient strength, plastic hinge formation in them can be avoided in all storeys above level 2, as shown in Figure 2.2(a). When columns are adequately detailed for hinge formation, the widely used system shown in Figure 2.2(b) may also be adopted. It is to be noted, however, that the possibility of simultaneous hinge formation at both ends of columns in any storey, such as in Figure 2.2(c), generally termed as soft storey, must not be permitted. It is evident that rotational ductility demands in the hinging columns, such as shown in Figure 2.2(c), may become excessive.[21]

When the exterior columns of a frame, absorbing actions from one beam only, are made strong enough to ensure that a soft storey cannot develop, the formation of simultaneous plastic hinges at both ends of all interior columns [Figure 2.2(d)] is acceptable, provided that all potential plastic hinges in these columns are detailed accordingly. A similar situation, usually encountered when long span beams need to be used, arises when walls [Figure 2.2(e)] ensure that a soft storey cannot develop. When structural walls, which are interconnected by adequately designed diaphragms are used to provide the necessary lateral force resistance, it is preferable to restrict the formation of plastic hinges to the base of the walls, acting as cantilevers, as shown in Figure 2.2(f). This can be readily achieved when the walls above the plastic zone are provided with sufficient reserve flexural capacity. Figure 2.2(g) shows the desirable and readily achievable plastic mechanism in a pair of structural walls interconnected by coupling beams which are specially reinforced to enable large member ductilities to be developed. This system is capable of delivering excellent energy dissipating capacity in a very stable manner. Frames in which soft storeys, as shown in Figure 2.2(h), can develop should only be used when the displacement ductility demand, μ_Δ , assigned to them is restricted. When long span beams need to be provided, the gravity load requirements for these may be more severe than those associated with seismic demands. Hence it may be difficult and even irrational to design interior columns under seismic attack to be stronger than the beams. Figure 2.2(i) depicts a gravity load dominated frame of this type.[21]

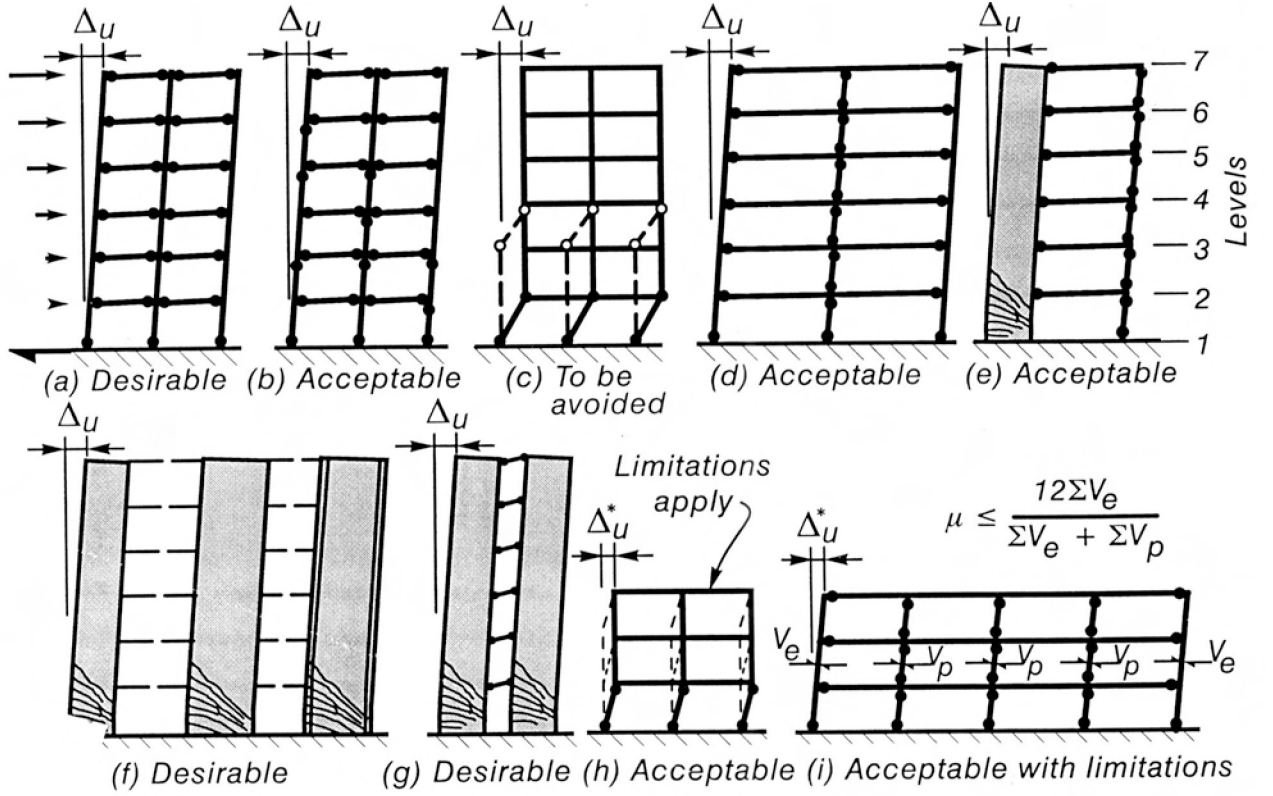


Figure 2.2: Plastic Mechanisms in Multistorey Buildings [21]

2.3.2 Capacity Design for RCPFs

The principle outlined for the example chain in Figure 2.1 can be extended to encompass the more complex design of a large structure (e.g. a multistorey building). The procedure uses the following major steps[9]:

1. A kinematically admissible plastic mechanism is chosen.
2. The mechanism chosen should be such that the necessary overall displacement ductility can be developed with the smallest inelastic rotation demands in the plastic hinges (Figure 2.3).
3. Once a suitable plastic mechanism is selected, the regions for energy dissipation (i.e plastic hinges) are determined with a relatively high degree of precision.
4. Parts of a structure intended to remain elastic in all events are designed so that under maximum feasible actions corresponding to overstrength in the plastic hinges, no elastic deformations should occur in those regions. Therefore, it is immaterial whether the failure of regions, intended to remain elastic, would be ductile or brittle. The actions originating from plastic hinges are those associated with the overstrength of these regions. The required strength of all other regions is then in

excess of the strength demand corresponding to the overstrength of relevant plastic hinges.

5. A clear distinction is made with respect to the nature and quality of detailing for potentially plastic regions and those which are to remain elastic in all events.

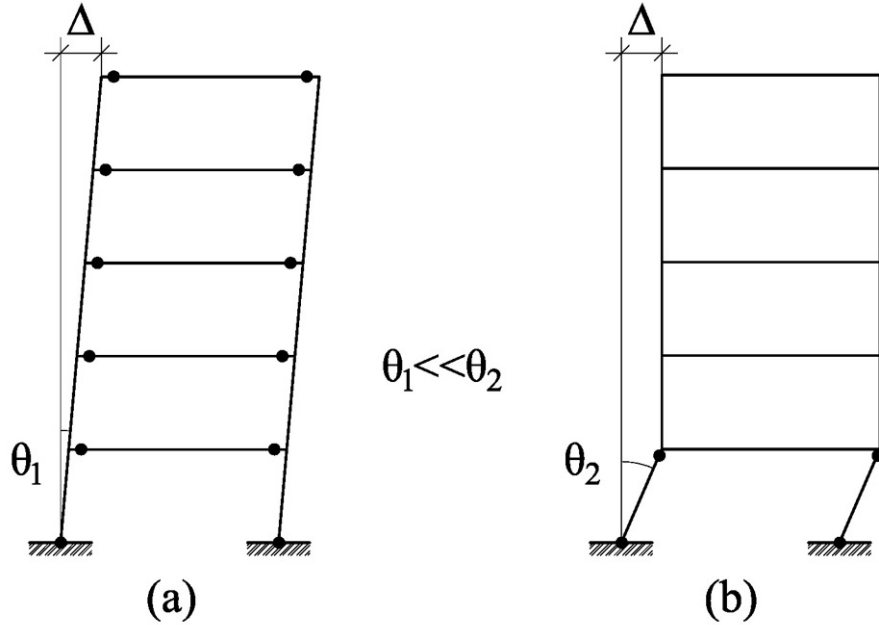


Figure 2.3: Comparison of Energy Dissipating Mechanisms[9]

A comparison of the two example frames in Figure 2.3 shows that for the same maximum displacement Δ at roof level, plastic hinge rotations θ_1 in case (a) are much smaller than those in case (b), θ_2 . Therefore, the overall ductility demand, in terms of the large deflection Δ , is much more readily achieved when plastic hinges develop in all the beams instead of only in the first-storey column. The column hinge mechanism, shown in Figure 2.3(b), also referred to as a soft-storey, may impose plastic hinge rotations, which even with good detailing of the affected regions, would be difficult to accommodate. This mechanism accounts for numerous collapses of framed buildings in recent earthquakes. In the case of the example given in Figure 2.3, the primary aim of capacity design will be to prohibit formation of a softy storey and, as a corollary, to ensure that only the mechanism shown in Figure 2.3(a), can develop[9].

The basics of capacity based design lie on strong column and weak beam concept in which the plastic hinges induced by the seismic forces form at the ends of the beams. The hinging regions are detailed to maintain their shear capacity while the plastic hinges undergo flexural yielding in both positive and negative bending[2]. The failure of a column can affect the stability of the whole building, but the failure of a beam causes localized effect. Therefore, it is better to make beams to be the ductile weak links than

columns[1]. A general rule for all types of frame buildings given in ES EN 1998:2015 is that the moment strength of columns connected to a particular node should be greater than the magnified (by an overstrength factor) moment strength of the beams:

Basic steps for capacity based design of reinforced concrete building frames[22]:

1. Design loads i.e. dead loads, live loads and earthquake loads are calculated.
2. Seismic analyses of the frame for all load combinations specified in the code are done.
3. Beams are designed for maximum sagging and maximum hogging moments. Provided reinforcements are calculated following the norms given in the code used.
4. Columns are designed for the combination of moment and corresponding axial force providing maximum interaction effect i.e. considering the eccentricity.
5. The flexural capacities of the beams under sagging and hogging condition for the provided reinforcements are calculated.
6. The flexural capacity of columns at a joint is compared with actual flexural capacity of joining beams. If the sum of capacities of columns is less than the sum of capacities of beams multiplied by overstrength factor, the column moments should be magnified by the factor (moment magnification factor) by which they are lacking in moment capacity over beams. If the sum of the column moments is greater than sum of beam moments, there is no need to magnify the column moments.
7. Columns are designed for the magnified column moments and the axial force coming on them from the analysis.
8. Shear capacity of beams are calculated on the basis of their actual moment capacities and shear reinforcements are calculated.
9. Similarly shear capacity of columns is calculated on the basis of magnified moment capacities. Then the columns are designed for shear.

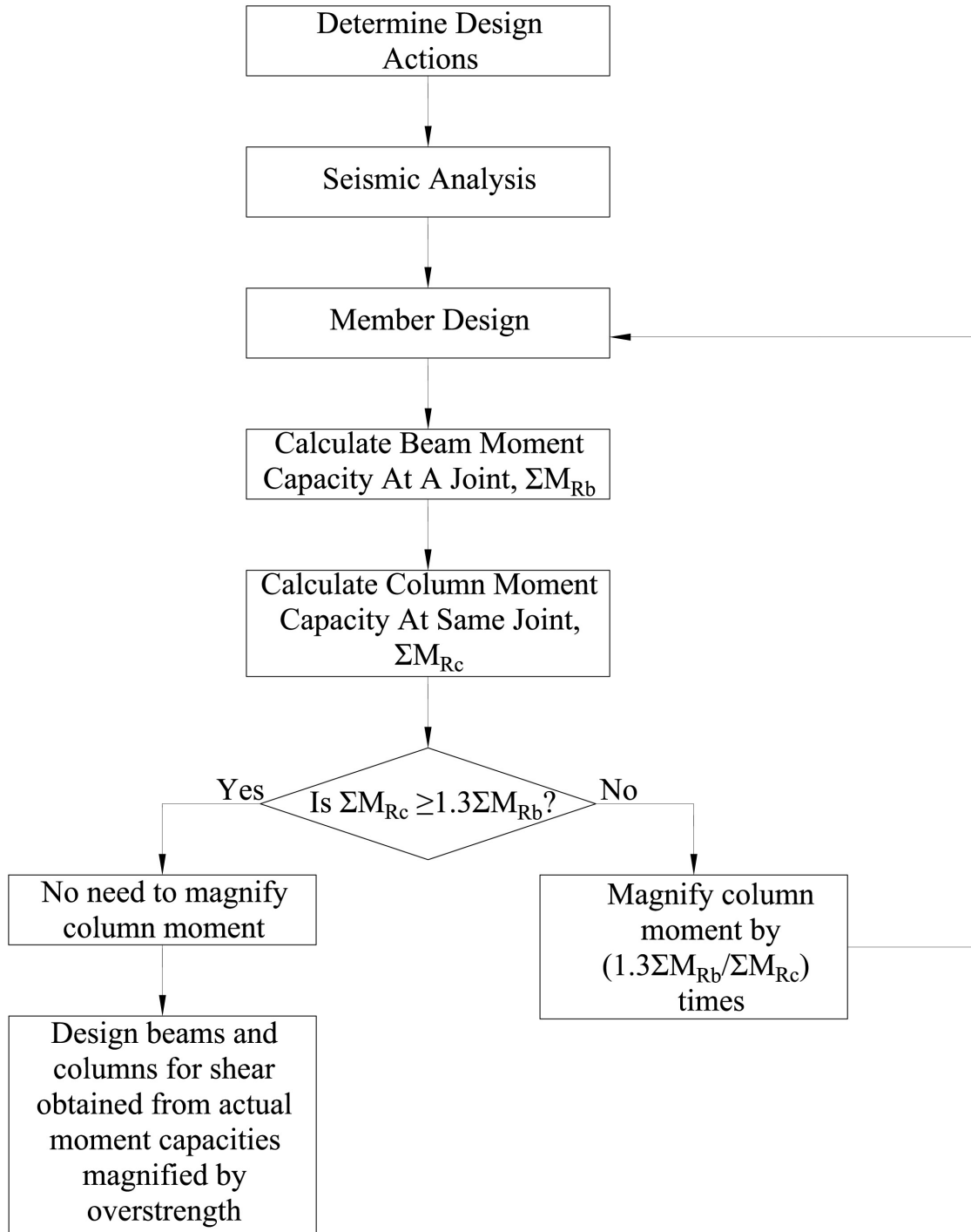


Figure 2.4: Capacity Design Flow Chart

A capacity design approach is likely to assure predictable and satisfactory inelastic response under conditions for which even sophisticated dynamic analyses techniques can yield no more than crude estimates. This is because the capacity-designed structure cannot develop undesirable hinge mechanisms or modes of inelastic deformation, and

is, as a consequence, insensitive to the earthquake characteristics, except insofar as the magnitude of inelastic flexural deformations are concerned. When combined with appropriate detailing for ductility, capacity design will enable optimum energy dissipation by rationally selected plastic mechanisms to be achieved. Moreover, as stated earlier, structures so designed will be extremely tolerant with respect to the magnitudes of ductility demands that future large earthquakes might impose[9].

It is emphasized that capacity design is not analysis technique but a powerful design tool. It enables the designer to “tell the structure what to do” and to desensitize it to the characteristics of the earthquake, which are after all, unknown. It is rational, deterministic, and relatively a simple approach[9]. Capacity design philosophy addresses the problem directly, and obliges the engineer to design the structure in such a way that yielding hinges can only form in pre-determined positions.

2.4 Computer Programming

Computer programs, known as software, are instructions to the computer, telling it what to do. Computers do not understand human languages, so one needs to use computer languages in computer programs. Programming is the creation of a program that is executable by a computer and performs the required tasks[23]. As humans have developed so many languages to communicate among themselves, computer scientists have developed several computer-programming languages to provide instructions to the computer (i.e., to write computer programs)[24].

2.4.1 History[25]

Programmable devices have existed at least as far back as 1206 AD, when the automata of Al-Jazari were programmable, via pegs and cams, to play various rhythms and drum patterns; and the 1801 Jacquard loom could produce entirely different weaves by changing the "program" - a series of pasteboard cards with holes punched in them. However, the first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. In the 1880s Herman Hollerith invented the concept of storing data in machine-readable form. Later a control panel (plugboard) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way; as were the first electronic computers. However, with the concept of the stored-program computers introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.

Machine code was the language of early programs, written in the instruction set of the

particular machine, often in binary notation. Assembly languages were soon developed that let the programmer specify instruction in a text format, (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. However, because an assembly language is little more than a different notation for a machine language, any two machines with different instruction sets also have different assembly languages.

High-level languages allow the programmer to write programs in terms that are more abstract, and less bound to the underlying hardware. They harness the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula directly (e.g., $Y = X*2 + 5*X + 9$). FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957 and many other languages were soon developed - in particular, COBOL aimed at commercial data processing, and Lisp for computer research. Programs were mostly still entered using punched cards or paper tape. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Text editors were developed that allowed changes and corrections to be made much more easily than with punched cards.

There are hundreds of programming languages, which can be used to write computer programs and following are a few of them: C, C++, C#, FORTRAN, Java, Oracle, R, Python, PHP, Perl, R, Ruby, Visual Basic, etc...

2.4.2 Programming in Java

Java is a high-level programming language, originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. Therefore, Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java is guaranteed to be Write Once, Run Anywhere (WORA); providing no-cost run-times on popular platforms. In addition, it is free and open source[26].

Java has become enormously popular. Its rapid rise and wide acceptance can be traced to its design characteristics. Java is [26]:

- Object Oriented: In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- Platform Independent: Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

- Secure: With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- Architecture-neutral: Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java run-time system.
- Portable: Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- Robust: Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and run-time checking.
- Multi-threaded: With Java's multi-threaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- Interpreted: Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- High Performance: With the use of Just-In-Time compilers, Java enables high performance.
- Distributed: Java is designed for the distributed environment of the internet.
- Dynamic: Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

2.4.3 Why Java?

Java is a full-featured, general-purpose programming language that can be used to develop robust mission-critical applications. Today, it is employed not only for Web programming, but also for developing standalone applications across platforms on servers, desktops, and mobile devices. It was used to develop the code to communicate with and control the robotic rover on Mars. Many companies that once considered Java to be more hype than substance are now using it to create distributed applications accessed by customers and partners across the Internet. For every new project being developed today, companies are asking how they can use Java to make their work easier[23].

Java is an object-oriented programming language used universally in a multitude of sectors from web-scripting to operating system development, artificial intelligence research, infrastructure control systems and consumer application production. Currently, Java runs on more than 3.5 billion devices including mainframes, smart phones, computers, cars, home appliances, communication platforms and several other electronic equipment[27].

Unlike other programming languages which run on the environment of the operating system they are installed on, Java runs exclusively on a virtual machine with pre-allocated resources known as the Java Virtual Machine, which is instantiated alongside the operating system when a program runs[27]. This feature allows a program written in Java to compile/run on any computer/OS capable of supporting the Java Virtual Machine (JVM). Another feature of the JVM would be customizable resource allocation (i.e. total amount of RAM and bandwidth allocated by the OS can be modified by the user/programmer). In addition, Java comes with a plethora of external and stock libraries for problems ranging from user interface design to numerical computations. Perhaps the most alluring feature of Java, in addition to the aforementioned features, would be its object-oriented paradigm. This allows for either a top-down or a bottom-up design to be feasible[27].

And specifically Java was selected for this research because the structural analysis software used (HELIX - See Section 3.3.3) was developed by Java, hence, enabling seamless integration of the newly written capacity design software. Furthermore, this research work has chosen Java for its ease of integration with other applications, its speed and customizability in using hardware resources, and for its unparalleled integration with XML for writing Graphic User Interfaces (GUIs).

Chapter 3

Research Methods, Materials and Procedures

3.1 General

This particular research is concerned with capacity design of earthquake resistant reinforced concrete plane frames. It attempts to simplify the process by the development of software. For this end, it deals with a few primary concerns. First is the thorough study of the design method involving real world worked examples. Second, all procedures from the capacity design method need to be implemented in easy-to-use software. Third, the developed software needs to be benchmarked and optimized to reduce time and memory requirements. Fourth, the results of the software need to be verified through examples. The following subsections outline some of the methods and tools required to accomplish these.

3.2 Methods and Materials

3.2.1 Literature

Several materials will be reviewed in developing the software among which are: books and/or online references on programming, on structural analysis and design, earthquake analysis, earthquake resistant design and capacity based design. Special emphasis to be given to codes pertaining to reinforced concrete design and seismic design particularly ES EN 1992:2015 and ES EN 1998:2015.

3.2.2 Software and Utilities

In developing the software the following software and utilities will be required;

- An integrated development environment (IDE); a program that provides tight-knit components with similar user interfaces and by so doing maximizes productivity. It presents a single program in which all development is done i.e. features for authoring, modifying, compiling, deploying and debugging software.
- Graphical user interface (GUI) framework that allows users to interact through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. Users perform actions in a GUI through direct manipulation of the graphical elements. It generally enhances the efficiency and ease of use of a given software.
- A fast lightweight structural analysis program for the determination of design actions on the frames. Thus, making analysis results ready for capacity design.

3.3 Software Design and Result Verification

3.3.1 Design Assumptions

In an attempt to achieve the objectives outlined in chapter 1 section 1.5, certain assumptions need to be made. They are stated as follows; all analyses are two dimensional, all members have rectangular cross-sections and inclined or curved members will not be addressed. Moreover, assumptions and procedures pertaining to capacity design according to Ethiopian Standards (especially ES EN 1992:2015 and ES EN 1998:2015[28, 5]) will be followed.

3.3.2 Design Procedure

Since the capacity design approach requires primary analysis and design results, HELIX; software developed by Mr. Elias Yilma[27] for his thesis (Cost Optimization of Reinforced Concrete Frames Using Genetic Algorithms), shall be put to use. This software was written in Java, the same programming language intended to be utilized for capacity design, making software integration possible. However, the aforementioned software does not carry out earthquake analysis and frame design. New pieces of code in addition to capacity design codes need to be added for earthquake load computation, earthquake analysis and frame design.

3.3.3 HELIX[27]

HELIX was written in Java 8 and is a single-threaded application that can run on any x86/x64 based Intel or AMD processor. HELIX can be used as a structural analysis

software as well as for structural optimization problems. It implements the direct stiffness method for structural analysis.

3.3.3.1 Main Interface and Grid Definition

The primary window is the one that is encountered when the program opens. At the start of the program, the ribbon shown at the left is the grid definition menu. Depending on the topology, empty two and three dimensional grids, can be defined here. In addition, predefined portal frames could also be drawn on the canvas.

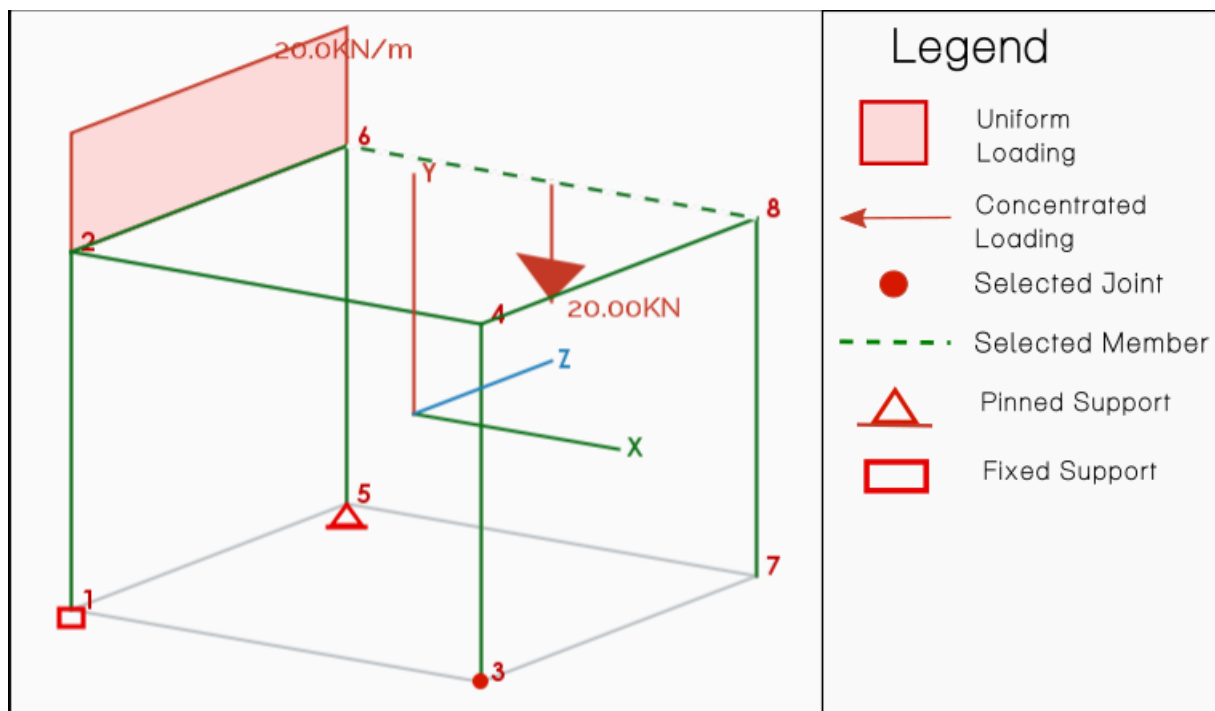


Figure 3.1: Main Canvas Components

3.3.3.2 Ribbon Functions

Through the ribbon, one could access the majority of functionalities featured in the program. Table 3.1 summarizes these functions.

Ribbon	Functions
 Draw Member	Sets the display canvas to drawing mode. In this mode, members could be quickly drawn by connecting points on the grid using the left mouse button.
 Select Joints	Sets the display canvas to joint selection mode. Joints can then be selected individually by placing the cursor near a labeled joint and clicking the left mouse button once. Multiple joints could also be selected by clicking the left mouse button and dragging through the area of interest.
 Select Members	Sets the display canvas to member selection mode. A member can be easily selected by moving the cursor near it, and single clicking using the left mouse button. Multiple members could also be selected by cross- dragging along the members using the left mouse button.
 Material Def.	Displays the material definition menu. Constitutive property data for concrete and reinforcing steel is to be entered here.
 Restraints	Displays the joint restraint options menu. Selected joints can be assigned appropriate restraints for any degree of freedom using this menu.
 Loads	Displays the load definition menu. Load data, such as direction of application, type(distributed or concentrated), magnitude and location is to be specified here.
 Section Def.	Displays the section definition menu. Cross-sectional data such as width and height of members
 Analyze	Performs structural analysis, if all relevant data has been entered, and topology has been defined appropriately.
 Design Member	Computes reinforcement quantities based on previously assigned section information and analysis results.

Table 3.1: Main Ribbon Functions

3.3.3.3 Sub-menus

There are five sub-menus that could be accessed from the main ribbon. These are:

1. **Grid Definition:** the user can enter parameters for defining an empty rectangular grid with regular spacing in this sub-menu. These include basic dimensions such as height and bay width in the horizontal directions. The number of grid lines in each direction can also be specified. If two dimensional frames/grids are required, then

one could set the number of grid lines in the unwanted direction to zero. If a regular portal frame is what's required, then manual drawing of all members can be avoided by selecting the "Generate Frame Elements" check-box. This option generates a portal frame aligned with the grid, and instantiates all necessary topological data. This is useful especially if large portal frames are to be drawn quickly.

2. **Material Definition:** Concrete and reinforcing steel material properties are to be entered here. These data is later utilized in forming the element and structure stiffness matrices. If these values are not known, or haven't been decided upon, then by checking the "Defaults" option, predetermined data could be set.
3. **Section Definition:** Member cross-sectional data are to be entered here and will be useful in the analysis and design of members.
4. **Loads:** Before load data can be entered, members to which the load is to be applied need to be selected. One can select members by clicking on the "Select Members" ribbon. Once selection is complete, load data could be entered. Through the sub-menu, concentrated force and moment magnitudes can be applied by entering the magnitude and relative position of the application point. Distributed load data can be input by selecting the "Distributed" check-box.
5. **Restraints:** Joints can be restrained against the six possible degrees of freedom. Similar to the load sub-menu, the joints sub-menu needs at least one selected joint as input.

Grid Definition...

Number of Stories

Story Height, m

Number of Bays, X

Bay Width, X, m

Number of Bays, Z

Bay Width, Z, m

☐ Generate Frame Elements

Generate Cancel

Materials...

☒ Defaults

Concrete

Conc. Char. Strength(fck), MPa

20

Young's Modulus, Ec, GPa

29

Shear Modulus, G, GPa

11.15

Steel

Steel Yield Strength(fyk), MPa

300

Young's Modulus, Es, GPa

200

Accept Cancel

Sections...

Breadth, B, mm

Height, H, mm

Bottom Reinforcement, mm²

Top Reinforcement, mm²

Accept Cancel

Figure 3.2: Sub-Menus 1

Loads...

☐ DISTRIBUTED

Force, X, KN	Rel Position, X
Force, Y, KN	Rel. Position, Y
Force, Z, KN	Rel. Position, Z
Moment, X, KN.m.	Rel Position, X
Moment, Y, KN.m.	Rel. Position, Y
Moment, Z, KN.m.	Rel. Position, Z

Accept Cancel

Restraints

Displacements

☐ x

☐ y

☐ z

Rotations

☐ x

☐ y

☐ z

Accept Cancel

Figure 3.3: Sub-Menus 2

3.3.4 Earthquake Load Computation and Analysis

Lateral force method of analysis is to be employed for the earthquake load computation as presented in ES EN 1998:2015 Section 4.3.3.2[5]. The steps required are base shear force determination and distribution of horizontal seismic forces.

a) Base Shear Force Determination

$$F_b = S_d(T_1) * m * \lambda \quad (3.1)$$

Where;

F_b : is base shear force.

$S_d(T_1)$: is the ordinate of the design spectrum at period T_1 .

$$S_d(T_1) = a_g * S * \left[\frac{2}{3} + \frac{T_1}{T_B} * \left(\frac{2.5}{q} - \frac{2}{3} \right) \right] \dots 0 \leq T_1 \leq T_B \quad (3.2)$$

$$S_d(T_1) = a_g * S * \eta * \frac{2.5}{q} \dots T_B \leq T_1 \leq T_C \quad (3.3)$$

$$S_d(T_1) = \begin{cases} a_g * S * \frac{2.5}{q} * \left[\frac{T_C}{T_1^2} \right] & \dots T_C \leq T_1 \leq T_D \\ \geq \beta * a_g & \end{cases} \quad (3.4)$$

$$S_d(T_1) = \begin{cases} a_g * S * \frac{2.5}{q} * \left[\frac{T_C T_D}{T_1^2} \right] & \dots T_D \leq T_1 \\ \geq \beta * a_g & \end{cases} \quad (3.5)$$

S , T_B , T_C and T_D as given in ES EN 1998:2015 Table 3.2 or Table 3.3 and shown below.

Table 3.2: Values of Parameters Describing the Recommended Type 1 Elastic Response Spectra[5]

Ground Type	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.00	0.15	0.40	2.00
B	1.20	0.15	0.50	2.00
C	1.15	0.20	0.60	2.00
D	1.35	0.20	0.80	2.00
E	1.40	0.15	0.50	2.00

Table 3.3: Values of Parameters Describing the Recommended Type 2 Elastic Response Spectra[5]

Ground Type	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.00	0.05	0.25	1.20
B	1.35	0.05	0.25	1.20
C	1.50	0.10	0.25	1.20
D	1.80	0.10	0.30	1.20
E	1.60	0.05	0.25	1.20

T_1 : is the fundamental period of vibration of the building for lateral motion in the direction considered.

$$T_1 = C_t * H^{\frac{3}{4}} \quad (3.6)$$

C_t : is 0.085 for moment resistant space steel frames, 0.075 for moment resistant space concrete frames and for eccentrically braced steel frames and 0.050 for all other structures.

H : is the height of the building in meters from the foundation or from the top of a rigid basement.

α_g : is the design ground acceleration on type A ground as presented in ES EN 1998:2015, Annex D, Table D1.

$$\alpha_g = \gamma_1 * \alpha_{gR} \quad (3.7)$$

γ_1 : is the importance factor based on the importance classes.

Table 3.4: Importance Classes and Importance Factor[5]

Importance Class	Buildings	Importance Factor (γ_1)
I	Buildings of minor importance for public safety e.g. agricultural buildings, etc.	0.80
II	Ordinary buildings, not belonging in the other categories.	1.00
III	Buildings whose seismic resistance is of importance in view of the consequences associated with a collapse, e.g. schools, assembly halls, cultural institutions etc.	1.20
IV	Buildings whose integrity during earthquakes is of vital importance for civil protection, e.g. hospitals, fire stations, power plants, etc.	1.40

α_{gR} : is the reference peak ground acceleration on type A ground.

$$\alpha_{gR} = \alpha_o * g \quad (3.8)$$

α_o : is the bedrock acceleration ratio.

Table 3.5: Bedrock Acceleration Ratio[5]

Zone	5	4	3	2	1	0
α_o	0.20	0.15	0.10	0.07	0.04	0

g : is the acceleration of gravity $\approx 9.81m/s^2$.

η : is the damping correction factor with a reference value of $\eta = 1$ for 5% viscous damping.

q : is behaviour factor.

$$q = q_o * k_w \geq 1.5 \quad (3.9)$$

q_o : is the basic value of the behaviour factor, dependent on the type of the structural system and on its regularity in elevation. Here is how it's value can be computed for a frame system that is regular in elevation.

Table 3.6: Basic Behaviour Factor Computation, q_o [5]

Ductility Class	Basic Behaviour Factor, q_o
Low (DCL)	1.5
Medium (DCM)	$3 * \frac{\alpha_u}{\alpha_1}$
High (DCH)	$4.5 * \frac{\alpha_u}{\alpha_1}$

$\frac{\alpha_u}{\alpha_1}$: is the multiplication factor equal to 1.3 for multi-storey multi-bay frame

k_w : is the factor reflecting the prevailing failure mode in structural systems with walls which is equal to 1 for frames.

β : is the lower bound factor for the horizontal design spectrum with a recommended value of 0.2.

m : is the total mass of the building above the foundation or above the top of a rigid basement.

The mass taken when calculating the earthquake loads should comprise the full permanent (or dead) load plus the variable (or live) load multiplied by a factor $\psi_{E,i}$.

$$m = \sum G_{k,j} + \sum \psi_{E,i} Q_{k,i} \quad (3.10)$$

$\sum G_{k,j}$: is sum of permanent loads.

$\sum Q_{k,i}$: is sum of variable actions.

$\psi_{E,i}$: is the combination coefficient for variable action i.

$$\psi_{E,i} = \varphi * \psi_{2,i} \quad (3.11)$$

φ given in ES EN 1992:2015 Table 4.2 and $\psi_{2,i}$ in ES EN 1990:2015 Annex A1, Table A1.1.

λ : is the correction factor, the value of which is equal to: $\lambda = 0.85$ if $T_1 < 2T_C$ and the building has more than two storeys, or $\lambda = 1.0$ otherwise.

b) Distribution of Horizontal Seismic forces

$$F_i = F_b * \frac{z_i m_i}{\sum z_j m_j} \quad (3.12)$$

Where;

z_i, z_j are the heights of the masses $m_i m_j$ above the level of application of the seismic action (foundation or top of a rigid basement).

A user input mechanism shall be put in place in order to allow feeding of necessary analysis and design data to the program. The primary results comprise of moments, shear forces, axial force and required amount of reinforcements for moment and shear resistance.

3.3.5 Beam Design[29]

Reinforced concrete beam design consists primarily of producing member details which will adequately resist the ultimate bending moments, shear forces and torsional moments. Since this particular research is concerned with plane frames torsional moments have been excluded.

Beam Design Flow Chart

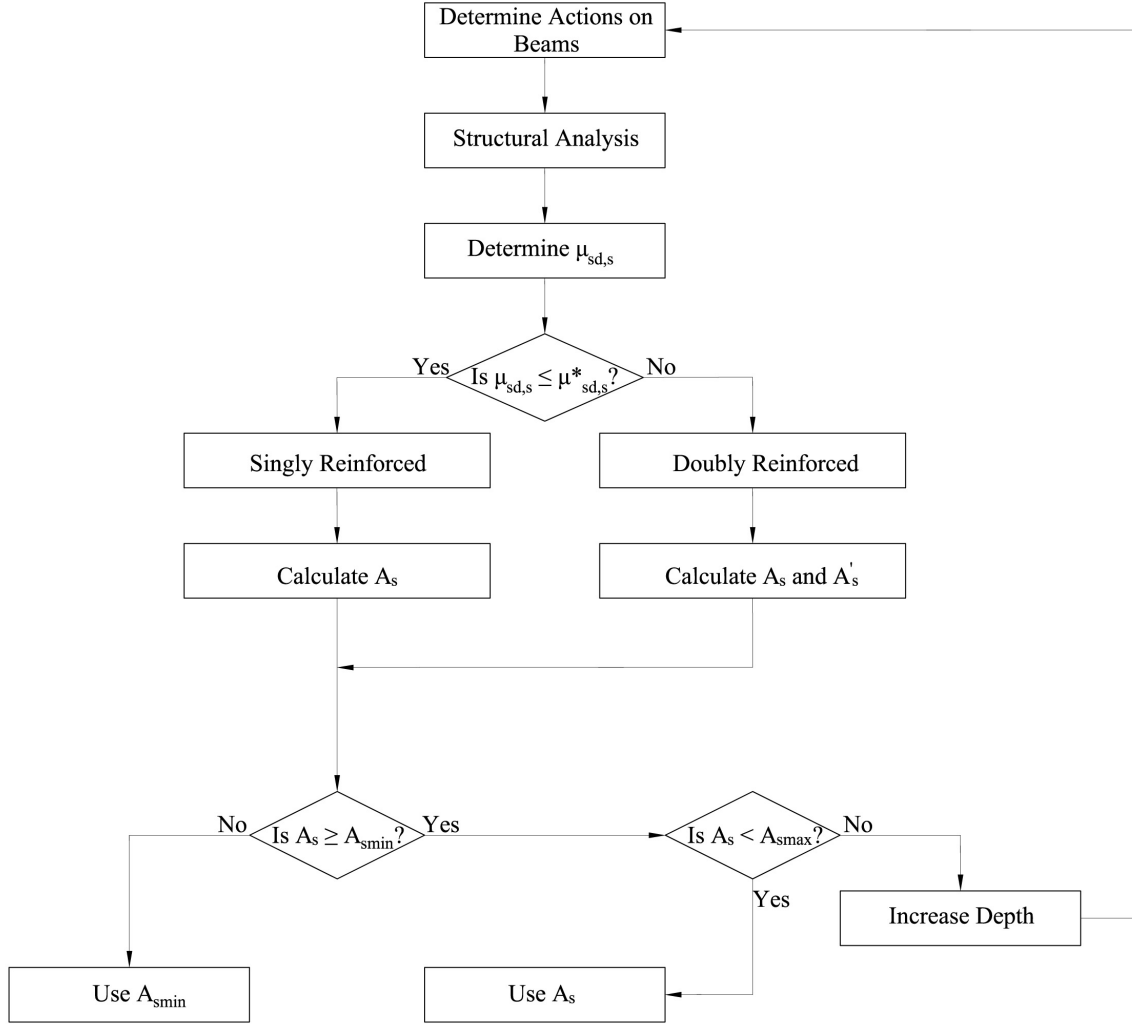


Figure 3.4: Beam Design Flow Chart

3.3.5.1 Design for Flexure: Singly reinforced rectangular sections with moment redistribution ($\mu_{sd,s} \leq \mu_{sd,s}^*$)

The design procedure using the equations based on the UK Annex to EC2 is:

1. Calculate

$$\mu_{sd,s} = \frac{M_{sd}}{bd^2 f_{cd}} \quad (3.13)$$

2. Calculate

$$\mu_{sd,s}^* = 0.454(\delta - 0.4) - 0.182(\delta - 0.4)^2 \quad (3.14)$$

If $\mu_{sd,s} < \mu_{sd,s}^*$, singly reinforced section.

3. Calculate

$$z = d \left[0.5 + \sqrt{0.25 - \frac{\mu_{sd,s}}{1.134}} \right] \quad (3.15)$$

4. Calculate

$$A_s = \frac{M_{sd}}{f_{yd}z} \quad (3.16)$$

5. Check that the area of steel provided is within the maximum and minimum limits required. This is because an excessive amount of reinforcement usually indicates that a member is undersized and it may also cause difficulty in fixing the bars and pouring the concrete. Therefore the code stipulates (except at laps) that:

$$\frac{100A_s}{A_c} \leq 4\% \quad (3.17)$$

On the other hand too little reinforcement is also undesirable therefore:

$$\frac{100A_s}{b_t} \geq 0.13\% \quad (3.18)$$

Where;

- M_{sd} : is the design bending moment.
- A_s : is the area of tensile steel.
- A_c : is the area of concrete= $b * h$ for a rectangular section.
- b : is the width of the beam.
- b_t : is the mean width of the beam's tension zone= b for a rectangular section.
- h : is the depth of the beam.
- d : is the effective depth of the section.
- x : is the neutral axis depth.
- f_{cd} : is the design value of concrete compressive strength.
- f_{yd} : is the design yield strength of reinforcement.
- δ : is the moment redistribution amount.
- z : is the lever arm.

3.3.5.2 Design for Flexure: Doubly reinforced rectangular sections with moment redistribution ($\mu_{sd,s} > \mu_{sd,s}^*$)

Steps for the design of doubly reinforced sections, i.e. those with tension and compression reinforcement, are:

1. Calculate

$$x = x_{bal} \leq (\delta - 0.4)d \quad (3.19)$$

2. Calculate

$$\mu_{sd,s} = \frac{M_{sd}}{bd^2 f_{cd}} \quad (3.20)$$

3. Calculate

$$\mu_{sd,s}^* = 0.454(\delta - 0.4) - 0.182(\delta - 0.4)^2 \quad (3.21)$$

If $\mu_{sd,s} > \mu_{sd,s}^*$, doubly reinforced section i.e compression steel is required.

4. Calculate the area of compression steel from

$$A'_s = \frac{(\mu_{sd,s} - \mu_{sd,s}^*) f_{cd} b d^2}{f_{sc} (d - d')} \quad (3.22)$$

where f_{sc} is the stress in the compression steel

- If $\frac{d'}{x} \leq 0.38$ the compression steel has yielded and $f_{sc} = 0.87 f_{yk}$.
- If $\frac{d'}{x} > 0.38$ then the strain ε_{sc} in the compression steel must be calculated from the proportions of the strain diagram and $f_{sc} = E_s \varepsilon_{sc} = 200 * 10^3 \varepsilon_{sc}$.

$$\varepsilon_{sc} = \frac{0.0035 * (x - d)}{x} \quad (3.23)$$

5. Calculate the area of tension steel from

$$A_s = \frac{\mu_{sd,s}^* f_{cd} b d^2}{f_{yd} z_{bal}} + A'_s \frac{f_{sc}}{f_{yd}} \quad (3.24)$$

where $z_{bal} = d - (\frac{0.8x_{bal}}{2})$

6. Check for the areas of steel required and the areas provided that

$$(A'_{sprov} - A'_{sreq}) \geq (A_{sprov} - A_{sreq}) \quad (3.25)$$

This is to ensure that the depth of the neutral axis has not exceeded the maximum value of x_{bal} by providing an over-excess of tensile reinforcement.

7. Check that the area of steel provided is within the maximum and minimum required as shown in 3.3.5.1.

Where;

- d' : is the depth to compression reinforcement.
- x : is the neutral axis depth.
- f_{sc} : is the stress in the compressive steel.
- E_s : is the modulus of elasticity of the reinforcing steel.
- ε_{sc} : is the steel strain compression.
- A'_s : is the area of compression steel.
- All the rest as defined for the singly reinforced beam.

3.3.6 Column Design[29]

Columns in a structure carry the loads from the beams and slabs down to the foundations, and therefore they are primarily compression members, although they may also have to resist bending forces due to the continuity of the structure. For the analysis of a section subjected to an axial load plus bending, a direct solution of the equations that determine the areas of reinforcement can be very laborious and impractical. It is recommended that design charts or computers be employed to facilitate the routine design of column sections. Slenderness, and the risk of lateral deflections leading to buckling, is an important consideration in determining failure modes and special requirements will apply to the design of slender columns.

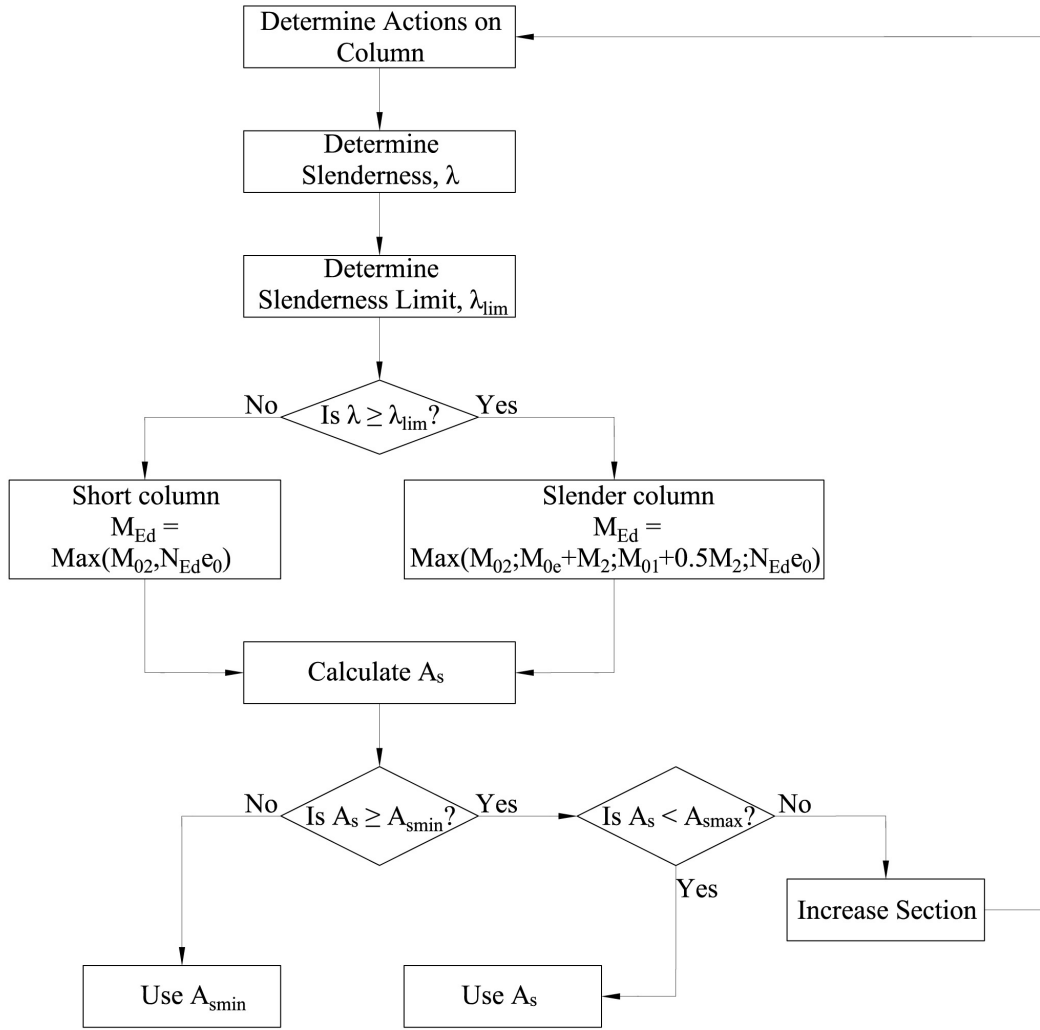


Figure 3.5: Column Design Flow Chart

3.3.6.1 Column Classification (Short or Slender)

The slenderness ratio λ of a column bent about an axis is given by

$$\lambda = \frac{l_o}{\sqrt{i}} = \frac{l_o}{\sqrt{\frac{I}{A}}} \quad (3.26)$$

where

- l_o : is the effective height of the column.
- i : is the radius of gyration about the axis considered, $i = \sqrt{\frac{I}{A}}$.
- I : is the second moment of area of the section about the axis.
- A : is the cross-sectional area of the column.

The effective height of a column, l_o , is the height of a theoretical column of pinned at both ends having the same section and elastic buckling (resistance) load as the original column. This depends on the degree of fixity at each end of the column, which itself depends on the relative stiffnesses of the columns and beams connected to either end of the column under consideration. It can be determined from the larger of the two formulae for frames.

$$l_o = \max \left\{ \begin{array}{l} l \sqrt{(1 + 10 \frac{k_1 * k_2}{k_1 + k_2})} \\ l(1 + \frac{k_1}{1+k_1}) * (1 + \frac{k_2}{1+k_2}) \end{array} \right. \quad (3.27)$$

In the above formulate, l is the clear height of the columns between end restraints. And k_1 and k_2 are the relative flexibilities of the rotational restraints at ends ‘1’ and ‘2’ of the column respectively. At each end k_1 and k_2 should not be less than 0.1 and can be taken as[30]:

$$k = \frac{ColumnStiffness}{\sum BeamStiffness} = \frac{(\frac{EI}{l})_{column}}{\sum 2(\frac{EI}{l})_{beam}} = \frac{(\frac{I}{l})_{column}}{\sum 2(\frac{I}{l})_{beam}} \geq 0.1 \quad (3.28)$$

It is assumed that any column above or below the column under consideration does not contribute anything to the rotational restraint of the joint and that the stiffness of each connecting beam is taken as $\frac{2EI}{l}$ to allow for cracking effects in the beam.

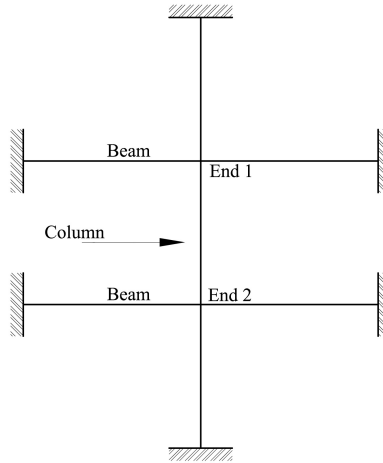


Figure 3.6: Effective Column Height Calculation[29]

ES EN 1992:2015 places an upper limit on the slenderness ratio of a single member below which second order effects may be ignored. This limit is given by:

$$\lambda_{lim} = \frac{20ABC}{\sqrt{n}} \quad (3.29)$$

where:

- $A = \frac{1}{(1+0.2\phi_{ef})}$
- $B = \sqrt{1 + 2w}$
- $C = 1.7 - r_m$
- ϕ_{ef} = effective creep ratio (if not known A can be taken as 0.7).
- $w = \frac{A_s f_{yd}}{A_c f_{cd}}$ (if not known B can be taken as 1.1).
- f_{yd} = the design yield strength of the reinforcement.
- f_{cd} = the design compressive strength of the concrete.
- A_s = the total area of longitudinal reinforcement.
- $n = \frac{N_{Ed}}{A_c f_{cd}}$
- N_{Ed} = the design ultimate axial load in the column.
- $r_m = \frac{M_{01}}{M_{02}}$ (if r_m not known then C can be taken as 0.7).
- M_{01}, M_{02} are the first order moments at the end of the column with $|M_{02}| \geq |M_{01}|$.

Whilst use of the default values for A and B are unlikely to introduce major errors, C may have a large effect and should be calculated wherever possible. The following conditions apply to the value of C :

- If the end moments, M_{01} and M_{02} , give rise to tension on the same side of the column r_m should be taken as positive from which it follows that $C \leq 1.7$.
- If the converse to (a) is true, i.e the column is in a state of double curvature, then r_m should be taken as negative from which it follows that $C > 1.7$.
- For braced members in which the first order moments arise only from transverse loads or imperfections; C can be taken as 0.7.
- For unbraced members; C can be taken as 0.7.

If the actual slenderness ratio, λ , is less than the calculated value of λ_{lim} then the column can be treated as short. Otherwise the column must be treated as slender and second order effects must be accounted for in the design of the column.

3.3.6.2 Design of Short Columns ($\lambda < \lambda_{lim}$)

Columns with $\lambda < \lambda_{lim}$ are classified to be short columns. The area of longitudinal steel for these columns is determined by:

1. using design charts or constructing M–N interaction diagrams
2. a solution of the basic design equations, or
3. an approximate method.

Design charts are usually used for columns having a rectangular or circular cross-section and a symmetrical arrangement of reinforcement, but interaction diagrams can be constructed for any arrangement of cross-section. The basic equations or the approximate method can be used when an unsymmetrical arrangement of reinforcement is required, or when the cross-section is non-rectangular.

Whichever design method is used, ES EN 1992:2015 requires that geometric imperfections in the structure should be taken into account. In the case of a single column it is assumed that the geometric imperfections will result in the column being slightly out of plumb such that the axial force acting at each end of the column will be eccentric to the other end by an amount taken as $\frac{l_o}{400}$, where l_o is the effective height of the column. This will result in an additional design moment of $N_{Ed}(\frac{l_o}{400})$. Furthermore, a column should not be designed for a moment less than $N_{Ed} * e_{min}$, where e_{min} is the larger of $\frac{h}{30}$ or 20mm where h is the overall size of the column cross-section in the plane of bending.

The design of a section subjected to bending plus axial load should deal with the analysis of the cross-section. The basic equations derived for a rectangular section and with a rectangular stress block are:

$$N = F_{cc} + F_{sc} + F_s = f_{cd}bs + f_{sc}A'_s + f_sA_s \quad (3.30)$$

$$M = F_{cc}(\frac{h}{2} - \frac{s}{2}) + F_{sc}(\frac{h}{2} - d') - F_s(d - \frac{h}{2}) \quad (3.31)$$

Where;

- N : is the section axial load resistance.
- M : is the section moment resistance.
- s : is the depth of the stress block = $0.8x$; x is the neutral axis depth.
- A'_s : is the area of longitudinal reinforcement in the more highly compressed face.
- A_s : is the area of reinforcement in the other face.

- f_{sc} : is the stress in reinforcement A'_s .
- f_s : is the stress in reinforcement A_s , negative when tensile.

For a symmetrical arrangement of reinforcement ($A'_s = A_s$), equations 3.30 and 3.31 can be rewritten after rearranging in the following form[30]:

$$\frac{A_{sN}}{2} = \frac{N - f_{cd}bs}{f_{sc} + f_s} \quad (3.32)$$

$$\frac{A_{sM}}{2} = \frac{N - [f_{cd}bs * (\frac{h}{2} - \frac{s}{2})]}{\frac{f_{sc}h}{2} - f_{sc}d' - f_s d + \frac{f_s h}{2}} \quad (3.33)$$

These equations are not suitable for direct solution and the design of a column with symmetrical reinforcement in each face is best carried out using design charts[29]. Then the calculated reinforcement amount, A_s , is checked against the minimum, $A_{smin} = 0.002 * A_c$, and maximum, $A_{smax} = 0.04 * A_c$, allowed quantities.

3.3.6.3 Design of Slender Columns ($\lambda \geq \lambda_{lim}$)

A column is classified as slender if the slenderness ratio about either axis exceeds the value of λ_{lim} . A slender column with must be designed for an additional moment caused by its curvature at ultimate conditions. ES EN 1992:2015 identifies three different approaches to designing slender columns:

1. A general method based on a non-linear analysis of the structure and allowing for second-order effects that necessitates the use of computer analysis.
2. A second-order analysis based on nominal stiffness values of the beams and columns that, again, requires computer analysis using a process of iterative analysis.
3. The ‘nominal curvature’ method where second-order moments are determined from an estimation of the column curvature. These second-order moments are added to the first-order moments to give the total column design moment.

Additionally, another approach is using the moment magnification factor to account for second-order effects (also used in the ACI code). Here the total design moment, including second order moment, may be expressed as a magnification of the bending moments resulting from a first order analysis.

The third method will be used for the purpose of this research. Accordingly, a slender column should be designed for an ultimate axial load (N_{Ed}) plus an increased moment given by

$$M_t = N_{Ed}e_{tot} \quad (3.34)$$

Where;

- $e_{tot} = e_o + e_a + e_2$
- e_o : is an equivalent first-order eccentricity given by greater of $e_o = 0.6e_{02} + 0.4e_{01}$ or $e_o = 0.4e_{02}$ (where e_{01} and e_{02} are the first-order eccentricities at the two ends of the column and $|e_{02}|$ is greater than $|e_{01}|$).
- e_a : is an accidental eccentricity which accounts for geometric imperfections in the column.
 $e_a = \nu \frac{l_o}{2}$; l_o is the effective column height about the axis considered and $\nu = \frac{1}{100\sqrt{l}} > \frac{1}{200}$ with l being the height of the column in meters. But, conservatively, e_a can be estimated from $e_a = \frac{l_o}{400}$.
- e_2 : is the second-order eccentricity.

The second-order eccentricity e_2 is an estimate of the deflection of the column at failure and is given by the equation:

$$e_2 = K_\varphi K_r \frac{l_o^2}{\pi^2} \left(\frac{\varepsilon_{yd}}{0.45d} \right) \quad (3.35)$$

In most practical cases the above equation may be simplified to [29]

$$e_2 = \frac{K_\varphi K_r l_o^2 f_{yk}}{\pi^2 * 103,500 * d} \quad (3.36)$$

Where;

- K_φ : is a factor for taking account of creep; $K_\varphi = 1 + (0.35 + \frac{f_{ck}}{200} - \frac{\lambda}{150}) \phi_{ef} \geq 1$
- K_r : is a correction factor depending on axial load; $K_r = \frac{N_{ud} - N_{Ed}}{N_{ud} - N_{bal}} \leq 1$
- N_{ud} : is the ultimate axial load; $N_{ud} = f_{cd}A_c + f_{yd}A_{sc}$
- N_{bal} : is the axial load at balanced failure and for symmetrical reinforcement may be approximately taken as $N_{bal} = 0.29f_{ck}A_c$ [29]

In order to calculate K_r , the area A_s of the column reinforcement must be known and hence a trial-and-error approach is necessary, taking an initial conservative value of $K_r = 1$.

Finally, the ultimate axial load and the increased moment can be used to solve for A_s using design charts and checked against $A_{smin} = 0.002 * A_c$ and $A_{smax} = 0.08 * A_c$.

3.3.6.4 Interaction Curve Development

Interaction curves are developed for each member based on their respective section sizes and reinforcement amounts. Points for the interaction curve were calculated from the following section analysis.

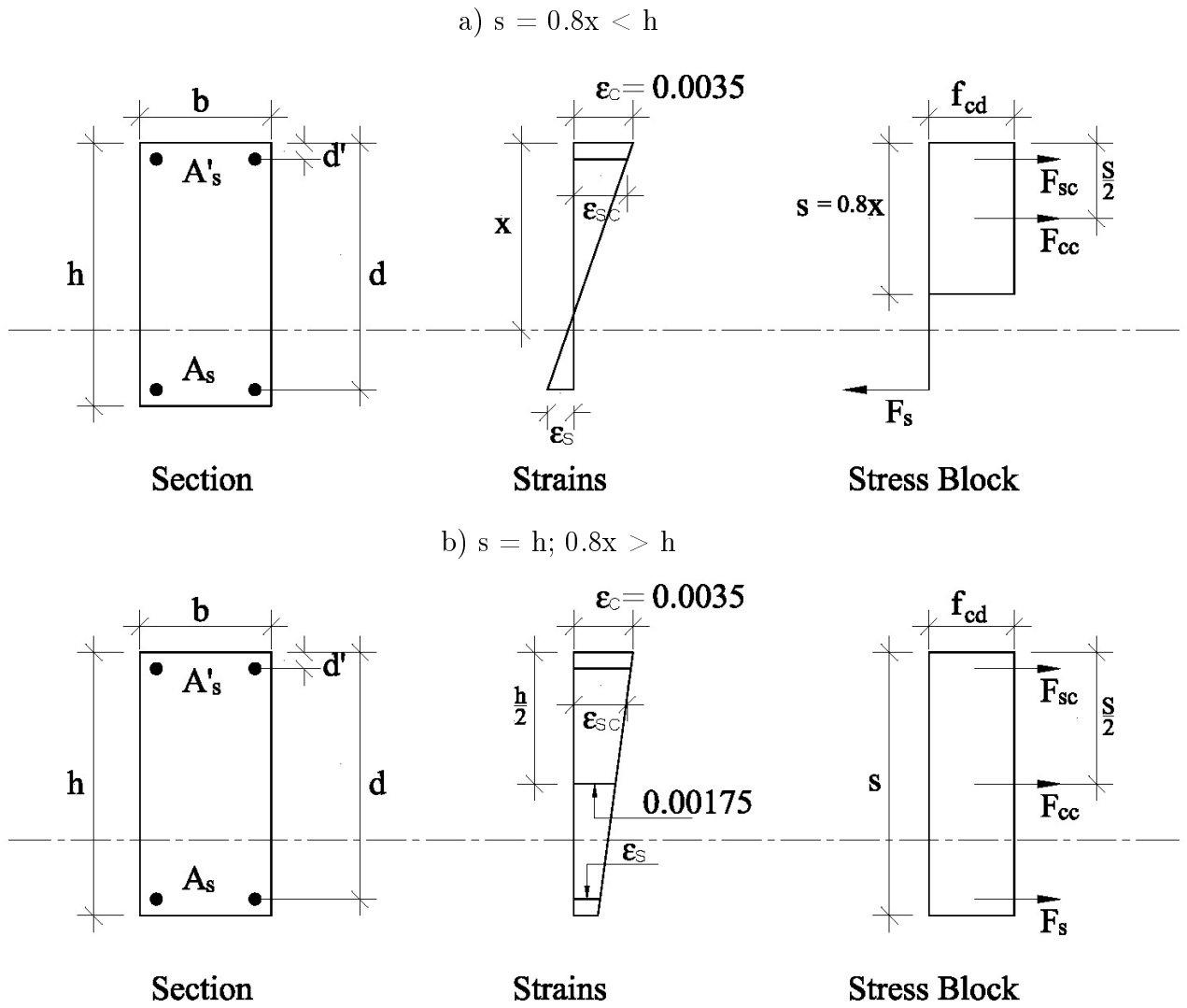


Figure 3.7: Section Analysis

1. Point A: Pure Compression

$$N_o = F_{cc} + F_{sc} + F_s = f_{cd}bh + f_{yd}A'_s + f_{yd}A_s$$

$$M = 0$$

2. Point B: Yielding of Compression Reinforcement

$$N = F_{cc} + F_{sc} - F_s = f_{cd}b * 0.8x + f_{yd}A'_s - f_{yd}A_s$$

$$M = F_{cc}(\frac{h}{2} - \frac{0.8x}{2}) + F_{sc}(\frac{h}{2} - d') + F_s(d - \frac{h}{2})$$

$$x = \frac{0.0035 * d'}{0.0035 - \varepsilon_{yd}}$$

3. Point C: Zero Strain in Tension Reinforcement

$$N = F_{cc} + F_{sc} = f_{cd}b * 0.8x + f_{sc}A'_s$$

$$M = F_{cc}(\frac{h}{2} - \frac{0.8x}{2}) + F_{sc}(\frac{h}{2} - d')$$

$$x = h - d'$$

$$\varepsilon_{sc} = \frac{0.0035 * (x - d')}{x}$$

If $\varepsilon_{sc} \geq \varepsilon_{yd}$, $f_{sc} = f_{yd}$. Otherwise $f_{sc} = \varepsilon_{sc} * E_s$.

4. Point D: Balanced Failure

$$N_{bal} = F_{cc} + F_{sc} - F_s = f_{cd}b * 0.8x_{bal} + f_{sc}A'_s - f_{yd}A_s$$

$$M_{bal} = F_{cc}(\frac{h}{2} - \frac{0.8x_{bal}}{2}) + F_{sc}(\frac{h}{2} - d') - F_s(d - \frac{h}{2})$$

$$x_{bal} = \frac{d}{1 + \frac{\varepsilon_{yd}}{0.0035}}$$

$$\varepsilon_{sc} = \frac{0.0035 * (x_{bal} - d')}{x_{bal}}$$

If $\varepsilon_{sc} \geq \varepsilon_{yd}$, $f_{sc} = f_{yd}$. Otherwise $f_{sc} = \varepsilon_{sc} * E_s$.

5. Point E: Pure Flexure

$$N = 0$$

$$M_o = F_{cc}(\frac{h}{2} - \frac{0.8x}{2}) + F_{sc}(\frac{h}{2} - d') + F_s(d - \frac{h}{2})$$

$$F_{cc} = f_{cd}b * 0.8x$$

$$F_{sc} = f_{sc}A'_s$$

$$F_s = f_{yd}A_s$$

$$\varepsilon_{sc} = \frac{0.0035 * (x - d')}{x}$$

$$f_{sc} = \varepsilon_{sc} * E_s$$

6. Point F: Pure Tension

$$N_o = -F_{sc} - F_s = -f_{yd}A'_s - f_{yd}A_s$$

$$M = 0$$

$$\varepsilon_c = 0$$

$$\varepsilon_s = \varepsilon_{sc} = 0.025$$

3.3.7 Design for Shear[29]

The actual behaviour of reinforced concrete in shear is complex, and difficult to analyze theoretically, but by applying the results from many experimental investigations, reasonable simplified procedures for analyses and design have been developed. The shear reinforcement will usually take the form of vertical links or a combination of links and bent-up bars. The procedure for designing shear links is as follows:

1. Calculate the ultimate design shear forces V_{Ed} along the beam's span.
2. Check the crushing strength $V_{Rd,max}$ of the concrete diagonal strut at the section of maximum shear, usually at the face of the beam support. For most cases the angle of inclination of the strut is $\theta = 22^\circ$, with $\cot\theta = 2.5$ and $\tan\theta = 0.4$.

$$V_{Rd,max} = 0.124b_wd(1 - \frac{f_{ck}}{250})f_{ck} \quad (3.37)$$

- If $V_{Rd,max} \geq V_{Ed}$, then go to step (3) with $\theta = 22^\circ$ and $\cot\theta = 2.5$
- If $V_{Rd,max} < V_{Ed}$, then $\theta > 22^\circ$ and θ must be calculated from equation 3.38. But if the resulting θ is greater than 45° , the concrete section has failed and should be increased.

$$\theta = 0.5 \sin^{-1} \left(\frac{V_{Ed}}{0.18b_w d f_{ck} \left(\frac{f_{ck}}{250} \right)} \right) \leq 45 \quad (3.38)$$

3. The shear links required can be calculated from

$$\frac{A_{sw}}{s} = \frac{V_{ED}}{0.78df_{yk}\cot\theta} \quad (3.39)$$

where A_{sw} is the cross-sectional area of the stirrups $2 * \frac{\pi\Phi^2}{4}$ for single stirrups. For a predominately uniformly distributed load the shear V_{Ed} should be taken at a distance d from the face of the support and the shear reinforcement should continue to the face of the support.

4. Calculate the minimum links required from

$$\frac{A_{sw,min}}{s} = \frac{0.08f_{ck}^{0.5}b_w}{f_{yk}} \quad (3.40)$$

and the shear resistance of the links actually specified as

$$V_{min} = \frac{A_{sw}}{s} * 0.78df_{yk}\cot\theta \quad (3.41)$$

3.3.8 Capacity Design[5]

The capacity design portion of the software utilizes results obtained from the above design results to calculate the flexural capacities of beams under sagging and hogging condition for the provided reinforcements. The flexural capacity of columns at a joint is compared with actual flexural capacity of joining beams. If the sum of capacities of columns is less than the sum of capacities of beams multiplied by over strength factor, the column moments will be magnified by the factor (moment magnification factor) by which they are lacking in moment capacity over beams. If the sum of the column moments is greater than the sum of beam moments, there is no need to magnify the column moments. Columns are designed for the revised moments and the axial force coming on them from the analysis.

$$\sum M_{Rc} \geq 1.3 \sum M_{Rb} \quad (3.42)$$

Where;

- $\sum M_{Rc}$: is the sum of the design values of the moments of resistance of the columns framing the joints.
- $\sum M_{Rb}$: is the sum of the design values of the moments of resistance of the beams framing the joint.

ES EN 1998:2015 aims to ensure life safety in a large earthquake together with damage limitation following a more frequent event. Whilst the code allows these events to be

resisted by either dissipative (ductile) or non-dissipative (essentially elastic) behaviour, there is a clear preference for resisting larger events through dissipative behaviour. Hence, much of the code is framed with the aim of ensuring stable, reliable dissipative performance in predefined “critical regions”, which limit the inertial loads experienced by other parts of the structure. The design and detailing rules are formulated to reflect the extent of the intended plasticity in these critical regions, with the benefits of reduced inertial loads being obtained through the penalty of more stringent layout, design and detailing requirements. This is particularly the case for reinforced concrete structures where such performance can only be achieved if strength degradation during hysteretic cycling is suppressed by appropriate detailing of these critical zones to ensure that stable plastic behaviour is not undermined by the occurrence of brittle failure modes such as shear or compression in the concrete or buckling of the reinforcing steel. With this in mind, three ductility (dissipation) classes are introduced[10]:

1. Ductility class low (DCL) in which virtually no hysteretic ductility is intended and the resistance to earthquake loading is achieved through the strength of the structure rather than its ductility.
2. Ductility class medium (DCM) in which quite high levels of plasticity are permitted and corresponding design and detailing requirements are imposed.
3. Ductility class high (DCH) where very large inelastic excursions are permitted accompanied by even more onerous and complex design detailing requirements.

For DCL structures, ES EN 1998:2015 imposes very limited material requirements in addition to the ES EN 1992:2015 provisions. Whereas for DCM and DCH structures, increasingly more onerous material requirements are imposed, together with geometrical constraints, capacity design provisions and detailing rules tied to local ductility demand. These rules are aimed at the suppression of brittle failure modes, provision of capacity to withstand non-linear load cycles without significant strength degradation and improving the ability of defined critical regions to undergo very high local rotational ductility demands in order to achieve the lower global demands. Typically this includes[10]:

- Ensuring flexural yielding prior to shear failure.
- Providing stronger columns than beams to promote a more efficient beam side-sway mode of response and avoid soft storey failure.
- Retention of an intact concrete core within confining links.
- Prevention of buckling of longitudinal reinforcement.
- Limiting flexural tension reinforcement to suppress concrete crushing in the compression zone.

These detailed requirements build upon the guidelines in Section 4 of ES EN 1998:2015 on:

- Regularity of structural arrangement, aiming to promote an even distribution of ductility demand throughout the structure.
- Providing adequate stiffness, both to limit damage in events smaller than the design earthquake and to reduce the potential for significant secondary P- Δ effects.

ES EN 1998:2015 Section 4.2.2 distinguishes between primary and secondary members. Primary members are those that contribute to the seismic resistance of the structure. Some structural elements can, however, be designated as secondary elements, which are taken as resisting gravity loads only. Their contribution to seismic resistance must be neglected. These members must be shown to be capable of maintaining their ability to support the gravity loads under the maximum deflections occurring during the design earthquake. This may be done by ensuring that the actions (moments, shears and axial forces) that develop in them under the calculated seismic deformations do not exceed their design strength, as calculated in ES EN 1992:2015. Otherwise no further seismic design or detailing requirements are demanded.

3.3.8.1 Primary Beams

Flexural capacities of singly reinforced beams can be calculated as follows:

$$M_{Rb} = A_{sProvided} * f_{yd} * z \quad (3.43)$$

And for doubly reinforced beams:

$$\mu_{sd,s} = \frac{A'_{sProvided} * f_{sc} * (d - d')}{f_{cd} b d^2} + \mu_{sd,s}^* \quad (3.44)$$

$$M_{Rb} = \mu_{sd,s} * b d^2 f_{cd} \quad (3.45)$$

Where f_{sc} and z are as defined in Section 3.3.5.1.

During an earthquake, to avoid under any circumstances the possibility of encountering a shear failure, the shear force associated with the development of two plastic hinges in each span with overstrength needs to be evaluated. To the earthquake induced shear force those due to the simultaneously acting gravity loads must be added. When determining the necessary shear reinforcement, conservative assumptions with respect to shear transfer mechanism in the plastic regions of the beam should be made in recognition of the expected deterioration of the shear resistance in these regions resulting from imposed cyclic reversing inelastic deformations.[21]

To satisfy the local ductility requirement the reinforcement ratio of the tension zone shall not exceed a value equal to:

$$\rho_{max} = \rho' + \frac{0.0018}{\mu_\phi \varepsilon_{sy,d}} * \frac{f_{cd}}{f_{yd}} \quad (3.46)$$

Where;

- $\rho' = \frac{A_{s(comp)}}{b*d}$
- $\mu_\phi = 2q_o - 1$ if $T_1 \geq T_c$ or $\mu_\phi = 1 + \frac{2*(q_o-1)*T_c}{T_1}$ if $T_1 < T_c$
- $\varepsilon_{sy,d} = \frac{f_{yd}}{E_s}$

with the reinforcement ratios of the tension zone and compression zone, ρ and ρ' , both normalized to $b * d$, where b is the width of the compression flange of the beam and d is the effective depth of the section.

Along the entire length of a primary seismic beam, the reinforcement ratio of the tension zone, ρ , shall be not less than the following minimum value ρ_{min} (f_{ctm} as given in ES EN 1992:2015 Table 3.1):

$$\rho_{min} = 0.5 * \left(\frac{f_{ctm}}{f_{yk}} \right) \quad (3.47)$$

And for DCH at least two high bond bars with $d_b = 14mm$ shall be provided both at the top and the bottom of the beam that run along the entire length of the beam.

In primary seismic beams the design shear forces shall be determined in accordance with the capacity design rule, on the basis of the equilibrium of the beam under:

- a) the transverse load acting on it in the seismic design situation and
- b) end moments $M_{i,d}$ (with $i=1,2$ denoting the end sections of the beam), corresponding to plastic hinge formation for positive and negative directions of seismic loading. The plastic hinges should be taken to form at the ends of the beams or (if they form there first) in the vertical elements connected to the joints into which the beam ends frame (Figure 3.8).

At end section i , two values of the acting shear force should be calculated, i.e. the maximum $V_{Ed,max,i}$ and the minimum $V_{Ed,min,i}$ corresponding to the maximum positive and the maximum negative end moments $M_{i,d}$ that can develop at ends 1 and 2 of the beam. End moments $M_{i,d}$ may be determined as follows:

$$M_{i,d} = \gamma_{Rd} M_{Rb,i} \min\left(1, \frac{\sum M_{Rc}}{\sum M_{Rb}}\right) \quad (3.48)$$

Shear forces at each ends of a beam weaker than the columns can then be calculated from

$$V_{Ed,i} = \frac{\sum M_{i,d}}{l_{cl}} \pm V_{o,g+\psi_2q} \quad (3.49)$$

Where;

- γ_{Rd} : is the factor accounting for possible overstrength due to steel strain hardening, which in the case of DCM beams may be taken as being equal to 1.0 and 1.20 for DCH;
- $M_{Rb,i}$: is the design value of the beam moment of resistance at end i in the sense of the seismic bending moment under the considered sense of the seismic action;
- $\sum M_{Rc}$ and $\sum M_{Rb}$: are the sum of the design values of the moments of resistance of the columns and the sum of the design values of the moments of resistance of the beams framing into the joint. The value of $\sum M_{Rc}$: should correspond to the column axial force(s) in the seismic design situation for the considered sense of the seismic action.
- $V_{o,g+\psi_2q}$: is the shear force calculated from permanent load and percentage of live load;
- l_{cl} : is the beam clear length.

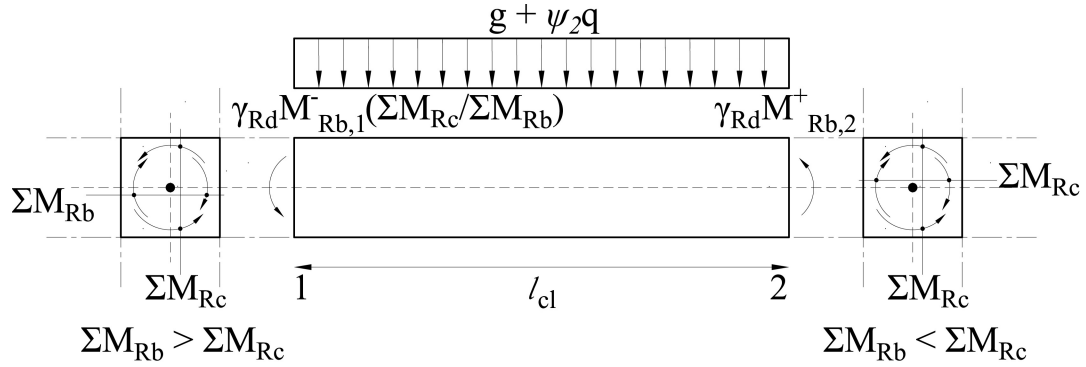


Figure 3.8: Capacity Design Values of Moments for Shear Force Calculation in Beams (a)[5]

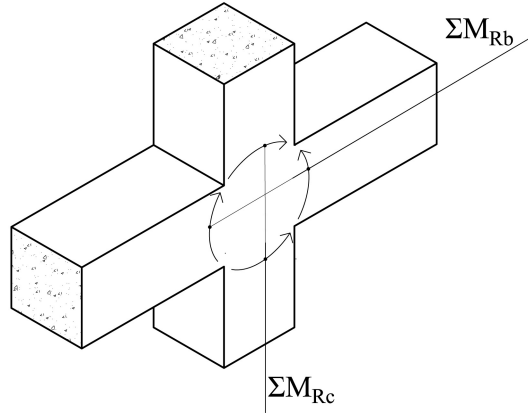


Figure 3.9: Capacity Design Values of Moments for Shear Force Calculation in Beams (b)[10]

Within the critical regions of primary seismic beams, hoops satisfying the following conditions shall be provided:

- The diameter d_{bw} of the hoops (in millimeters) shall be not less than 6 for both DCM and DCH.
- The spacing, s , of hoops (in millimeters) shall not exceed: $s = \min \frac{h_w}{4}; 24d_{bw}; 225; 8d_{bL}$ for DCM and $s = \min \frac{h_w}{4}; 24d_{bw}; 175; 6d_{bL}$ for DCH.

where d_{bL} is the minimum longitudinal bar diameter (in millimeters); and h_w the beam depth (in millimeters).

3.3.8.2 Primary Columns

In primary seismic columns the value of the normalized axial force, v_d , shall not exceed 0.65 for DCM and 0.55 for DCH. This is to ensure minimum flexural ductility in columns.

$$v_d = \frac{N_{Ed}}{A_c f_{cd}} \quad (3.50)$$

Flexural capacities of columns can be calculated from provided reinforcement amount by first computing for the mechanical ratio of reinforcement.

$$\omega = \frac{A_{s,Provided} * f_{yd}}{A_c * f_{cd}} \quad (3.51)$$

Then μ_{sd} that corresponds to this mechanical ratio of reinforcement and the column axial force, N_{Ed} , is read from the appropriate interaction chart and the column flexural capacity, M_{Rc} , can be calculated as follows:

$$M_{Rc} = \mu_{sd} * f_{cd} A_c h \quad (3.52)$$

This was done by first constructing an interaction curve (see Section 3.3.6.4) using the provided reinforcement amount and by reading, from this constructed interaction curve, the moment resistance that corresponds to the applied axial load.

To satisfy the local ductility requirement the total longitudinal reinforcement ratio ρ_1 , for both DCM and DCH, shall be not less than 0.01 and not more than 0.04. In symmetrical cross-sections symmetrical reinforcement should be provided ($\rho = \rho'$).

The design shear force for columns should be based on the flexural overstrengths that could be developed at both ends of the columns under consideration[21]. In primary seismic columns the design values of shear forces shall be determined in accordance with the capacity design rule, on the basis of the equilibrium of the column under end moments $M_{i,d}$ (with $i=1,2$ denoting the end sections of the column), corresponding to plastic hinge formation for positive and negative directions of seismic loading. The plastic hinges should be taken to form at the ends of the beams connected to the joints into which the column end frames, or (if they form there first) at the ends of the columns (Figure 3.10). End moments $M_{i,d}$ may be determined from the following expression:

$$M_{i,d} = \gamma_{Rd} M_{Rc,i} \min\left(1, \frac{\sum M_{Rb}}{\sum M_{Rc}}\right) \quad (3.53)$$

Shear forces at each ends of a column can then be calculated from

$$V_{Ed,i} = \frac{\sum M_{i,d}^{ends}}{l_{cl}} \quad (3.54)$$

Where;

- γ_{Rd} : is the factor accounting for overstrength due to steel strain hardening and confinement of concrete of the compression zone of the section, which in the case of DCM columns may be taken as being equal to 1.10 and 1.30 for DCH;
- $M_{Rc,i}$: is the design value of the column moment of resistance at end i in the sense of the seismic bending moment under the considered sense of the seismic action;
- $\sum M_{Rc}$ and $\sum M_{Rb}$: are as defined above for beam. The value of $\sum M_{Rc}$ should correspond to the column axial force(s) in the seismic design situation for the considered sense of the seismic action.
- l_{cl} : is the column clear length.

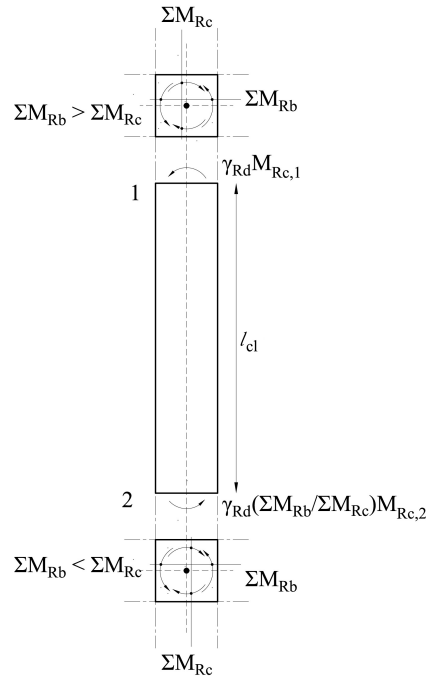


Figure 3.10: Capacity Design Values of Moments for Shear Force Calculation in Columns (a)[5]

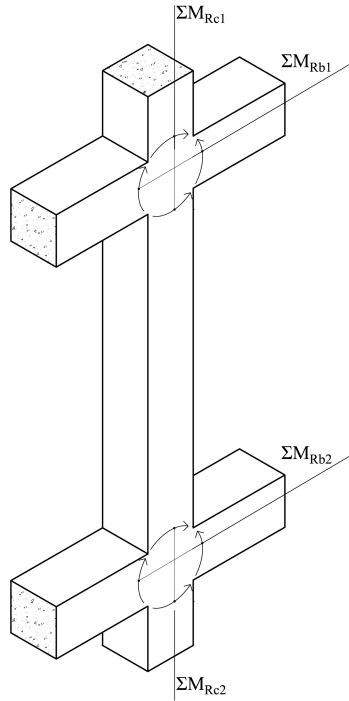


Figure 3.11: Capacity Design Values of Moments for Shear Force Calculation in Columns (b)[10]

Within the critical regions of primary seismic columns, hoops and cross-ties satisfying

the following conditions shall be provided:

- The diameter d_{bw} of the hoops and cross-ties (in millimeters) shall be not less than 6 for both DCM and DCH.
- The spacing, s , of hoops (in millimeters) shall not exceed: $s = \min \frac{b_o}{2}; 175; 8d_{bL}$ for DCM and $s = \min \frac{b_o}{3}; 125; 6d_{bL}$ for DCH.

where d_{bL} is the minimum longitudinal bar diameter (in millimeters); and b_o (in millimeters) is the minimum dimension of the concrete core (to the centreline of the hoops).

3.3.9 Output

All necessary results will be presented in tables. Apart from that, visual results will also be used to give better visualization of the outputs. And all information can also be exported in a tabular form by using .TSV file format, which can be readily opened by any spreadsheet application like Microsoft Excel and can be printed from there.

3.3.10 Testing and Verification

Output results from the developed software will be compared with results obtained from CSI ETABS v.18.0.2; an engineering software product that caters to multi-story building analysis and design.

Chapter 4

Implementation

4.1 General Program Structure

The software, Capacity Design Program for Frames (fCDP), follows a top-down object-oriented class structure where one primary routine can inherit all of its superclasses' attributes. This approach eliminates the need for extremely long and hard-to-debug scripts familiar in programming languages such as FORTRAN. Thus, as dictated by Java's object-oriented paradigm, the program flows in such a way that one entity or data structure contains the other. Highest in the organization are packages, which contain class definitions, data files, images and other files necessary for the isolated operation of the module/package. The software implemented comprises of three main packages, namely:

- Earthquake Analysis Package: responsible for computing element forces and moments acting on the structure as the result of applied earthquake load.
- Member Design Package: designs structural member attributes for adequacy in bending moment and shear force resistance.
- Capacity Design Package: checks design attributes of individual structural members for adequacy in capacity requirement and redesigns them if found inadequate.

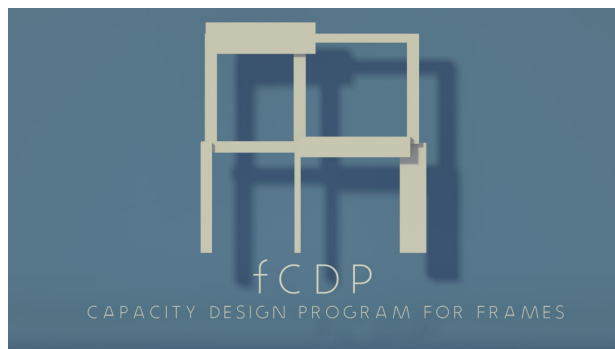


Figure 4.1: Capacity Design Program for Frames

4.2 Earthquake Load and Analysis

4.2.1 Earthquake Load Computation

Procedures outlined in ES EN 1998:2015 were followed for earthquake load computation. Thus, all seismic forces are obtained from lateral force method of analysis. Based on the input data the program calculates base shear force and distributes storey shear to each level of the frame as presented in section 3.3.4. All forces should have a unit of kilo Newtons (kN), while all moments are to have units of kilo-Newton-Meters (kN-m). Major required data are storey weight, storey height, ground type, soil condition, seismic zone, response spectra type and ductility class. The first two will automatically be computed from the already available structure data whereas the rest are to be fed in by the user.

In determining storey weight the sum of all imposed loads and the weight of every member in that specific storey are computed. And storey heights are computed by calculating the vertical length difference between two consecutive levels. Algorithms listed in section 4.7.2 of this chapter summarize these procedures as carried out by the program.

4.2.2 Earthquake Analysis

Earthquake induced loads obtained from the above computations need be combined with gravity loads for analysis purpose. Here load combinations necessary for earthquake analysis are first generated as per the code. These combinations are employed in computing the earthquake design actions. ES EN 1990:2015 Section 6.4.3.4[31] gives how seismic actions should be combined with other actions as shown below.

$$E_d = \sum G_{k,j} + P + A_d + \sum \psi_{2,i} Q_{ki} \quad (4.1)$$

Where;

- E_d : is the design value of actions;
- $G_{k,j}$: is the characteristic value of permanent action j;
- P : is the relevant representative value of a prestressing action (not relevant here);
- A_d : is design value of an accidental/seismic action;
- $\psi_{2,i}$: is partial factor given in ES EN 1990:2015 Annex A, Table A1.1 which reduces the variable action from its characteristic (upper bound) value to its value, expected to be present for most of the time;
- Q_{ki} : is the characteristic value of variable action j.

4.3 Member Design Implementation

4.3.1 Assumptions

The “members” referred to here are rectangular reinforced concrete frame elements, specifically beams and columns, and all algorithms for the program were developed with this assumption. Besides, all members are to have either vertical or horizontal alignment that is inclined members are not part of this research. Section size requirements for beams and columns imposed by ES EN 1991:2015 and 1992:2015 have also been dealt with. Member design algorithms can be found in sections 4.7.3 and 4.7.4.

4.3.2 Beam Design

Beams shall be designed to resist bending moments and shear forces obtained from structural analysis. Depending on the magnitude of their design moments, beams could be singly or doubly reinforced. The program can do the design of beams that require tension reinforcement alone as well as both tension and compression reinforcements as per sections 3.3.5.1 and 3.3.5.2. It also carries out shear design as presented in section 3.3.7.

4.3.3 Column Design

Column slenderness ratios for all columns are computed before designing them. Columns with slenderness ratios less than the slenderness ratio limit are termed short columns but otherwise known as slender (See section 3.3.6.1). Doing this will allow the program to know which design procedure to follow in designing a specific column as outlined in sections 3.3.6.2 and 3.3.6.3. Short columns shall be designed for axial loads and uniaxial moments coming on them. As for the slender ones, their design shall in addition consider moments arising from second-order effects. Then shear design is done as per section 3.3.7.

4.4 Capacity Design Implementation

Capacity design begins by comparing the sum of moment capacities of beams with that of columns at a joint. The first step then is to scan through the list of joints generated for constructing the frame and collect those where beams and columns frame into. In other words, joints at frame bases with only columns joining them will be left out. A simple way to achieve this was to sort out which joints are restrained and which are not. Since only those joints at frame bases are restrained, all the unrestrained joints could be used for the capacity check mentioned earlier.

Members adjoining at these unrestrained joints should be identified as column or beams to use their respective moment capacities in equation 3.42. Sum of beam moment

capacities, $\sum M_{Rb}$, at each unrestrained joint are calculated, multiplied with overstrength factor ($OF = 1.3$ as given in ES EN 1998:2015[5]) and compared with sum of column moment capacities, $\sum M_{Rc}$, at the same joint. If the sum of column moments is greater, their capacity will not be magnified. But if they are less, their capacity will be increased by multiplying them with the moment magnification factor (MMF) calculated using equation 4.2. All columns with magnified moment capacities are redesigned to accommodate the increases. Finally, shear capacities for all members are calculated as outlined in 3.3.8.1 and 3.3.8.2. The algorithm presented in section 4.7.4 depicts all step by step capacity design processes followed by the program.

$$MMF = \frac{1.3 \sum M_{Rb}}{\sum M_{Rc}} \quad (4.2)$$

4.5 Modification to Helix's Interface

In order to make use of Helix's interface for Capacity Design Program for Frames (fCDP) some changes and introduction of some new functions were necessary.

4.5.1 Modified Ribbon and New Functions

Ribbon	Functions	Remark
 Draw Member	Sets the display canvas to drawing mode.	No Change
 Select Joints	Sets the display canvas to joint selection mode.	No Change
 Select Members	Sets the display canvas to member selection mode.	No Change
 Restraints	Displays the joint restraint options menu.	Modified
 Material Def.	Displays the material definition menu.	No Change
 Member Def.	Displays the member definition menu. New input fields shown in the next section.	Modified
 EQ Analysis Input	Displays the earthquake analysis input data menu. The required input data are presented in the next section.	New
 Sup. Imp. Loads	Displays the superimposed load definition menu where direction of application, type (distributed or concentrated), magnitude and location can be specified.	Modified
 Live Loads	Displays the live load definition menu where direction of application, type (distributed or concentrated), magnitude and location can be specified.	Modified
 Capacity Design	Performs structural analysis, structural design and capacity design if all relevant data has been entered, and topology has been defined appropriately.	New

Table 4.1: Modified Main Ribbon Functions

4.5.2 Modified and New Sub-menus

Two modified and one new sub-menus could be accessed from the modified main ribbon. These are:

1. **Restraints:** Similar to the previously available “Restraints” sub-menu but with a new option of “Fixed” check box. While checked the “Fixed” option assigns restraints to all degrees of freedom facilitating the restraint input process since fixed base is usually the case in concrete building design.
2. **Member Definition:** This sub-menu was previously labeled section definition but since additional data input are required for capacity design it had to be modified to accommodate new input fields. Additionally, “Defaults” option is provided using which predetermined data could be set.
3. **Earthquake Analysis Input:** This is a new sub-menu which enables users to feed in earthquake specific-analysis data. These include Importance Class of building, Seismic Zone of city or town, Ground Type of site, Spectra Type of choice, Ductility Class of structure and Building Category. It also provides a “Defaults” option with preset input data.

The figure displays three separate dialog boxes side-by-side. The first, titled 'Restraints', contains a 'Fixed' checkbox, a 'Displacements' section with 'x' and 'y' checkboxes, and a 'Rotations' section with a 'z' checkbox. The second, titled 'Member Definition...', features a 'Defaults' checkbox and five input fields: 'Breadth (B), mm', 'Height (H), mm', 'Concrete Cover, mm', 'Moment Redistribution Ratio (δ), %', and 'Member Ductility Class; L, M or H'. The third, titled 'EQ Input Data...', also has a 'Defaults' checkbox and six input fields: 'Importance Class; I, II, III or IV', 'Seismic Zone; 0, 1, 2, 3, 4 or 5', 'Ground Type; A, B, C, D or E', 'Spectra Type; 1 or 2', 'Ductility Class; L, M or H', and 'Building Category; A, B, C, D, E, F or G'. Each dialog box includes 'Accept' and 'Cancel' buttons at the bottom.

Figure 4.2: Modified Sub-Menus

4.5.3 The Results Output Dialogs and Tables

Output dialog boxes are displayed, if and only if, a successful design or structural analysis has been performed. Through these dialogs output data such as internal action magnitudes, design reinforcement areas and sizing information can be viewed. Additionally, result output tables can be accessed from these dialogs. And all information can also be exported in a tabular form by using .TSV file format, which can be readily opened by any spreadsheet application like Microsoft Excel.

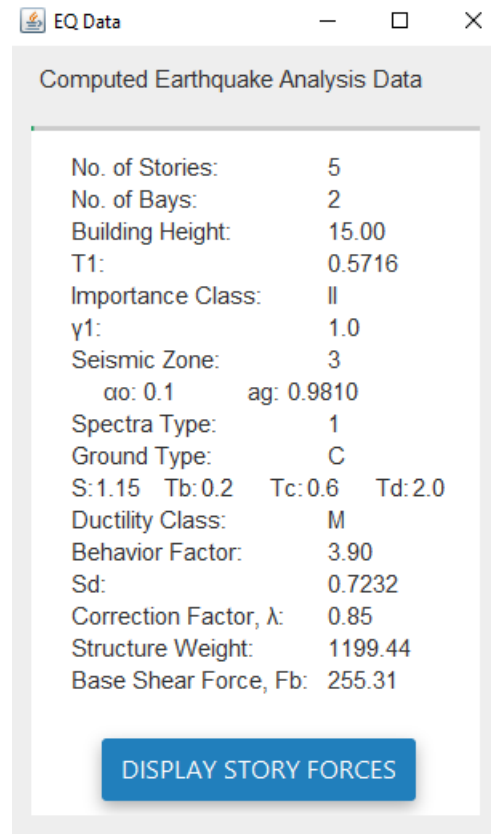


Figure 4.3: Earthquake Data Display Dialog

Story Forces Output Table					
Story	Elevation(m)	Level, Zi(m)	Story Mass, Mi(kN)	ZiMi(kN-m)	Story Force(kN)
5	12.00	15.00	81.80	1227.06	74.37
4	9.00	12.00	92.09	1105.05	66.98
3	6.00	9.00	100.26	902.35	54.69
2	3.00	6.00	106.31	637.88	38.66
1	0.00	3.00	113.34	340.02	20.61
Foundation	-3.00				
		Sum	493.81	4212.37	255.31
EXPORT TO TSV					

Figure 4.4: Story Forces Output Table

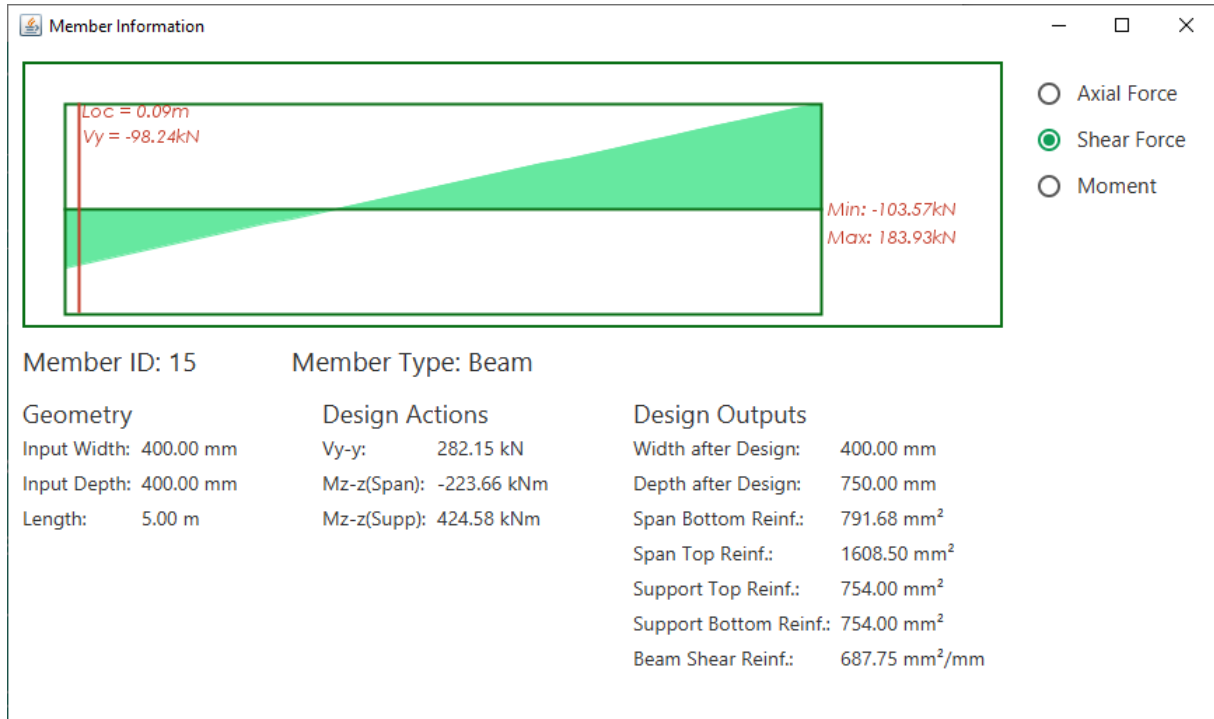


Figure 4.5: Member Result Information Dialog

Design Output Table												
Member ID	Member Type	Input Width (mm)	Final Width (mm)	Input Depth (mm)	Final Depth (mm)	Member Length (m)	Ast1	Asc1	Ast2	Asc2	As	Shear Reinf...
0	Column	400	650	400	650	3.00	-	-	-	-	8Φ16	Φ8c/c110
1	Column	400	600	400	600	3.00	-	-	-	-	8Φ16	Φ8c/c110
2	Column	400	600	400	600	3.00	-	-	-	-	4Φ20	Φ8c/c160
3	Column	400	650	400	650	3.00	-	-	-	-	4Φ14	Φ6c/c110
4	Column	400	400	400	400	3.00	-	-	-	-	4Φ20	Φ6c/c110
5	Column	400	500	400	500	3.00	-	-	-	-	8Φ16	Φ6c/c110
6	Column	400	450	400	450	3.00	-	-	-	-	8Φ14	Φ10c/c130
7	Column	400	450	400	450	3.00	-	-	-	-	8Φ14	Φ10c/c150
8	Column	400	650	400	650	3.00	-	-	-	-	8Φ16	Φ8c/c110
9	Column	400	550	400	550	3.00	-	-	-	-	8Φ20	Φ8c/c120
10	Column	400	700	400	700	3.00	-	-	-	-	8Φ20	Φ6c/c110
11	Column	400	450	400	450	3.00	-	-	-	-	4Φ20	Φ10c/c100
12	Column	400	600	400	600	3.00	-	-	-	-	4Φ14	Φ8c/c100
13	Column	400	750	400	750	3.00	-	-	-	-	8Φ14	Φ10c/c150
14	Column	400	450	400	450	3.00	-	-	-	-	8Φ16	Φ8c/c150
15	Beam	400	400	400	500	5.00	3Φ16	6Φ12	8Φ16	6Φ12	-	Φ8c/c140
16	Beam	400	400	400	500	5.00	3Φ20	8Φ12	7Φ20	8Φ12	-	Φ8c/c110
17	Beam	400	400	400	450	5.00	4Φ16	7Φ12	8Φ16	7Φ12	-	Φ8c/c140
18	Beam	400	400	400	550	5.00	3Φ20	8Φ12	7Φ20	8Φ12	-	Φ8c/c110
19	Beam	400	400	400	550	5.00	5Φ14	6Φ12	8Φ14	6Φ12	-	Φ8c/c160
20	Beam	400	400	400	450	5.00	3Φ20	8Φ12	7Φ20	8Φ12	-	Φ8c/c110
21	Beam	400	400	400	550	5.00	6Φ12	4Φ12	8Φ12	4Φ12	-	Φ6c/c110
22	Beam	400	400	400	500	5.00	2Φ20	7Φ12	6Φ20	7Φ12	-	Φ8c/c130
23	Beam	400	400	400	400	5.00	7Φ12	3Φ12	7Φ12	3Φ12	-	Φ8c/c170
24	Beam	400	400	400	500	5.00	3Φ16	5Φ12	7Φ16	5Φ12	-	Φ8c/c160

Ast1: Span Bottom Reinforcement | Asc1: Span Top Reinforcement
Ast2: Support Bottom Reinforcement | Asc2: Support Top Reinforcement | As: Column Reinforcement

Figure 4.6: Design Output Table

4.6 Main Program Flowchart

The following flow chart summarizes the chain of commands the software, Capacity Design Program for Frames (fCDP), carries out.

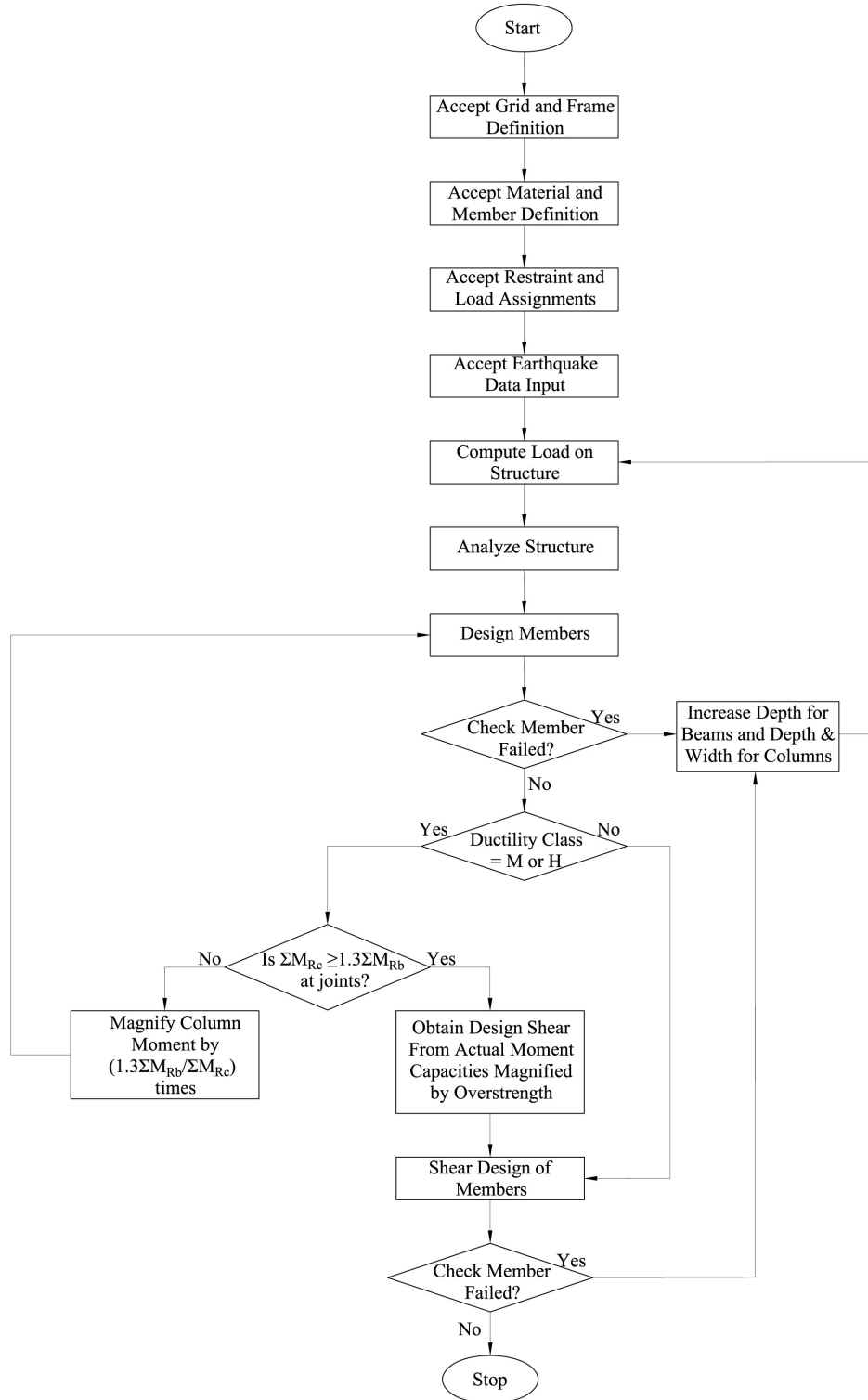


Figure 4.7: Main Program Flowchart

4.7 Algorithm Listings

This section presents a brief summary of algorithms developed for the topics discussed in previous sections of this chapter. While the algorithms were written in the hope of having a clear and concise description, some details haven't been included for the sake of brevity. The reader is advised to read the preceding sections before going through this section.

4.7.1 Main Program

Algorithm 4.1 Main Program

```
function Capacity Design(Structure S)

    Accept input
    Compute earthquake load
    Analyze structure
    Design members for flexure
    Check members failed in flexure
    If yes
        Increase member section, update
        input, reanalyze and redesign
    else
        Do capacity check
        Check members failed in capacity
        If yes
            Magnify column moments and
            redesign failed members for flexure
            Check members failed in flexure
            If yes
                Increase depth for beams and width
                & depth for columns, update
                input, reanalyze and redesign
            else
                Design members for shear
                Check members failed in shear
                If yes
                    Increase depth for beams and width
                    & depth for columns, update
                    input, reanalyze and redesign
                else
                    Output results
            end if
        else
            Design members for shear
            Check members failed in shear
            If yes
                Increase depth for beams and width
                & depth for columns, update
                input, reanalyze and redesign
            else
                Output results
            end if
        end if
    end if
end function
```

4.7.2 Earthquake Load Computation

Algorithm 4.2 Calculate Fundamental Period of Structure, T_1

```
function DetermineH(Structure S)

    for each joint J in  $J_i$  do
        take vertical coordinate  $y_i$ 
        remove redundant ones
    end for
    count number of coordinates
    determine number of stories (no. of coordinates - 1)
    sort these coordinates in ascending order
    calculate  $H$  by deducting the smallest coordinate from the largest

end function

function Calculate $T_1$ ()

     $T_1 = c_t * H^{3/4}$ 
     $c_t = 0.075$  for concrete frames

end function
```

Algorithm 4.3 Calculate Design Ground Acceleration, α_g

```
function DetermineValueOf $\alpha_o$ ()

    If Seismic Zone = 0
        return  $\alpha_o = 0$ 
    else if Seismic Zone = 1
        return  $\alpha_o = 0.04$ 
    else if Seismic Zone = 2
        return  $\alpha_o = 0.07$ 
    else if Seismic Zone = 3
        return  $\alpha_o = 0.10$ 
    else if Seismic Zone = 4
        return  $\alpha_o = 0.15$ 
    else if Seismic Zone = 5
        return  $\alpha_o = 0.20$ 

end function

function DetermineValueOf $\gamma_1$ ()

    If Importance Class = I
        return  $\gamma_1 = 0.8$ 
    else if Importance Class = II
        return  $\gamma_1 = 1.0$ 
    else if Importance Class = III
        return  $\gamma_1 = 1.2$ 
    else if Importance Class = IV
        return  $\gamma_1 = 1.4$ 

end function

function Calculate $\alpha_{gr}$ ()

     $\alpha_{gr} = \alpha_o * g$ 
     $g = 9.81m/s^2$ 

end function

function Calculate $\alpha_g$ ()

     $\alpha_g = \gamma_1 * \alpha_{gr}$ 

end function
```

Algorithm 4.5 Compute Base Shear Force, F_b

```
function Calculate $q$ ()

    If Ductility Class = L;  $q_o = 1.5$ 
    else if Ductility Class = M;  $q_o = 3 * MF$ 

        Multiplication factor,  $MF =$ 
        1.3, for multi-storey multi-bay frames which are regular in plan

    else if Ductility Class = H;  $q_o = 4.5 * MF$ 
    end if
 $q = \max \begin{cases} q_o * k_w \\ 1.5 \end{cases}$ 
 $k_w = 1$  for frames

end function
function Compute $\psi_{E,i}()$ 

    If Building Category = A, B or G
        return  $\psi_{2i} = 0.3$ 
    else if Building Category = C, D or F
        return  $\psi_{2i} = 0.6$ 
    else if Building Category = E
        return  $\psi_{2i} = 0.8$ 
    else
        return default:  $\psi_{2i} = 0.3$ 
    end if
 $\psi_{E,i} = \varphi * \psi_{2,i}$ 

end function
function Calculate $m()$ 


$$m = \sum G_{k,j} + \sum \psi_{E,i} Q_{k,i}$$


end function
function DetermineValueOf $\lambda()$ 

    If  $T_1 \leq 2T_C$  and if Number of Stories  $\geq 2$ 
         $\lambda = 0.85$ 
    else
         $\lambda = 1$ 
    end if

end function
function Calculate $F_b()$ 


$$F_b = S_d * m * \lambda$$


end function
```

Algorithm 4.4 Calculate S_d

function Calculate S_d ()

If $0 \leq T_1 \leq T_B$

$$S_d(T_1) = a_g * S * \left[\frac{2}{3} + \frac{T_1}{T_B} * \left(\frac{2.5}{q} - \frac{2}{3} \right) \right]$$

else if $T_B \leq T_1 \leq T_C$

$$S_d(T_1) = a_g * S * \eta * \frac{2.5}{q}$$

else if $T_C \leq T_1 \leq T_D$

$$S_d(T_1) = \max \left\{ \begin{array}{l} a_g * S * \frac{2.5}{q} * \left[\frac{T_C}{T_1^2} \right] \\ \geq \beta * a_g \end{array} \right.$$

else if $T_D \leq T_1$

$$S_d(T_1) = \max \left\{ \begin{array}{l} a_g * S * \frac{2.5}{q} * \left[\frac{T_C T_D}{T_1^2} \right] \\ \geq \beta * a_g \end{array} \right.$$

end function

function Determine $S, T_b, T_c \& T_d$ ()

If Spectra Type = 1

Ground Type	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.00	0.15	0.40	2.00
B	1.20	0.15	0.50	2.00
C	1.15	0.20	0.60	2.00
D	1.35	0.20	0.80	2.00
E	1.40	0.15	0.50	2.00

else if Spectra Type = 2

Ground Type	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.00	0.05	0.25	1.20
B	1.35	0.05	0.25	1.20
C	1.50	0.10	0.25	1.20
D	1.80	0.10	0.30	1.20
E	1.60	0.05	0.25	1.20

end if

return $S, T_b, T_c \& T_d$

end function

Algorithm 4.6 Storey Distribution of Horizontal Seismic Force

```
function Storey Force  $F_i$ (Structure S)

    for each storey level  $y_i$ 

        calculate weight of beams  $m_b$ 
        calculate weight of columns  $m_c$ 
        calculate imposed weight  $Imp_w$ 

    end for
    Storey weight,  $m_i = m_b + m_c + Imp_w$ 
    determine storey height
     $F_i = F_b * \frac{z_i m_i}{\sum z_j m_j}$ 

end function
```

4.7.3 Member Design

Algorithm 4.7 Beam Design for Flexure

```

function DesignBeamForFlexure(Member m,  $M_{Ed}$ )

    for each beam in  $m_i$  do

        calculate  $d = h - d'$  and accept input  $\delta$ 
        compute  $\mu_{sd,s} = \frac{M_{Ed}}{bd^2 f_{ck}}$ 
        compute  $\mu_{sd,s}^* = 0.454(\delta - 0.4) - 0.182(\delta - 0.4)^2$ 
        If  $K \leq K_{bal}$  (Singly Reinforced)

            compute  $z = d[0.5 + \sqrt{(0.25 - \frac{K}{1.134})}]$ 
            calculate  $A_s = \frac{M_{Ed}}{0.87 f_{yk} z}$ 
        else  $K > K_{bal}$  (Doubly Reinforced)
            calculate neutral axis depth  $x = (\delta - 0.4) * d$ 
            If  $\frac{d'}{x} \leq 0.38$ 
                 $f_{sc} = 0.87 * f_{yk}$ 
            else
                 $\varepsilon_{sc} = \frac{0.0035 * (x - d)}{x}$ 
                 $f_{sc} = /E_s * \varepsilon_{sc} / \leq f_{yd}$ 
            end if
            calculate  $z_{bal} = d - (\frac{0.8x_{bal}}{2})$ 
            Compute  $A'_s = \frac{(K - K_{bal}) f_{ck} b d^2}{f_{sc} (d - d')}$ 
            Compute  $A_s = \frac{K_{bal} f_{ck} b d^2}{0.87 f_{yk} z_{bal}} + A'_s \frac{f_{sc}}{0.87 f_{yk}}$ 
            If  $(A'_{sprov} - A'_{sreq}) \geq (A_{sprov} - A_{sreq})$ 
                return  $A'_s$  and  $A_s$ 
            else
                add to failed members list
            end if
        end if
        compute  $A_{smin} = 0.0013 * A_c$  for DCL
             $A_{smin} = 0.5 * \frac{f_{ctm}}{f_{yk}} * A_c$  for DCM & DCH
        compute  $A_{smax} = 0.04 * A_c$  for DCL
             $A_{smax} = \left[ \frac{A_{spro}}{b * d} + \frac{0.0018 * f_{cd}}{\mu_\phi * \varepsilon_{sy,d} * f_{yd}} \right] * A_c$ 
            for DCM & DCH
        If  $A_s \leq A_{smax}$  and  $A_s \geq A_{smin}$ 
            return  $A_s$ 
        else if  $A_s < A_{smin}$ 
            return  $A_{smin}$ 
        else
            add to failed members list
        end if
    end for
end function

```

Algorithm 4.8 Determine Slenderness and Slenderness Limit of Column

function DetermineSlenderness(Member m)

for each column in m_i do

determine stiffness = $\frac{I_{col}}{l_{col}}$

determine beam stiffness =

$\frac{I_b}{l_b}$ at both ends of column (BeamStiffness1 and BeamStiffness2)

determine k at end 1 $k_1 = \frac{ColumnStiffness}{\sum BeamStiffness1} \geq 0.1$

determine k at end 2 $k_2 = \frac{ColumnStiffness}{\sum BeamStiffness2} \geq 0.1$

determine effective height l_o

$$l_o = \max \left\{ \begin{array}{l} l \sqrt{(1 + 10 \frac{k_1 * k_2}{k_1 + k_2})} \\ l(1 + \frac{k_1}{1+k_1}) * (1 + \frac{k_2}{1+k_2}) \end{array} \right.$$

determine radius of gyration i

$i = \sqrt{\frac{I}{A}}$

determine slenderness $\lambda = \frac{l_o}{i}$

determine $n = \frac{N_{Ed}}{A_e f_{cd}}$

determine slenderness limit $\lambda_{lim} = \frac{20ABC}{\sqrt{n}}$

end for

end function

Algorithm 4.9 Column Design

```
function DesignColumn(Member m)

  for each column in  $m_i$  do

    determine slenderness  $\lambda$ 
    determine slenderness limit  $\lambda_{lim}$ 
    If  $\lambda < \lambda_{lim}$ 
      short column  $M_{Ed} = \text{Max}(M_{02}, N_{Ed}e_0)$ 
    else
      slender column  $M_{Ed} =$ 
         $\text{Max}(M_{02}; M_{0e} + M_2; 0.5M_2; N_{Ed}e_0)$ 
    end if
    Assume Reinforcement amount,  $A_s$ 
    Draw interaction curve (See Section 3.3.6.4)
    Form triangles using interaction points
    if the  $(N_{Ed}, M_{Ed})$  falls inside curve/triangles
      Accept assumed  $A_s$ 
      break
    else
      Assume another  $A_s$  and draw curve again
    end if
    calculate  $A_{smin} = 0.002 * A_c$  for DCL
       $A_{smin} = 0.01 * A_c$  for DCM and DCH
    calculate  $A_{smax} = 0.04 * A_c$  for DCL, DCM and DCH
    If  $A_{smin} \leq A_s < A_{smax}$ 
      return  $A_s$ 
    else if  $A_s < A_{smin}$ 
      return  $A_{smin}$ 
    else
      add to failed members list
    end if
  end for

end function
```

Algorithm 4.10 Design for Shear

```
function DesignForShear(Member m,  $V_{Ed}$ )

  for each beam in  $m_i$  do
    calculate  $V_{Rd,max} = 0.124b_w d(1 - \frac{f_{ck}}{250})f_{ck}$ 
    If  $V_{Rd,max} \geq V_{Ed}$ 
       $\theta = 22^\circ$  and  $\cot\theta = 2.5$ 
    else
       $\theta = 0.5\sin^{-1}(\frac{V_{Ed}}{0.18b_w d f_{ck}(\frac{f_{ck}}{250})}) \leq 45$ 
    end if
    calculate shear links required  $\frac{A_{sw}}{s} = \frac{V_{ED}}{0.78d f_{yk} \cot\theta}$ 
    If  $\frac{A_{sw}}{s} \geq \frac{0.08f_{ck}^{0.5}b_w}{f_{yk}}$ 
      use calculate amount
    else
      use minimum requirement
    end if
  end for

end function
```

4.7.4 Capacity Design

Algorithm 4.11 Capacity Design

```

function CapacityDesignCheck(Structure S)

    for each joint J in  $J_i$  do

        check if unrestrained
        add the unrestrained to  $\{UnrestJoLsit\}$ 

    end for
    for each joint j in  $\{UnrestJoLsit\}$  do

        find members with common joint j
        sort members as beams or columns
        Compute beam flexural capacity
        If singly reinforced
             $M_{Rb} = A_{sProvided} * f_{yd} * z$ 
        else i.e doubly reinforced
             $\mu_{sd,s} == \frac{A'_{sProvided} * f_{sc} * (d-d')}{f_{cd} b d^2} + \mu_{sd,s}^*$ 
             $M_{Rb} = \mu_{sd,s} * b d^2 f_{cd}$ 
        end if
        add capacities of beams  $\sum M_{Rb}$ 
        Compute column flexural capacity:

        Using the provided reinforcement amount calculate
        construct interaction curve
        Read moment capacity,  $M_{Rc}$ , that corresponds to  $N_{Ed}$ 
        from curve
        add capacities of columns  $\sum M_{Rc}$ 
        If  $\sum M_{Rc} \geq 1.3 \sum M_{Rb}$ 
            Use column moments as are
        else
            Magnify column moments by  $MMF$ 
             $MMF = \frac{1.3 \sum M_{Rb}}{\sum M_{Rc}}$ 
        end if
        redesign columns if moment magnified
        design beams and columns for shear forces from actual
        moment capacities (See sections 3.3.8.1 & 3.3.8.2)
        design beams for shear from moment capacities plus gravity
        load(permanent load and percentage of variable load)
         $V_{Ed,i} = \frac{\gamma_{Rd} \sum M_{i,d}}{l_{cl}} \pm V_{o,g+\psi_2q}$ 
        design columns for shear from moment capacities
         $V_{Ed,i} = \frac{\gamma_{Rd} \sum M_{i,d}^{ends}}{l_{cl}}$ 

    end for

end function

```

Chapter 5

Results and Discussion

5.1 General

This section presents how results obtained from, Capacity Design Program for Frames (fCDP), were crosschecked and verified. CSI ETABS v.18.0.2, referred to as ETABS from henceforth, was selected for this cause. In the process, three planar frames with different number of stories, number of bays and seismic properties were designed using fCDP and ETABS. Later, outputs from both software were checked against each other. The following sub-sections present the results.

Frame Name	A	B	C
No. of Stories	3	5	10
Story Height (m)	3.00	3.00	3.00
No. of Bays	4	5	4
Bay Width (m)	4.00	4.00	4.00
Ductility Class	M	M	H
Importance Class	II	II	II
Seismic Zone	3	3	3
Ground Type	C	C	C
Spectra Type	1	1	1
Building Category	B	B	B

Table 5.1: Input Data for Each Frame

Input Parameter	Value
Concrete Grade, f_{cu} (MPa)	30
Concrete Modulus of Elasticity, E_c (GPa)	31
Concrete Shear Modulus, G (GPa)	11.50
Steel Grade, f_{yk} (MPa)	500
Steel Modulus of Elasticity, E_s (GPa)	200
Initial Beam Sections (mm)	300 x 200
Initial Column Sections (mm)	300 x 300
Live Load on Beams (kN/m)	15
Superimposed Load on Beams (kN/m)	35

Table 5.2: Material, Section and Loading Input Data (Same for all frames)

5.2 Result Comparison

5.2.1 Frame A (3 Storey x 4 Bay)

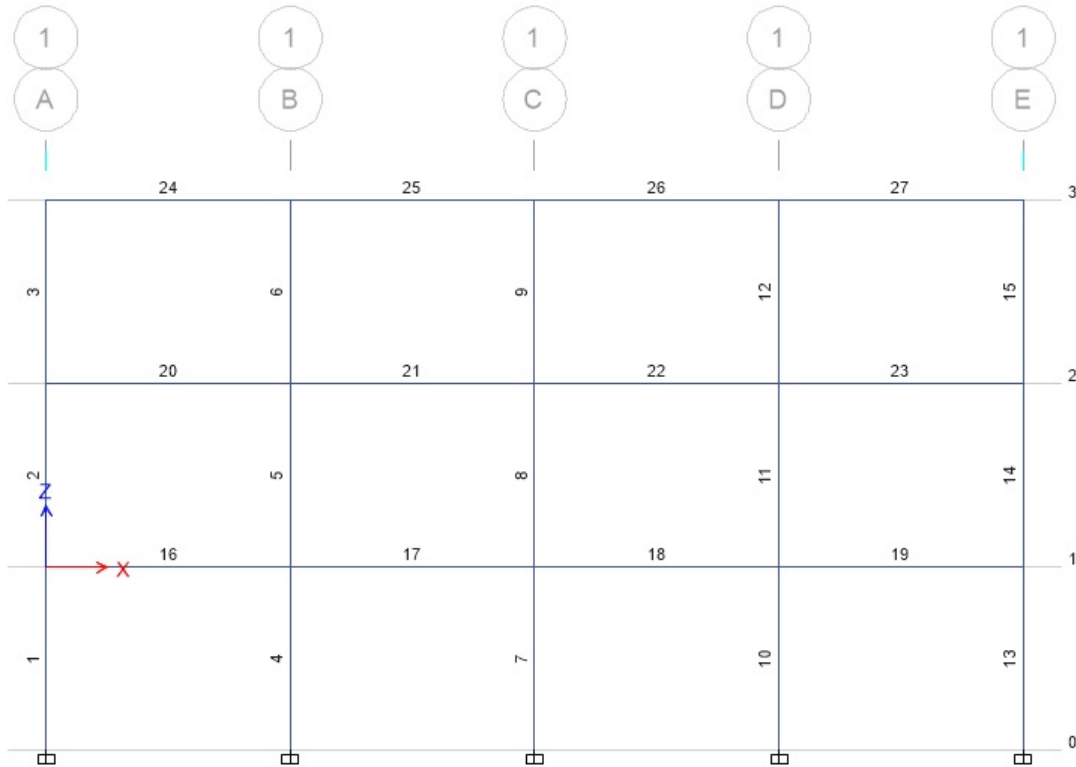


Figure 5.1: Frame A

5.2.1.1 Section Comparison

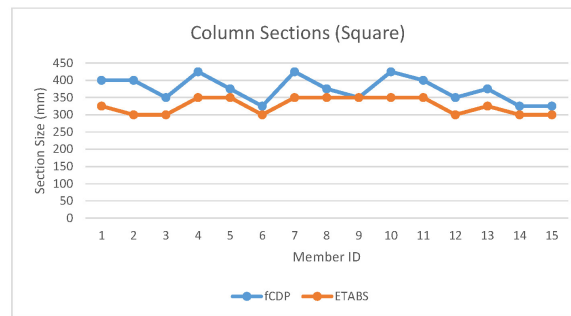


Figure 5.2: Column Section Comparison - Frame A

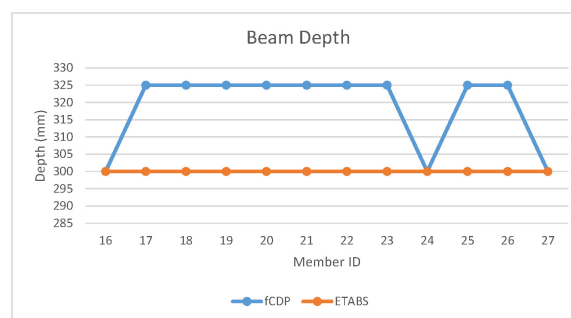


Figure 5.3: Beam Depth Comparison - Frame A

5.2.1.2 Reinforcement Comparison

Same section sizes were used by both software for these reinforcement comparisons.

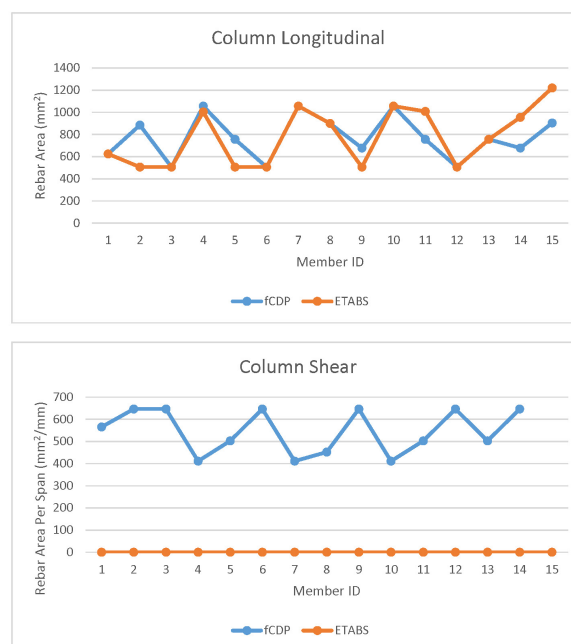


Figure 5.4: Column Reinforcement Comparison- Frame A

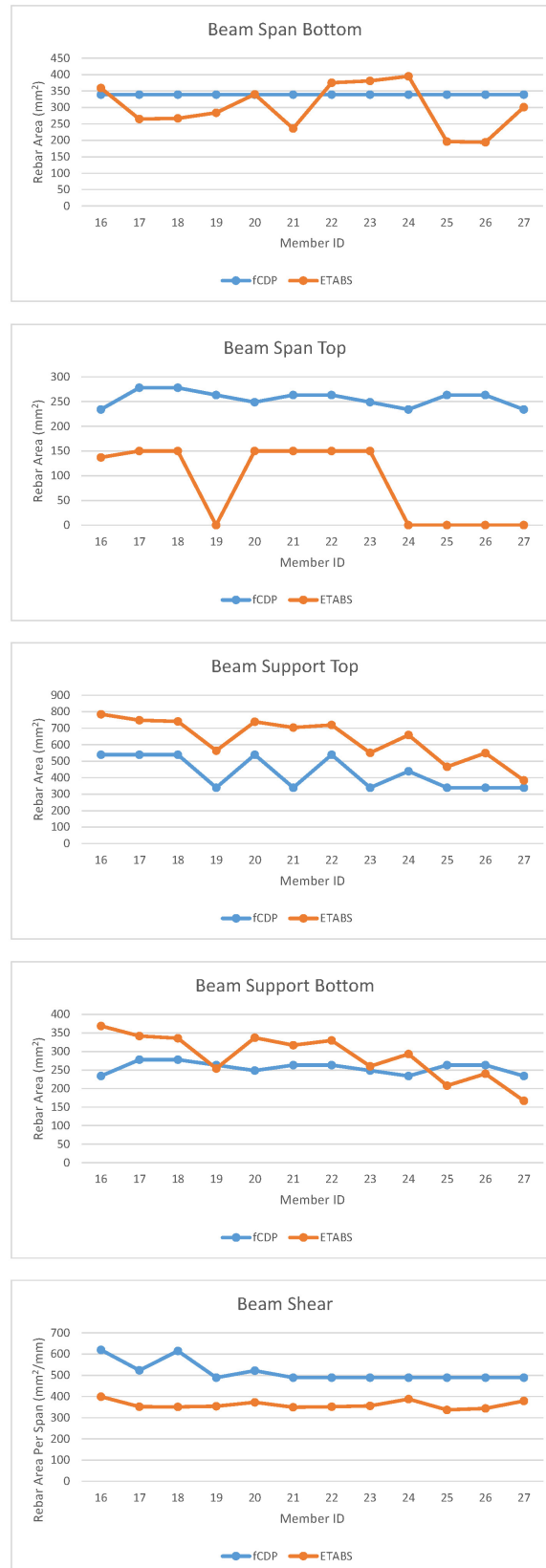


Figure 5.5: Beam Reinforcement Comparison - Frame A

5.2.2 Frame B (5 Storey x 5 Bay)

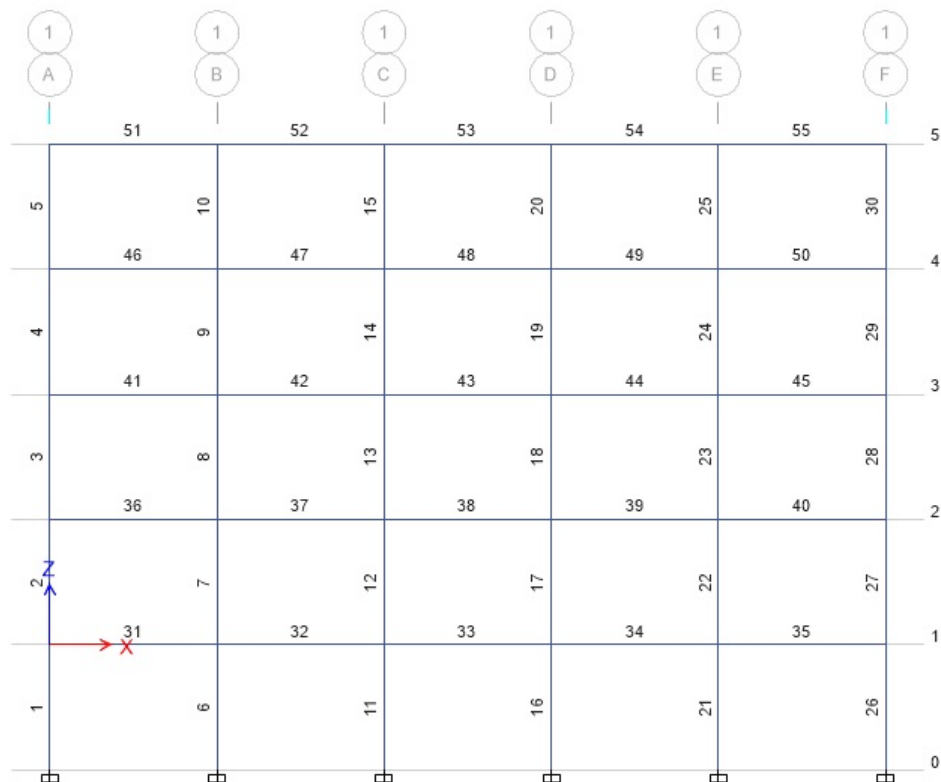


Figure 5.6: Frame B

5.2.2.1 Section Comparison

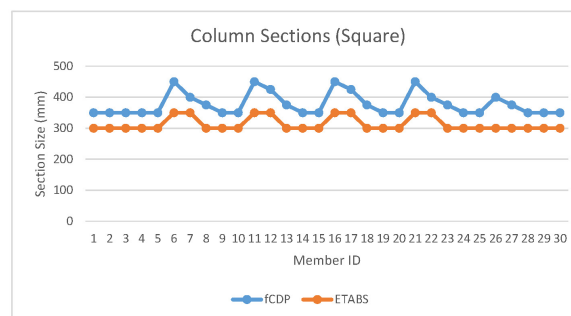


Figure 5.7: Column Section Comparison - Frame B

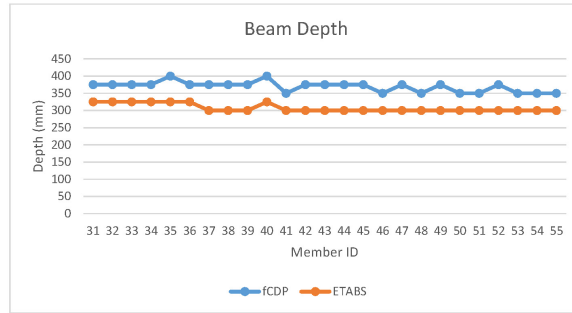


Figure 5.8: Beam Depth Comparison - Frame B

5.2.2.2 Reinforcement Comparison

Same section sizes were used by both software for these reinforcement comparisons.

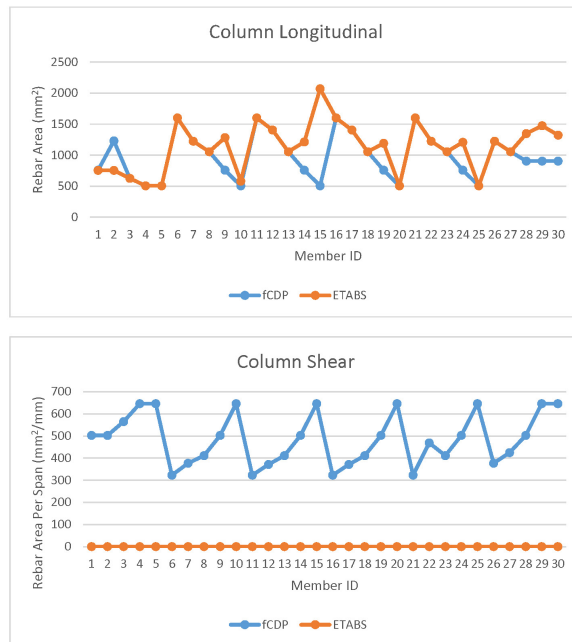


Figure 5.9: Column Reinforcement Comparison- Frame B

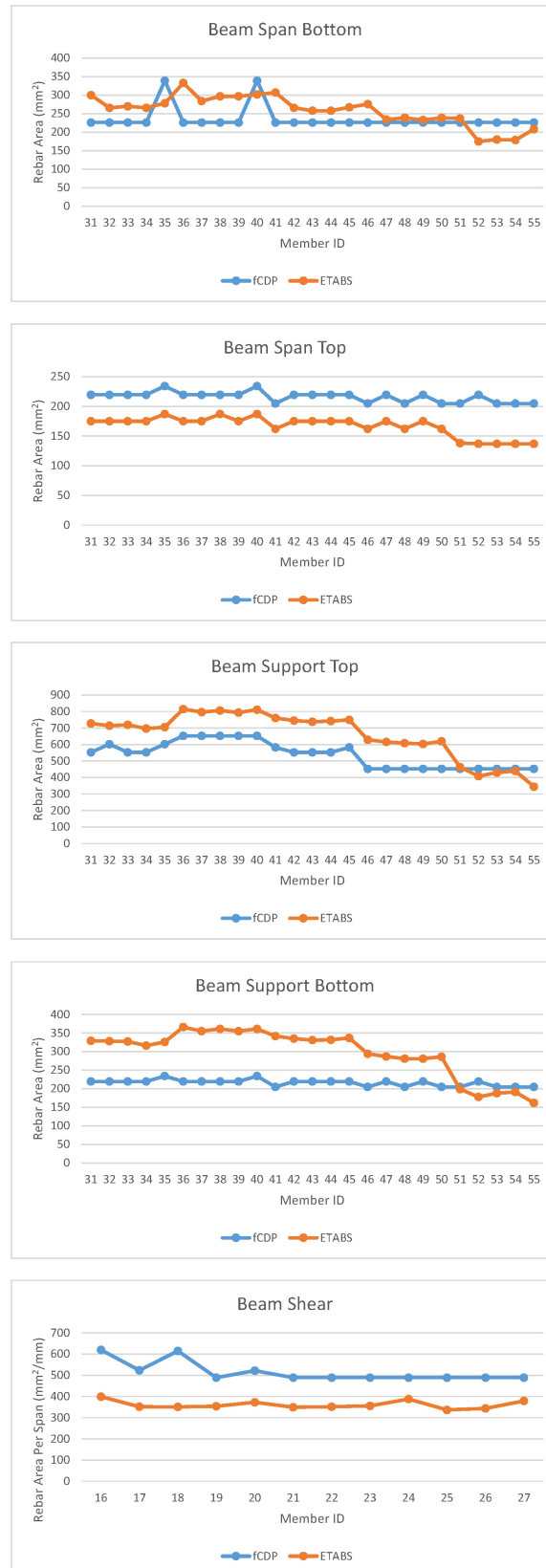


Figure 5.10: Beam Reinforcement Comparison - Frame B

5.2.3 Frame C (10 Storey x 4 Bay)

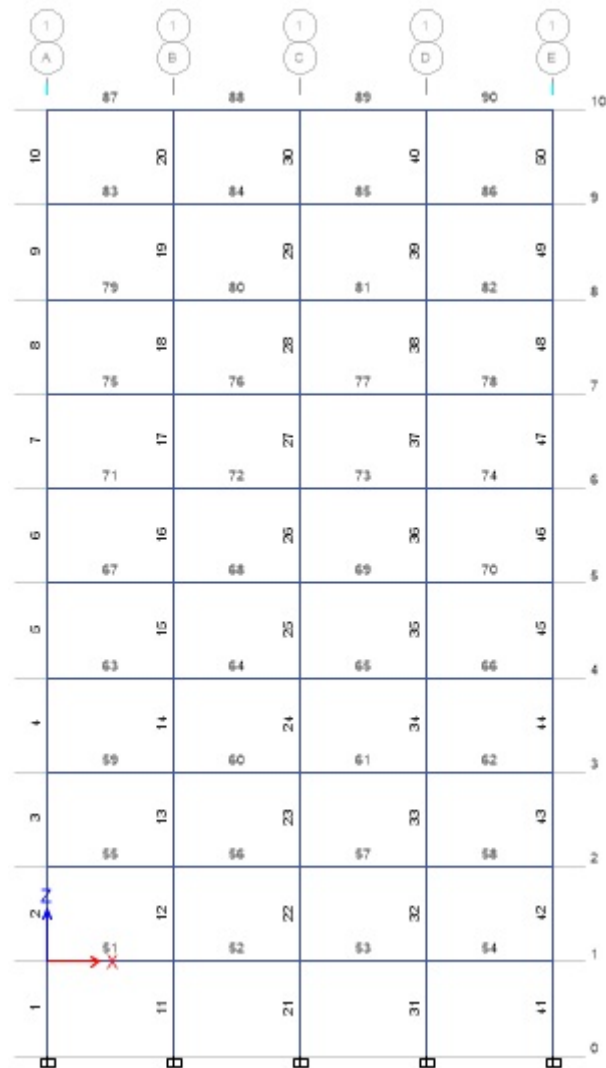


Figure 5.11: Frame C

5.2.3.1 Section Comparison

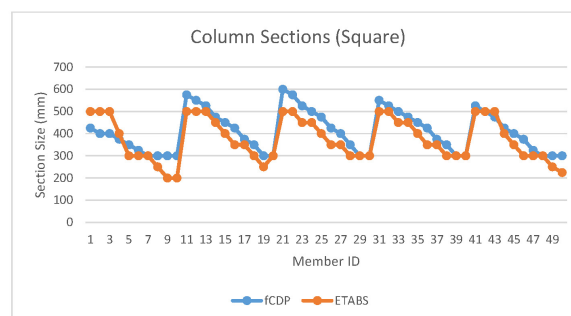


Figure 5.12: Column Section Comparison - Frame C

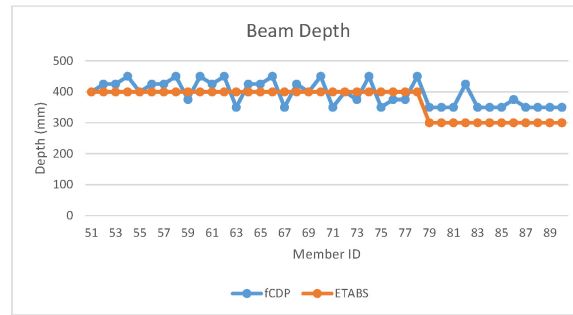


Figure 5.13: Beam Depth Comparison - Frame C

5.2.3.2 Reinforcement Comparison

Same section sizes were used by both software for these reinforcement comparisons.

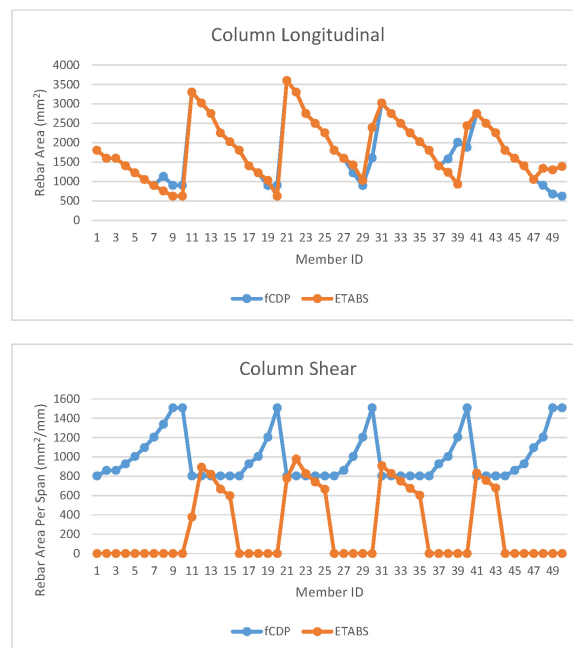


Figure 5.14: Column Reinforcement Comparison- Frame C

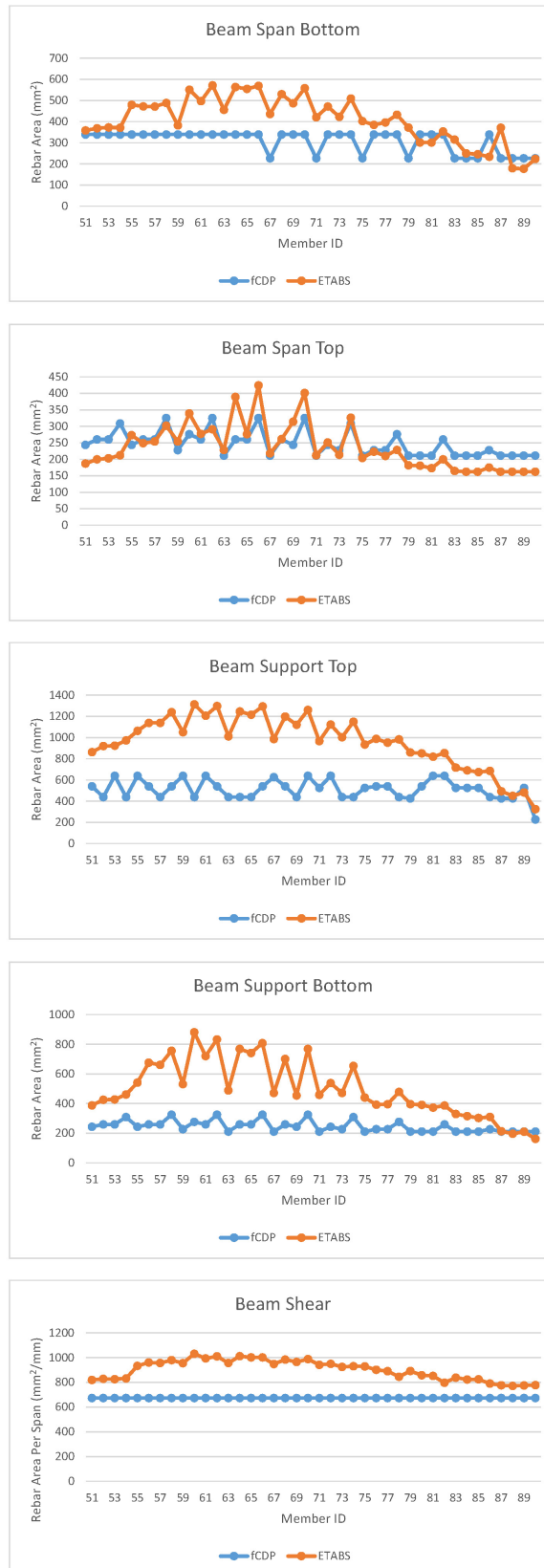


Figure 5.15: Beam Reinforcement Comparison - Frame C

Chapter 6

Conclusions

6.1 General

In this research an attempt was made to elaborately present the capacity design process of reinforced concrete plane frames for earthquake resistance; namely capacity design. Design rules for capacity design were based on those described in the Ethiopian Building Code (specifically ES EN 1998:2015) and software was developed to implement this process. This software has been written to perform analysis, regular design and capacity design. It also automates the reanalysis and redesign of failed members after increasing their sections and converts continuous reinforcement areas to discrete number of bars (or bar center-to-center spacings in case of shear reinforcing) for the program determined bar diameter. The software, Capacity Design Program for Frames (fCDP), employs the direct stiffness method to analyze frames and the Ethiopian Building Code procedures for concrete design. ETABS design outputs for three benchmark frames have been used to verify the results obtained from the developed software.

6.2 Application of Results

The developed software, Capacity Design Program for Frames (fCDP), is intended to serve as an auxiliary design tool for professionals in this field. It is expected to facilitate the proper application of the capacity design method in designing buildings, reviewing other designs and assessment of existing buildings. Since the software bases its design on the Ethiopian Code, it can be used to validate results obtained from existing software packages which have not incorporated the local code. It can also come in useful for preliminary design as may be required during conceptual or architectural design stage.

6.3 Future Use

The resulting platform from this research can serve as a ground work for including other features such as expansion to 3D analysis or design, incorporation of curved or inclined members, consideration of other structural element sections besides rectangular ones, involve the design of irregular frames and so on. In addition, individual components from the platform could also be adopted for use in other projects in any extent.

Bibliography

- [1] Rasmi Ranjan Sahoo (2008), *Analysis and Capacity Based Earthquake Resistant Design of Multi Bay Multi Storied 3D-RC Frame*, Thesis – National Institute of Technology Rourkela.
- [2] James K. Wight and James G. Macgregor (2012), *Reinforced Concrete: Mechanics and Design*, Pearson Education, Inc.
- [3] Arthur H. Nilson et al (2010), *Design of Concrete Structures*, 14th Edition, The McGraw-Hill Companies, Inc.
- [4] Bhav Priyanka and Banarase Mayur (2016), *Analysis and Capacity Based Earthquake Resistance Design of Multi Bay Multi Storied Residential Building*, Int. Journal of Engineering research and Applications Vol. 6, Issue 4.
- [5] IS 1893 (2015), *Design of Structures for Earthquake Resistance Part 1; Specific Rules for Design and Detailing of Concrete Buildings*, Ministry of Urban Development and Construction.
- [6] Ray W. Clough and Joseph Penzien (2003), *Dynamics of Structures*, 3rd Edition, Computers and Structures, Inc.
- [7] FEMA P-79 (2010), *Earthquake-Resistant Design Concepts: An Introduction to the NEHRP Recommended Seismic Provisions for New Buildings and Other Structures*, National Institute of Building Sciences - Building Seismic Safety Council, National Earthquake Hazards Reduction Program (NEHRP).
- [8] Encyclopedia of Science, Technology and Ethics (2020), *Earthquake Engineering*, accessed 04 June 2020, <<https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/earthquake-engineering>>.
- [9] T. Paulay and M. J. N. Priestley (1992), *Seismic Design of Reinforced Concrete and Masonry Buildings*, John Wiley & Sons, Inc.
- [10] Ahmed Y. Elghazouli et al (2009), *Seismic Design of Buildings to Eurocode 8*, Spon Press an imprint of Taylor and Francis.

- [11] Andrew King n.d., *Earthquake Loads and Earthquake Resistant Design of Buildings*, Building Research Association of New Zealand.
- [12] Anil K. Chopra (2012), *Dynamics of Structures: Theory and Applications to Earthquake Engineering*, 4th Edition, Pearson Education, Inc. publishing as Prentice Hall.
- [13] Charles E. Reynolds et al (2008), *Reynolds's Reinforced Concrete Designer's Handbook*, 11th Edition, Taylor and Francis.
- [14] Response Spectrum (2020), accessed 04 June 2020, <https://en.wikipedia.org/wiki/Response_spectrum>.
- [15] Structural Dynamics (2020), accessed 04 June 2020, <https://en.wikipedia.org/wiki/Structural_dynamics#Time_history_analysis>.
- [16] C.V.R. Murty et al (2012), *Some Concepts in Earthquake Behaviour of Buildings*, Gujarat State Disaster Management Authority, Government of Gujarat.
- [17] Jack P. Moehle (1996), *Displacement Based Seismic Design Criteria*, Earthquake Engineering Research Center, University of California at Berkeley.
- [18] Harsh Bakhaswala et al (2017), *Capacity Based Earthquake Resistant Design of Multi-storied RC Building*, *SJIF*, International Journal of Advanced Engineering and Research Development Vol. 4, Issue 11.
- [19] V. Ozsarac et al (2017), *Energy-based Response of Simple Structural Systems by Using Simulated Ground Motions*, *Procedia Engineering*, Vol. 199.
- [20] P Khashaee et al (2017), *Energy-Based Design Approach for Seismic Design: Energy Spectra*, accessed 20 August 2020, NIST Interagency/Internal Report (NISTIR) - 6599, <<https://www.nist.gov/publications/energy-based-design-approach-seismic-design-energy-spectra>>.
- [21] T. Paulay (1996), *Seismic Design of Concrete Structures - The Present Needs of Societies*, Paper No. 2001, Eleventh World Conference on Earthquake Engineering, Elsevier Science Ltd.
- [22] Naresh Choubisa et al (2017), *Improved Design of Five Storey Building Frame Using Capacity Based Design*, *European Journal of Advances in Engineering and Technology*, 4 (1): 7-20.
- [23] Y. Daniel Liang (2008), *Introduction to Java Programming*, 8th Edition, Pearson Higher Education.

- [24] Tutorials Point (I) Pvt. Ltd. (2014), *Computer Programming*, e-book, accessed 18 July 2018, <<https://www.tutorialspoint.com>>.
- [25] Computer Programming (2020), accessed 11 March 2020, <https://en.wikipedia.org/wiki/Computer_programming>.
- [26] Tutorials Point (I) Pvt. Ltd. (2014), *Java*, e-book, accessed 18 July 2018, <<https://www.tutorialspoint.com>>.
- [27] Elias Yilma (2017), *Cost Optimization of Reinforced Concrete Frames Using Genetic Algorithms*, Thesis – Addis Ababa University, Addis Ababa Institute of Technology.
- [28] ES EN 1992 (2015), *Design of Concrete Structures*, Ministry of Urban Development and Construction.
- [29] Bill Mosley et al (2012), *Reinforced Concrete Design to Eurocode 2*, 7th Edition, Palgrave Macmillan.
- [30] Jenny Burrridge (2016), *Practical Design to Eurocode 2*, Lecture 8: Columns, MPA, The Concrete Center.
- [31] ES EN 1990 (2015), *Basis of Structural Design*, Ministry of Urban Development and Construction.

Appendix A

Program Listings

A.1 Earthquake Load Computation and Distribution

```
public void AnalyzeMember (){
    for (Member3 memb1 : memberVector) {
        memb1.addLoads();
        memb1.computeArea();
        memb1.computeIxx();
        memb1.computeIyy();
        memb1.computeIzz();
        memb1.memLength();
        memb1.directionCosines();
        memb1.rotationMatrix();
        memb1.localStiffness();
        memb1.globalStiffness();
    }
}

public void AnalyzeStructure (){
    DetermineDof();
    CountRestrainedDOFs();
    AssembleStiffnessC();
    EquivalentLoadVectorC();
    ApplyBoundaryConditionsC();
    AssembleReactionStiffnessC();
    ComputeDisplacementsC(false);
    AssembleDisplacementsC();
    ComputeReactionsC();
    AssembleReactionsC();
    MemberEndActionsC();
    for (Member3 memb2 : memberVector){
        VandMPanel.internalActionCoordinates(memb2);
    }
}

//Earthquake Load Computation
public double Fb; //base shear force
public double Sd;
public double T1;
public double Mass;
public double Lambdasto;
public final double ct=0.075;//for concrete frame
public double H;
public int NoStories;//number of stories
```

```

public ArrayList<Double> AllYCoordinates = new ArrayList<>();
public ArrayList<Double> YCoordinates = new ArrayList<>();
public ArrayList<Double> FloorLevels = new ArrayList<>();
public int NoBays; //number of bays
public ArrayList<Double> AllXCoordinates = new ArrayList<>();
public ArrayList<Double> XCoordinates = new ArrayList<>();
public double ag;
public double S;
public double Tb;
public double q;
public final double eta=1; // for 5% viscous damping
public double Tc;
public final double beta=0.2; //lower bound factor recommended value
public double Td;
public double Gamma1; //importance factor
public double agR;
public double AlphaNot; //bedrock acceleration ratio
public final double g = 9.81;
public String IC; //importance class
public int Zone;
public int SpectraType; //elastic response spectra type
public char GroundType;
public double [][] Type1 = {{1,0.15,0.4,2},{1.2,0.15,0.5,2},
                             {1.15,0.2,0.6,2},{1.35,0.2,0.8,2},{1.4,0.15,0.5,2}};
public double [][] Type2 = {{1,0.05,0.25,1.2},{1.35,0.05,0.25,1.2},
                             {1.5,0.1,0.25,1.2},{1.8,0.1,0.3,1.2},{1.6,0.05,0.25,1.2}};
public double qo;
public char DC; //ductility class
public final double MF = 1.3; /*multiplication factor i.e.
                               Alphau/Alphaa = 1.3 for multi storey
                               multi bay frames which are regular in plan*/
public final int kw = 1; //for frames
public double PsiEi;
public final double Psi = 1;
public double Psi2i;
public char BuildingCategory;
public double StrWeight;
public ArrayList<Double> StoreyForces = new ArrayList<>();
public ArrayList<Load3> EQLoadXPos = new ArrayList<>();
public ArrayList<Load3> EQLoadXNeg = new ArrayList<>();
public void DetermineNoStoriesNoBaysH() {
    for (Joint3 j : jointVector) {
        AllYCoordinates.add(j.y);
    }
    for (int j=0; j < AllYCoordinates.size(); j++){
        if (YCoordinates.contains(AllYCoordinates.get(j))){
        }
        else{
            YCoordinates.add(AllYCoordinates.get(j));
        }
    }
    NoStories = YCoordinates.size() - 1;
    java.util.Collections.sort(YCoordinates);
    H = YCoordinates.get(YCoordinates.size() - 1) - YCoordinates.get(0);
    for (int k=0; k < YCoordinates.size() - 1; k++){
        FloorLevels.add(YCoordinates.get(k+1) - YCoordinates.get(0));
    }
    for (Joint3 j : jointVector) {
        AllXCoordinates.add(j.x);
    }

```

```

    }
    for (int j=0; j < AllXCoordinates.size(); j++){
        if (XCoordinates.contains(AllXCoordinates.get(j))){
            }
        else {
            XCoordinates.add(AllXCoordinates.get(j));
        }
    }
    NoBays = XCoordinates.size() - 1;
    java.util.Collections.sort(XCoordinates);
}
public void CalculateT1(){
    T1 = ct * Math.pow(H,(0.75));
}
public void DetermineValueOfGamma1(){
    if (IC.equals("I")){
        Gamma1 = 0.8;
    }
    else if (IC.equals("II")){
        Gamma1 = 1.0;
    }
    else if (IC.equals("III")){
        Gamma1 = 1.2;
    }
    else if (IC.equals("IV")){
        Gamma1 = 1.4;
    }
}
public void DetermineValueOfAlphaNot(){
    switch (Zone) {
        case 0:
            AlphaNot = 0;
            break;
        case 1:
            AlphaNot = 0.04;
            break;
        case 2:
            AlphaNot = 0.07;
            break;
        case 3:
            AlphaNot = 0.1;
            break;
        case 4:
            AlphaNot = 0.15;
            break;
        case 5:
            AlphaNot = 0.2;
            break;
    }
}
public void CalculateAgr(){
    agR = AlphaNot * g;
}
public void CalculateAg(){
    ag = Gamma1 * agR;
}
public void DetermineSTbTcTd(){
    if (SpectraType == 1){
        switch (GroundType) {

```

```

        case 'A':
            S = Type1[0][0];
            Tb = Type1[0][1];
            Tc = Type1[0][2];
            Td = Type1[0][3];
            break;
        case 'B':
            S = Type1[1][0];
            Tb = Type1[1][1];
            Tc = Type1[1][2];
            Td = Type1[1][3];
            break;
        case 'C':
            S = Type1[2][0];
            Tb = Type1[2][1];
            Tc = Type1[2][2];
            Td = Type1[2][3];
            break;
        case 'D':
            S = Type1[3][0];
            Tb = Type1[3][1];
            Tc = Type1[3][2];
            Td = Type1[3][3];
            break;
        case 'E':
            S = Type1[4][0];
            Tb = Type1[4][1];
            Tc = Type1[4][2];
            Td = Type1[4][3];
            break;
    }
}
else if (SpectraType == 2){
    switch (GroundType) {
        case 'A':
            S = Type2[0][0];
            Tb = Type2[0][1];
            Tc = Type2[0][2];
            Td = Type2[0][3];
            break;
        case 'B':
            S = Type2[1][0];
            Tb = Type2[1][1];
            Tc = Type2[1][2];
            Td = Type2[1][3];
            break;
        case 'C':
            S = Type2[2][0];
            Tb = Type2[2][1];
            Tc = Type2[2][2];
            Td = Type2[2][3];
            break;
        case 'D':
            S = Type2[3][0];
            Tb = Type2[3][1];
            Tc = Type2[3][2];
            Td = Type2[3][3];
            break;
        case 'E':

```

```

        S = Type2[4][0];
        Tb = Type2[4][1];
        Tc = Type2[4][2];
        Td = Type2[4][3];
        break;
    }
}
}
public void DetermineQo(){
    if (DC == 'L'){
        qo = 1.5;
    }
    else if (DC == 'M'){
        qo = 3*MF;
    }
    else if (DC == 'H'){
        qo = 4.5*MF;
    }
}
public void CalculateQ(){
    q = max((qo * kw) , 1.5);
}
public void CalculateSd(){
    if (0 <= T1 && T1 <= Tb){
        Sd = ag * S * ((2/3)+((T1/Tb)*((2.5/q) (2/3))));
    }
    else if (Tb <= T1 && T1 <= Tc){
        Sd = ag * S * eta * (2.5/q);
    }
    else if (Tc <= T1 && T1 <= Td){
        Sd = max ((ag * S * (2.5/q) * (Tc/T1)), (beta * ag));
    }
    else if (Td <= T1){
        Sd = max ((ag*S*(2.5/q)*((Tc*Td)/(T1*T1))), beta*ag);
    }
}
public void DetermineValueOfLambdasto(){
    if (T1 <= 2*Tc && NoStories >=2){
        Lambdasto = 0.85;
    }
    else {
        Lambdasto = 1;
    }
}
public void ComputePsiEi(){
    switch (BuildingCategory) {
        case 'A':
        case 'B':
        case 'G':
            Psi2i = 0.3;
            break;
        case 'C':
        case 'D':
        case 'F':
            Psi2i = 0.6;
            break;
        case 'E':
            Psi2i = 0.8;
            break;
    }
}

```

```

        default:
            Psi2i = 0.3;
            break;
    }
    PsiEi = Psi * Psi2i;
}
public double getPsi2i(){
    return Psi2i;
}
public void ComputeMass(){
    double SupImpWeight;
    double SupImpPointWeight = 0;
    double SupImpUniformWeight = 0;
    double LImpWeight;
    double LImpPointWeight = 0;
    double LImpUniformWeight = 0;
    for (Member3 k: memberVector){
        for (Load3 l : k.SupImploading) {
            if(l.type == 0){
                SupImpPointWeight += l.Y;
            }
            else if(l.type ==1){
                SupImpUniformWeight += l.Y * k.length;
            }
        }
        for (Load3 l : k.Lloading) { //Live loads
            if(l.type == 0){
                LImpPointWeight += l.Y;
            }
            else if(l.type ==1){
                LImpUniformWeight += l.Y * k.length;
            }
        }
    }
    SupImpWeight = SupImpPointWeight + SupImpUniformWeight;
    LImpWeight = LImpPointWeight + LImpUniformWeight;
    for (Member3 m: memberVector){
        m.LoadSelfweight();
        if (m.Type == 0){
            StrWeight += m.selfweight.getY()*m.length;
        }
        else if (m.Type == 1){
            StrWeight += m.selfweight.getY();
        }
    }
    Mass = 1*(StrWeight + SupImpWeight + PsiEi*LImpWeight)/g;
}
public void BaseShearForceComputation(){
    Fb = Sd * Mass * Lambdasto;
}
public ArrayList<Double> StoreyMasses = new ArrayList<>();
public ArrayList<Double> ZiMi = new ArrayList<>();
public void EarthquakeLoadStoreyDistribution() {
    double [] ArrayOfStoreyMasses = new double [FloorLevels.size()];
    double [] ArrayOfZiMi = new double [FloorLevels.size()];
    double [] ArrayOfFi = new double [FloorLevels.size()];
    double sumZiMi = 0;
    for (int j = 0; j < FloorLevels.size(); j++){
        double beamWeight = 0;

```

```

double columnWeight = 0;
double stoBeamPointLoad = 0;
double stoBeamUniformLoad = 0;
double stoLoadOnBeam;
double stoColumnPointLoad = 0;
double stoColumnUniformLoad = 0;
double stoLoadOnColumn;
for (Member3 k : memberVector) {
    if (k.Type == 0){
        if ((k.start.y == YCoordinates.get(j+1)) ||
            (k.end.y == YCoordinates.get(j+1))) {
            beamWeight += k.selfweight.getY()*k.length;
            for (Load3 l : k.SupImploading) {
                if(l.type == 0){
                    stoBeamPointLoad += l.Y;
                }
                else if(l.type ==1){
                    stoBeamUniformLoad += l.Y * k.length;
                }
            }
            for (Load3 l : k.Lloading) { //Live loads
                if(l.type == 0){
                    stoBeamPointLoad += l.Y;
                }
                else if(l.type ==1){
                    stoBeamUniformLoad += l.Y * k.length;
                }
            }
        }
        else {
        }
    }
    else if (k.Type == 1){
        if (k.start.y == YCoordinates.get(j+1)){
            columnWeight += k.selfweight.getY();
            for (Load3 l : k.SupImploading) {
                if(l.type == 0){
                    stoColumnPointLoad += l.Y;
                }
                else if(l.type ==1){
                    stoColumnUniformLoad += l.Y * k.length;
                }
            }
            for (Load3 l : k.Lloading) { //Live loads
                if(l.type == 0){
                    stoColumnPointLoad += l.Y;
                }
                else if(l.type ==1){
                    stoColumnUniformLoad += l.Y * k.length;
                }
            }
        }
        else {
        }
    }
}

stoLoadOnBeam = 1*(beamWeight + stoBeamPointLoad + stoBeamUniformLoad);
stoLoadOnCol = 1*(colWeight+stoColumnPointLoad+stoColumnUniformLoad);
ArrayOfStoreyMasses[j] = ((stoLoadOnBeam + stoLoadOnColumn)/g);

```

```

StoreyMasses.add(ArrayOfStoreyMasses[j]);
ArrayOfZiMi[j] = (ArrayOfStoreyMasses[j] * FloorLevels.get(j));
ZiMi.add(ArrayOfZiMi[j]);
}
for (int x = 0; x < FloorLevels.size(); x++){
    sumZiMi += ArrayOfStoreyMasses[x] * FloorLevels.get(x);
}
for (int h = 0; h < FloorLevels.size(); h++){
    ArrayOfFi[h] = (Fb * (ArrayOfZiMi[h]/sumZiMi));
    StoreyForces.add(ArrayOfFi[h]);
EQLoadXPos.add(new Load3(0, 1, StoreyForces.get(h), 0, 0, 0, 0, 0));
EQLoadXNeg.add(new Load3(0, 1, StoreyForces.get(h), 0, 0, 0, 0, 0));
for (Member3 k : memberVector) {
    if (k.Type == 1){
        for (int ii = 0; ii < XCoordinates.size(); ii++){
            if(k.end.x == XCoordinates.get(ii) &&
            k.end.y == YCoordinates.get(h+1)){
                double X;
                if (ii == 0 || ii == XCoordinates.size() - 1){
                    X = EQLoadXPos.get(h).getX()/NoBays/2;
                }
                else {
                    X = EQLoadXPos.get(h).getX()/NoBays;
                }
                Load3 DistEQLoad = new Load3(0, 1, X, 0, 0, 0, 0, 0);
                k.addEQLoads(DistEQLoad);
            }
            else {
            }
        }
    }
    else {
    }
}
}

public void EarthquakeLoadComputationInputs(){
    DetermineNoStoriesNoBaysH();
    CalculateT1();
    DetermineValueOfGamma1();
    DetermineValueOfAlphaNot();
    CalculateAgr();
    CalculateAg();
    DetermineSTbTcTd();
    DetermineQo();
    CalculateQ();
    CalculateSd();
    DetermineValueOfLambdasto();
    ComputePsiEi();
}

public void ResetEQResults(){
    StrWeight = 0;
    Mass = 0;
    Fb = 0;
    StoreyMasses.clear();
    ZiMi.clear();
    StoreyForces.clear();
    EQLoadXPos.clear();
    EQLoadXNeg.clear();
}

```

```

}
//SETTERS AND GETTERS//
public String getIC() {
    return IC;
}
public void setIC(String IC) {
    this.IC = IC;
}
public int getSZ() {
    return Zone;
}
public void setSZ(int SZ) {
    this.Zone = SZ;
}
public int getGT() {
    return GroundType;
}
public void setGT(char GT) {
    this.GroundType = GT;
}
public int getST() {
    return SpectraType;
}
public void setST(int ST) {
    this.SpectraType = ST;
}
public int getDC() {
    return DC;
}
public void setDC(char DC) {
    this.DC = DC;
}
public int getBC() {
    return BuildingCategory;
}
public void setBC(char BC) {
    this.BuildingCategory = BC;
}
}

```

A.2 Member Design

```
public ArrayList<Member3> FailedMembers = new ArrayList<>();
//Beam Design
public double d;
public final double Es = 2000000000;
public double Musd,s;
public double Mu*sd,s;
public double z;
public double zbal;
public double AstReq;
public double AstPro;
public double Ast;
public double AscReq;
public double AscPro;
public double Asc;
public double Xna;
public double Xnabal;
public final double EpsilonC = 0.0035;
public double EpsilonSc;
public double fsc;
public double fcm, fctm;
public double MuPhi;
public double RhoMin, RhoMax;
public void BeamDesignForFlexure(){
    for (Member3 m: memberVector){
        if (m.Type == 0){
            d = m.h m.dc;
            Xna = 0.45*d;
            Xnabal = (m.delta 0.4)*d;
            EpsilonSc = ((EpsilonC/Xnabal)/(Xnabal m.dc));
            fcm = m.fck + 8000;
            if (m.fcu == 15000){
                fctm = 1600;
            }
            else if (m.fcu == 20000){
                fctm = 1900;
            }
            else if (m.fcu == 25000){
                fctm = 2200;
            }
            else if (m.fcu == 30000){
                fctm = 2600;
            }
            else if (m.fcu == 37000){
                fctm = 2900;
            }
            else if (m.fcu == 45000){
                fctm = 3200;
            }
            else if (m.fcu == 50000){
                fctm = 3500;
            }
            else if (m.fcu == 55000){
                fctm = 3800;
            }
            else if (m.fcu == 60000){
                fctm = 4100;
            }
        }
    }
}
```

```

else if (m.fcu == 67000){
    fctm = 4200;
}
else if (m.fcu == 75000){
    fctm = 4400;
}
else if (m.fcu == 85000){
    fctm = 4600;
}
else if (m.fcu == 95000){
    fctm = 4800;
}
else if (m.fcu == 105000){
    fctm = 5000;
}
else if (m.fcu > 105000){
    fctm = (2.12 * Math.log(1 + fcm/100000))*1000;
}
if (T1>=Tc){
    MuPhi = (2*qo) 1;
}
else {
    MuPhi = 1+(2*(qo 1)*Tc/T1);
}

switch (m.DCmem) {
    case 'L':
        RhoMin = max((0.26*fctm/m.fyk),0.0013);
        break;
    case 'M':
        RhoMin = 0.5*fctm/m.fyk;
        break;
    case 'H':
        RhoMin = 0.5*fctm/m.fyk;
        break;
    default:
        RhoMin = max((0.26*fctm/m.fyk),0.0013);
        break;
}

double [] SuppSpanMo = new double [2];
SuppSpanMo[0] = m.minMz * m.delta;
SuppSpanMo[1] = m.maxMz * m.delta;
double PhiC = 0, PhiT = 0;
for (int Mo = 0 ; Mo < SuppSpanMo.length; Mo++){
    Musd,s = (Math.abs(SuppSpanMo[Mo])/(m.b*d*d*m.fcd));
    Mu*sd,s = (0.454*(m.delta 0.4)) (0.182*(m.delta 0.4)*(m.delta 0.4));
    if (Musd,s <= Mu*sd,s){
        z = d*(0.5+(Math.sqrt(0.25 (Musd,s/1.134))));
        AscReq = RhoMin*m.b*d;
        AstReq = max((Math.abs(SuppSpanMo[Mo])/(m.fyd*z)),(RhoMin*m.b*d));
        int barDia[] = {12, 14, 16, 20, 24, 32};
        boolean secSuffC = false, secSuffT = false;
        for (int ii = 0; ii < barDia.length; ii++){
int maxNumBars = (int) (((m.b*1000) (2*m.dc*1000) + 17)
                        /(barDia[ii] + 33.0));
            if (((int) (AscReq/((Math.PI*barDia[ii]*barDia[ii]/4)
                        /1000000))) <= maxNumBars){
                PhiC = barDia[ii]/1000.0;
                secSuffC = true;
                if (Mo==0){

```

```

        m.PhiCSupp = PhiC;
    }
    else if (Mo==1){
        m.PhiCSpa = PhiC;
    }

    break;
}
else {
}
}
for (int ii = 0; ii < barDia.length; ii++){
int maxNumBars = (int) (((m.b*1000)    (2*m.dc*1000) + 17)
                        /(barDia[ii] + 33.0));
if (((int) (AstReq/((Math.PI*barDia[ii]*barDia[ii]/4)
                    /1000000))) <= maxNumBars){
    PhiT = barDia[ii]/1000.0;
    secSuffT = true;
    if (Mo==0){
        m.PhiTSupp = PhiT;
    }
    else if (Mo==1){
        m.PhiTSpa = PhiT;
    }

    break;
}
else {
}
}
if (secSuffC == false || secSuffT == false){
    m.SS = 0;
}
else {
    AscPro = (Math.PI*PhiC*PhiC/4)*(Math.ceil(AstReq/
        (Math.PI*m.PhiC*m.PhiC/4)));
    AstPro = (Math.PI*PhiT*PhiT/4)*(Math.ceil(AstReq/
        (Math.PI*PhiT*PhiT/4)));
    switch (m.DCmem) {
        case 'L':
            RhoMax = 0.04;
            break;
        case 'M':
            RhoMax = (AscPro/(m.b*d))+(0.0018*m.fcd
                /(MuPhi*m.fyd*m.fyd/Es));
            break;
        case 'H':
            RhoMax = (AscPro/(m.b*d))+(0.0018*m.fcd
                /(MuPhi*m.fyd*m.fyd/Es));
            break;
        default:
            RhoMax = 0.04;
            break;
    }
    if (AstPro >= (RhoMin*m.b*d) && AstPro
        < (RhoMax*m.b*d)){
        Asc = RhoMin*m.b*d;
        Ast = AstPro;
        if (Mo==0){
            m.AscSpan = Asc;
            m.AstSpan = Ast;

```

```

        m.PRRSpan = Ast/AstReq;
    }
    else if (Mo==1) {
        m.AscSupp = Asc;
        m.AstSupp = Ast;
        m.PRRSupp = Ast/AstReq;
    }
}
else if (AstPro < RhoMin*m.b*d){
    Asc = RhoMin*m.b*d;
    Ast = RhoMin*m.b*d;
    if (Mo==0){
        m.AscSpan = Asc;
        m.AstSpan = Ast;
        m.PRRSpan = Ast/AstReq;
    }
    else if (Mo==1) {
        m.AscSupp = Asc;
        m.AstSupp = Ast;
        m.PRRSupp = Ast/AstReq;
    }
}
else {
    m.SS = 0;
}
}
else {
    if (m.dc/Xnabal <= 0.38){
        fsc = m.fyd;
    }
    else {
        fsc = Es*EpsilonSc;
    }
    zbal = d (0.8*Xnabal/2);
    AscReq = max(((Musd,s*Musd,s)*m.fcd*m.b*d*d)/(fsc*(d m.dc)))
        ,(RhoMin*m.b*d));
    AstReq = max(((Musd,s*m.fcd*m.b*d*d)/(m.fyd*zbal)) +
        (AscReq*(fsc/m.fyd)) ,(RhoMin*m.b*d));
    int barDia[] = {12, 14, 16, 20, 24, 32};
    boolean secSuffC = false, secSuffT = false;
    for (int ii = 0; ii < barDia.length; ii++){
        int maxNumBars = (int) (((m.b*1000)
            (2*m.dc*1000) + 17)/(barDia[ii] + 33.0));
        if (((int) (AscReq/((Math.PI*barDia[ii]*barDia[ii]/4)
            /1000000))) <= maxNumBars){
            PhiC = barDia[ii]/1000.0;
            secSuffC = true;
            if (Mo==0){
                m.PhiCSupp = PhiC;
            }
        }
        else if (Mo==1){
            m.PhiCSpa = PhiC;
        }
        break;
    }
    else {
    }
}
}

```

```

    for (int ii = 0; ii < barDia.length; ii++){
        int maxNumBars = (int) ((m.b*1000)
                                (2*m.dc*1000) + 17)/(barDia[ii] + 33.0));
    if (((int) (AstReq/((Math.PI*barDia[ii]*barDia[ii]/4)
                        /1000000))) <= maxNumBars){
        PhiT = barDia[ii]/1000.0;
        secSuffT = true;
        if (Mo==0){
            m.PhiTSupp = PhiT;
        }
    else if (Mo==1){
        m.PhiTSpa = PhiT;
    }
        break;
    }
    else {
    }
}
if (secSuffC == false || secSuffT == false){
    m.SS = 0;
}
else {
    AscPro = (Math.PI*PhiC*PhiC/4)*(Math.ceil(AscReq
                                                /(Math.PI*PhiC*PhiC/4)));
    AstPro = (Math.PI*PhiT*PhiT/4)*(Math.ceil(AstReq
                                                /(Math.PI*PhiT*PhiT/4)));
    switch (m.DCmem) {
        case 'L':
            RhoMax = 0.04;
            break;
        case 'M':
            RhoMax = (AscPro/(m.b*d))+(0.0018*m.fcd
            /(MuPhi*m.fyd*m.fyd/Es));
            break;
        case 'H':
            RhoMax = (AscPro/(m.b*d))+(0.0018*m.fcd
            /(MuPhi*m.fyd*m.fyd/Es));
            break;
        default:
            RhoMax = 0.04;
            break;
    }
    if ((AscPro - AscReq) >= (AstPro - AstReq)){
    if (AstPro >= (rhoMin*m.b*d) && AstPro < (rhoMax*m.b*d)){
        Ast = AstPro;

        if (Mo==1){
            m.AstSpan = Ast;
            m.PRRSpan = Ast/AstReq;
        }
        else if (Mo==0) {
            m.AstSupp = Ast;
            m.PRRSupp = Ast/AstReq;
        }
    }
    else if (AstPro < rhoMin*m.b*d){
        Ast = rhoMin*m.b*d;
        if (Mo==1){
            m.AstSpan = Ast;

```



```

        m.SpanCNumBars = (int) (m.AscSpan/(Math.PI*m.PhiC*m.PhiC/4));
    }
    else {
        m.SpanCNumBars = 2;
    }
    if (((int) (m.AstSpan/(Math.PI*m.PhiT*m.PhiT/4))) > 2){
        m.SpanTNumBars = (int) (m.AstSpan/(Math.PI*m.PhiT*m.PhiT/4));
    }
    else {
        m.SpanTNumBars = 2;
    }
    if (((int) (m.AscSupp/(Math.PI*m.PhiC*m.PhiC/4))) > 2){
        m.SuppCNumBars = (int) (m.AscSupp/(Math.PI*m.PhiC*m.PhiC/4));
    }
    else {
        m.SuppCNumBars = 2;
    }
    if (((int) (m.AstSupp/(Math.PI*m.PhiT*m.PhiT/4))) > 2){
        m.SuppTNumBars = (int) (m.AstSupp/(Math.PI*m.PhiT*m.PhiT/4));
    }
    else {
        m.SuppTNumBars = 2;
    }
}

public double AswPerSReq, AswPerSmin;
public double Vrdmax;
public double theta;
final public double PhiSw = 0.006;
public double PhiLmin;
public void BeamDesignForShear(){
    for (Member3 m: memberVector){
        if (m.Type == 0){
            theta = (0.5*Math.asin(m.Vsdb/(0.18*m.b*d*m.fck*
                (1 (m.fck/250000))))));

            if (theta < Math.toRadians(45)){
                Vrdmax = 0.124*m.b*d*(1 (m.fck/250000))*m.fck;
                if (Vrdmax >= m.Vsdb){
                    theta = Math.toRadians(22);
                }
                else {
                }
            }
            AswPerSReq = m.Vsdb/(0.78*d*m.fwyk*(1/(Math.tan(theta))));

            PhiLmin = Math.min(m.PhiC,m.PhiT);
            switch (m.DCmem) {
                case 'L':
                    AswPerSmin = 0.08*1000*(Math.sqrt(m.fck/1000))
                        *m.b/m.fwyk;
                    break;
                case 'M':
                    AswPerSmin = (2*Math.PI*PhiSw*PhiSw/4)/Math.min
                        (Math.min(m.h/4,24*PhiSw),Math.min(0.225,8*PhiLmin));
                    break;
                case 'H':
                    AswPerSmin = (2*Math.PI*PhiSw*PhiSw/4)/Math.min
                        (Math.min(m.h/4,24*PhiSw),Math.min(0.175,6*PhiLmin));
                    break;
                default:
                    AswPerSmin = 0.08*1000*(Math.sqrt(m.fck/1000))

```

```

        *m.b/m.fwyk;
        break;
    }
    if (AswPerSReq >= AswPerSmin){
        m.AswPerSb = AswPerSReq;
    }
    else {
        m.AswPerSb = AswPerSmin;
    }
}
else {
    m.SSS = 0;
}
}
else{
}
}
}
//Column Design
public double Lambda;
public double lo , lo1 , lo2;
public double i;
public double k1,k2;//for stiffness calculation
public double Lambdalim;
public final double A = 0.7;
public final double B = 1.1;
public double C = 0.7;//for unbraced members
public double n;
public double Med;
public double Mmin;
public double emin , etot , e0 , e01 , e02 , ea , e2;
public double K1, K2, K2cal; //for second order moment calculation
public final double PhiEf = 0.87;
public double Nud, Nbal;
public double EpsilonS;
public double fs;
public double s;
public double AsN;
public double AsM;
public double AsPro;
public double Nobars;//number of bars for column
public double nud;
public double nudlim;
public ArrayList<Member3> FailedColumnsArray = new ArrayList<>();
public void ColumnDesign(){
    for (Member3 m : memberVector){
        if (m.Type ==1){
            switch (m.DCmen) {
                case 'L':
                    nudlim = 1;
                    RhoMin = max((0.1*(max(Math.abs(m.minAx),
Math.abs(m.maxAx)))/(m.A*m.fyd)),0.002);
                    RhoMax = 0.04;
                    break;
                case 'M':
                    nudlim = 0.65;
                    RhoMin = 0.01;
                    RhoMax = 0.04;
                    break;
            }
        }
    }
}

```

```

        case 'H':
            nudlim = 0.55;
            RhoMin = 0.01;
            RhoMax = 0.04;
            break;
        default:
            nudlim = 1;
            RhoMin = max((0.1*(max(Math.abs(m.minAx),
Math.abs(m.maxAx)))/(m.A*m.fyd)),0.002);
            RhoMax = 0.04;
            break;
    }
    nud = (max(Math.abs(m.minAx),Math.abs(m.maxAx)))/(m.A*m.fcd);
    if (nud <= nudlim){
        double ColumnStiff;
        double Beam1Stiff = 0;
        double Beam2Stiff = 0;
        double Beam3Stiff = 0;
        double Beam4Stiff = 0;
        ColumnStiff = m.Iz/m.length;
        for (Member3 k : memberVector){
            if (k.Type == 0 && (m.end == k.end)){
                Beam1Stiff = 2*k.Iz/k.length;
            }
            else if (k.Type == 0 && (m.end == k.start)){
                Beam2Stiff = 2*k.Iz/k.length;
            }
            else if (k.Type == 0 && (m.start == k.end)){
                Beam3Stiff = 2*k.Iz/k.length;
            }
            else if (k.Type == 0 && (m.start == k.start)){
                Beam4Stiff = 2*k.Iz/k.length;
            }
            else{
            }
        }
        if (Beam1Stiff + Beam2Stiff == 0){
            k1 = 0.1;
        }
        else {
            k1 = Math.max((ColumnStiff)/(Beam1Stiff+Beam2Stiff),0.1);
        }
        if (Beam3Stiff + Beam4Stiff == 0){
            k2 = 0.1;
        }
        else {
            k2 = Math.max((ColumnStiff)/(Beam3Stiff+Beam4Stiff),0.1);
        }
        lo1 = m.length*(Math.sqrt(1+(10*((k1*k2)/(k1+k2)))));
        lo2 = m.length*(1+(k1/(1+k1)))*(1+(k2/(1+k2)));
        lo = Math.max(lo1,lo2);
        i = Math.sqrt(m.Iz/m.A);
        Lambda = lo/i;
        n = nud;
        Lambdalim = 20*A*B*C/Math.sqrt(n);
        if (Lambda < Lambdalim){
            ShortColumnDesign(m);
        }
        else {

```

```

        SlenderColumnDesign(m);
    }
}
else {
    m.SS = 0;
}
}
else {
}
if (m.SS == 0){
    FailedColumnsArray.add(m);
}
else {
}
}
}
public void ShortColumnDesign(Member3 m){
    emin = Math.max(m.h/30, 0.020);
    Mmin = (max(Math.abs(m.minAx), Math.abs(m.maxAx)))*emin;
    Med = (max((max(Math.abs(m.minMz), Math.abs(m.maxMz)))
        + ((max(Math.abs(m.minAx), Math.abs(m.maxAx)))*lo/400)
        ,Mmin))*m.CMMF;
    m.Med = Med;
    d = m.h m.dc;
    ColumnEfficiencyCheck(m);
    if (m.SS == 0) {
    }
    else {
        if (RhoMin*m.A <= m.AsReq && m.AsReq < RhoMax*m.A){
            m.As = m.AsReq;
            int barDia[] = {12, 14, 16, 20, 24, 32};
            for (int ii = 0; ii < barDia.length; ii++){
int NumOfBars = (int) (Math.ceil(m.As/(Math.PI*barDia[ii]
                *barDia[ii]/4000000.0)));
                if (NumOfBars % 2 > 0){
                    NumOfBars = NumOfBars +1 ;
                }
                else {
                }
int maxNumBars = (int) (((m.b*1000) (2*m.dc*1000) + 17)
                    /(barDia[ii] + 33.0));
                    if (NumOfBars > 4){
                        if ((NumOfBars/2) < maxNumBars){
                            m.NumBars = NumOfBars;
                            break;
                        }
                        else {
                        }
                    }
                    else {
                        m.NumBars = 4;
                    }
                }
            }
            else {
                m.SS = 0;
            }
        }
    }
}
}

```

```

public void SlenderColumnDesign(Member3 m){
    d = m.h m.dc;
    emin = Math.max(m.h/30, 0.020);
    if (Math.abs(m.maxMz) < Math.abs(m.minMz)){
        e01 = m.maxMz/(max(Math.abs(m.minAx),Math.abs(m.maxAx)));
        e02 = m.minMz/(max(Math.abs(m.minAx),Math.abs(m.maxAx)));
    }
    else {
        e01 = m.minMz/(max(Math.abs(m.minAx),Math.abs(m.maxAx)));
        e02 = m.maxMz/(max(Math.abs(m.minAx),Math.abs(m.maxAx)));
    }
    e0 = Math.max((0.6*e02+0.4*e01), 0.4*e02);
    ea = lo/400;
    do {
        if (K2cal == 0){
            K2=1;
        }
        else {
            K2 = K2cal;
        }
        K1 = Math.max((1 + ((0.35+((m.fck/1000)/200) (Lambda/150))
                                                                *PhiEf)),1);
        e2 = (K1*K2*lo*lo*m.fyk)/(Math.PI*Math.PI*103500*d*1000);
        etot = Math.max((e0 + ea + e2), emin);
        Med = ((max(Math.abs(m.minAx),Math.abs(m.maxAx)))*etot)*m.CMMF;
        m.Med = Med;
        ColumnEfficiencyCheck(m);
        Nud = m.fcd*m.h*m.b + m.fyd*(m.AsReq);
        Nbal = 0.29*m.fck*m.h*m.b;
        K2cal = Math.min(((Nud (max(Math.abs(m.minAx),Math.abs
                                                                (m.maxAx)))/(Nud Nbal)),1);
    } while (!(Math.abs(K2 - K2cal) < 0.001));
    if (m.SS == 0) {
    }
    else {
        if (RhoMin*m.A <= m.AsReq && m.AsReq < RhoMax*m.A){
            m.As = m.AsReq;
        }
        else if (m.AsReq < RhoMin*m.A){
            m.As = RhoMin*m.A;
            int barDia[] = {12, 14, 16, 20, 24, 32};
            for (int ii = 0; ii < barDia.length; ii++){
int NumOfBars = (int) (Math.ceil(m.As/(Math.PI*barDia[ii]
                                *barDia[ii]/4000000.0)));
                if (NumOfBars % 2 > 0){
                    NumOfBars = NumOfBars +1 ;
                }
                else {
                }
            }
int maxNumBars = (int) (((m.b*1000) (2*m.dc*1000) + 17)
                        /(barDia[ii] + 33.0));
            if (NumOfBars > 4){
                if ((NumOfBars/2) < maxNumBars){
                    m.NumBars = NumOfBars;
                    break;
                }
                else {
                }
            }
        }
    }
}

```

```

        else {
            m.NumBars = 4;
        }
    }
    else {
        m.SS = 0;
    }
}

}

public class columnDesignCheck {
    public static ArrayList<interPoint> DevelopInteractionCoordinates
        (Member3 m, double as1, double as2) {
        ArrayList<interPoint> interactionCoordinates = new ArrayList<>();
        interPoint p1;
        double E = m.getE();
        double Es = 200000000;
        double ycpara, ycrect, yc, ys1, ys2;
        double x; //neutral depth
        double fcpara, fcrect, fctotal; //compressive force in concrete
        double fs1, fs2; //forces in tension and compression steel
        double eyd = m.fyd / Es, es1, es2; //steel strains
        double ecx, ec, ec2; //concrete strains
        double V, M;
        //POINT 0: pure tension
        es1 = 0.025;
        es2 = 0.025;
        ec = 0;
        fs1 = as1 * m.fyd;
        fs2 = as2 * m.fyd;
        M = 0;
        fctotal = 0;
        V = ( fs1 + ( fs2 ));
        p1 = new interPoint();
        p1.x = M;
        p1.y = V;
        interactionCoordinates.add(p1);
        //POINT 1: pure flexure
        double n = 0.8 * m.fcd * m.b;
        double o = 0.0035 * Es * as2 * m.fyd * as1;
        double p = m.dc * as2 * 0.0035 * Es;
        double x1 = ( o + Math.sqrt(o * o - 4 * n * p)) / (2 * n);
        double x2 = ( o - Math.sqrt(o * o - 4 * n * p)) / (2 * n);
        if (x1 > x2) {
            x = x1;
        } else {
            x = x2;
        }
        fs2 = 0.0035 * Es * (x - m.dc) / x;
        double cc = m.fcd * 0.8 * x * m.b;
        double cs = fs2 * as2;
        double ts = m.fyd * as1;
        V = 0;
        M = (cc * (0.5 * m.h - 0.4 * x) + cs * (0.5 * m.h - m.dc)
            + ts * (m.h - m.dc - 0.5 * m.h));
        p1 = new interPoint();
        p1.x = M;
        p1.y = V;
        interactionCoordinates.add(p1);
        //POINT 2: yielding of compression reinforcement steel
    }
}

```

```

x = (0.0035 * m.dc) / (0.0035 * eyd);
cc = m.fcd * 0.8 * x * m.b;
cs = m.fyd * as2;
ts = m.fyd * as1;
V = (cc + cs * ts);
M = (cc * (0.5 * m.h - 0.4 * x) + cs * (0.5 * m.h - m.dc)
      + ts * (m.h - m.dc - 0.5 * m.h));
p1 = new interPoint();
p1.x = M;
p1.y = V;
interactionCoordinates.add(p1);
//POINT 3: balanced failure
x = 0.0035 * (m.h - m.dc) / (eyd + 0.0035);
es2 = 0.0035 * (x - m.dc) / x;
if (es2 < eyd) {
    fs2 = es2 * Es;
} else {
    fs2 = m.fyd;
}
cc = m.fcd * 0.8 * x * m.b;
cs = fs2 * as2;
ts = m.fyd * as1;
V = (cc + cs * ts);
M = (cc * (0.5 * m.h - 0.4 * x) + cs * (0.5 * m.h - m.dc)
      + ts * (m.h - m.dc - 0.5 * m.h));
p1 = new interPoint();
p1.x = M;
p1.y = V;
interactionCoordinates.add(p1);
//POINT 4: zero strain in tension reinforcement
x = m.h - m.dc;
es2 = 0.0035 * (x - m.dc) / x;
if (es2 < eyd) {
    fs2 = es2 * Es;
} else {
    fs2 = m.fyd;
}
cc = m.fcd * 0.8 * x * m.b;
cs = fs2 * as2;
V = (cc + cs);
M = (cc * (0.5 * m.h - 0.4 * x) + cs * (0.5 * m.h - m.dc));
p1 = new interPoint();
p1.x = M;
p1.y = V;
interactionCoordinates.add(p1);
//POINT 5: pure compression
V = (m.fcd * (m.b * m.h * (as1 + as2)) + m.fyd * (as1 + as2));
M = 0;
p1 = new interPoint();
p1.x = M;
p1.y = V;
interactionCoordinates.add(p1);
}

public static double ComputeColumnEfficiency
    (ArrayList<interPoint> interactionCoordinates, interPoint load) {
    double percentageEfficiency;
    int pointLocationTriangleIndex;
    pointLocationTriangleIndex = CheckInteraction(interactionCoordinates, load.x, load.y);
    if (pointLocationTriangleIndex == 100) {

```

```

        percentageEfficiency = 0;
    } else {
        interPoint p3 = interactionCoordinates.get(pointLocationTriangleIndex);
        interPoint p4 = interactionCoordinates.get(pointLocationTriangleIndex + 1);
        interPoint p1 = new interPoint(0.0, interactionCoordinates.get(5).y);
        interPoint p2 = load;
        float delx21 = (float) (p2.x - p1.x);
        float delx43 = (float) (p4.x - p3.x);
        if (Math.abs(delx21 - 0) < 0.001) {
            delx21 = (delx21 > 0) ? 0.01f : -0.01f;
        }
        if (Math.abs(delx43 - 0) < 0.001) {
            delx43 = (delx43 > 0) ? 0.01f : -0.01f;
        }
        double m1 = (p2.y - p1.y) / delx21;
        double m2 = (p4.y - p3.y) / delx43;
        double b1 = p1.y - p1.x * m1;
        double b2 = p3.y - p3.x * m2;
        double y = (b2 * m1 - b1 * m2) / (m1 - m2);
        double x = (y - b1) / m1;
        double l1 = Math.sqrt((p1.x - x) * (p1.x - x) + (p1.y - y) * (p1.y - y));
        double l2 = Math.sqrt((p1.x - load.x) * (p1.x - load.x) + (p1.y - load.y)
            * (p1.y - load.y));
        percentageEfficiency = l2 / l1;
    }
    //check interaction if out of the region return 1.0
    //else get the two points(other than the contour origin from the
        triangle that contains the point
    return percentageEfficiency;
}

public static int CheckInteraction(ArrayList<interPoint> interactionCoordinates,
    Double Mz, Double Fy) {
    boolean insideRegion = false;
    //identify sixth point in interactionCoordinates and form contour origin point
    interPoint contourOrigin = new interPoint();
    interPoint designAction = new interPoint(Mz, Fy);
    interPoint maxM = interactionCoordinates.get(3);
    contourOrigin.x = 0.0;
    contourOrigin.y = maxM.y;
    ArrayList<interTriangle> regions = new ArrayList<>();
    //form and store all triangles
    //the triangles are formed in two phases:i.e. first all triangles above the maxM point
    //and then all points below maxM.

    for (int i = 1; i <= 4; i++) {
        interTriangle trigon = new interTriangle(interactionCoordinates.get(i - 1),
            interactionCoordinates.get(i), contourOrigin);
        regions.add(trigon);
    }
    for (int i = 4; i < 5; i++) {
        interTriangle trigon = new interTriangle(contourOrigin,
            interactionCoordinates.get(i - 1), interactionCoordinates.get(i));
        regions.add(trigon);
    }
    //for each triangle, check whether Mz,Fy Lies inside of its region
    int pointLocation = 100;//in which region is the point located?
    for (int i = 0; i < 5; i++) {
        if (isInsideTriangle(regions.get(i), designAction)) {
            pointLocation = i;
        }
    }
}

```

```

        insideRegion = true;
    }
}
return pointLocation;
}
public static void ColumnEfficiencyCheck(Member3 m) {
    int barDia[] = {12, 14, 16, 20, 24, 32};
    boolean secSuff = false;
    for (int i = 0; i < barDia.length; i++){
        int maxNumBars = (int) (((m.b*1000) - (2*m.dc*1000) + 17)/(barDia[i] + 33.0));
        for (int j = 2; j < maxNumBars + 1; j++){
            double as1 = (j*Math.PI*barDia[i]*barDia[i]/4)/1000000;
            double as2 = (j*Math.PI*barDia[i]*barDia[i]/4)/1000000;
            ArrayList<interPoint> res = DevelopInteractionCoordinates(m, as1, as2);
            interPoint load = new interPoint(m.Med,
                (Math.max(Math.abs(m.minAx), Math.abs(m.maxAx))));
            CheckInteraction(res, load.x, load.y);
            double eff = ComputeColumnEfficiency(res, load);
            if (eff > 0.50 && eff < 1) {
                m.AsReq = as1 + as2;
                m.rebarDia = barDia[i];
                m.NumBars = 2*j;
                secSuff = true;
                DetermineColumnCapacity(m, res);
                break;
            }

            else //if (eff < 0.5 || eff > 1) Check another reinforcement amount;
            }
        }
        if (secSuff == true){
            break;
        }
        else {
        }
    }
    if (secSuff == true){
    }
    else {
        m.SS = 0;
    }
}

public static void DetermineColumnCapacity(Member3 m, ArrayList<interPoint> interCoor) {
    interPoint Pload = new interPoint (0, (Math.max(Math.abs(m.minAx), Math.abs(m.maxAx))));
    for (int i = 0; i < interCoor.size() - 1; i++){
        if (Pload.y > interCoor.get(i).y && Pload.y < interCoor.get(i+1).y){
            double slope; // line equation: y = mx + b
            double intercept;
            slope = (interCoor.get(i+1).y - interCoor.get(i).y) /
                (interCoor.get(i+1).x - interCoor.get(i).x);
            intercept = interCoor.get(i).y - (slope * interCoor.get(i).x);
            m.Mcap = (Pload.y - intercept) / slope;
            break;
        }
        else {
        }
    }
}
}
}

```

//Column Design for Shear

```

public void ColumnDesignForShear(){
    for (Member3 m: memberVector){
        if (m.Type == 1){
            theta = (0.5*Math.asin(m.Vsdc/(0.18*m.b*d*m.fck
                *(1 (m.fck/250000)))));
            if (theta < Math.toRadians(45)){
                Vrdmax = 0.124*m.b*d*(1 (m.fck/250000))*m.fck;
                if (Vrdmax >= m.Vsdc){
                    theta = Math.toRadians(22);
                }
                else {
                }
                AswPerSReq = m.Vsdc/(0.78*d*m.fwyk*(1/(Math.tan(theta))));
                PhiLmin = m.rebarDia;
                double bo = m.h (2*m.dc);
                switch (m.DCmem) {
                    case 'L':
                        AswPerSmin = 0.08*1000*(Math.sqrt(m.fck/1000))
                            *m.b/m.fwyk;

                        break;
                    case 'M':
                        AswPerSmin = (2*Math.PI*PhiSw*PhiSw/4)/Math.min
                            (Math.min(bo/2,0.175),8*PhiLmin);

                        break;
                    case 'H':
                        AswPerSmin = (2*Math.PI*0.008*0.008/4)/Math.min
                            (Math.min(bo/3,0.125),6*PhiLmin);

                        break;
                    default :
                        AswPerSmin = 0.08*1000*(Math.sqrt(m.fck/1000))
                            *m.b/m.fwyk;

                        break;
                }
                if (AswPerSReq >= AswPerSmin){
                    m.AswPerSc = AswPerSReq;
                }
                else {
                    m.AswPerSc = AswPerSmin;
                }
            }
            else {
                m.SSS = 0;
            }
        }
        else{
        }
    }
}

public void ToDiscreteShearReinf (Member3 m) {
    int barDia[] = {6, 8, 10, 12};
    for (int ii = 0; ii < barDia.length; ii++){
        if (m.Type == 0){
            m.StirSpa = (2*Math.PI*barDia[ii]
                *barDia[ii]/4/1000000.0)/m.AswPerSb;

            if (m.StirSpa > 0.1){
                m.StirDia = barDia[ii]/1000.0;
                break;
            }
        }
    }
}

```

```

        else {
        }
    }
    else {
        m.TieSpa = (2*Math.PI*barDia[ii]
                    *barDia[ii]/4/1000000.0)/m.AswPerSc;
        if (m.TieSpa > 0.1){
            m.TieDia = barDia[ii]/1000.0;
            break;
        }
        else {
        }
    }
}
}

public void DesignMembers(){
    BeamDesignForFlexure();
    ColumnDesign();
    FailedMembers.clear();
    for (Member3 m : memberVector) {
        if (m.SS == 0){
            FailedMembers.add(m);
            m.failedTimes++;
        }
        else {
        }
    }
    ArrayList<Member3> FailedBeams = new ArrayList<>();
    ArrayList<Member3> FailedColumns = new ArrayList<>();
    for (Member3 m : FailedMembers){
        if (m.Type == 0){
            FailedBeams.add(m);
        }
        else if (m.Type == 1){
            FailedColumns.add(m);
        }
    }
}

public void ResetDesignResults(){
    d = 0;
    Musd,s = 0; Mu*sd,s = 0;
    z = 0; zbal = 0;
    AstReq = 0; AstPro = 0; Ast = 0;
    AscReq = 0; AscPro = 0; Asc = 0;
    Xna = 0; Xnabal = 0;
    EpsilonSc = 0;
    fsc = 0; fcm = 0; fctm = 0;
    MuPhi = 0;
    RhoMin = 0; RhoMax = 0;
    for (Member3 m : memberVector) {
        m.AscSupp = 0; m.AstSupp = 0; m.PRRSupp = 0;
        m.AscSpan = 0; m.AstSpan = 0; m.PRRSpan = 0;
        m.SS = 1;
        m.AsReq = 0; m.As = 0;
        m.s = 0;
        m.fs = 0; m.fsc = 0;
        m.Mcap = 0;
    }
    Lambda = 0;
}

```

```

    lo = 0; lo1 = 0; lo2 = 0;
    i = 0;
    k1 = 0; k2 = 0;
    Lambdalim = 0;
    n = 0;
    Med = 0; Mmin = 0;
    emin = 0; etot = 0; e0 = 0; e01 = 0; e02 = 0; ea = 0; e2 = 0;
    K1 = 0; K2 = 0; K2cal = 0;
    Nud = 0; Nbal = 0;
    EpsilonS = 0;
    fs = 0;
    s = 0;
    AsN = 0; AsM = 0; AsPro = 0;
    Nobars = 0;
    nud = 0; nudlim = 0;
        FailedColumnsArray.clear();
}

public void ResetColumnDesignResults(){
d =0;
rhoMin = 0; rhoMax = 0;
for (Member3 m : memberVector) {
    m.AsReq = 0; m.As = 0;
    m.s = 0;
    m.fs = 0; m.fsc = 0;
    m.Mcap = 0;
}
Lambda = 0;
lo = 0; lo1 = 0; lo2 = 0;
i = 0;
k1 = 0; k2 = 0;
LambdaLim = 0;
n = 0;
Med = 0; Mmin = 0;
emin = 0; etot = 0; e0 = 0; e01 = 0; e02 = 0; ea = 0; e2 = 0;
K1 = 0; K2 = 0; K2cal = 0;
Nud = 0; Nbal = 0;
EpsilonS = 0;
fs = 0;
s = 0;
AsN = 0; AsM = 0; AsPro = 0;
Nobars = 0;
nud = 0; nudlim = 0;
FailedColumnsArray.clear();
}

```

A.3 Capacity Design Check

```
//Capacity Design
public ArrayList<Joint3> UnrestJoList = new ArrayList<>();
final double OF = 1.30 ;//Overstrength Factor
double MMF;//Moment Magnification Factor
public ArrayList<Member3> CapacityFailedColumns = new ArrayList<>();
public ArrayList<Member3> ShearFailedMembers = new ArrayList<>();
public void MomentCapacityCheckAtJoint() {
    for (Joint3 j : jointVector) {//Collect Unrestrained Joints
        if (!j.isRrx() && !j.isRry() && !j.isRrz() && !j.isRrmx() &&
            !j.isRrmy() && !j.isRrmz()) {
            UnrestJoList.add(j);
        }
    }
    for (Joint3 j : UnrestJoList) {
        double Mb1span = 0;//Beam1 span moment capacity
        double Mb1supp = 0;//Beam1 support moment capacity
        double Mb2span = 0;//Beam2 span moment capacity
        double Mb2supp = 0;//Beam2 support moment capacity
        double Mc1 = 0;//Column1 moment capacity
        double Mc2 = 0;//Column2 moment capacity
        double sumMc;
        double sumMb;
        for (Member3 k : memberVector) {
            if (j == k.start && k.Type == 0) {
                Mb1span = Math.abs(k.maxMz) * k.PRRSpan;
                Mb1supp = Math.abs(k.minMz) * k.PRRSupp;
            }
            else if (j == k.end && k.Type == 0) {
                Mb2span = Math.abs(k.maxMz) * k.PRRSpan;
                Mb2supp = Math.abs(k.minMz) * k.PRRSupp;
            }
            else {
            }
        }
        sumMb = Math.max(Mb1span, Mb2span) + Math.max(Mb1supp, Mb2supp);
        for (Member3 k : memberVector) {
            if (j == k.start && k.Type == 1) {
                Mc1 = Math.abs(k.Mcap);
            }
            else if (j == k.end && k.Type == 1) {
                Mc2 = Math.abs(k.Mcap);
            }
            else {
            }
        }
        sumMc = Mc1 + Mc2;
        if (sumMc >= OF*sumMb){
            MMF = 1; //No need to magnify column moments
        }
        else {
            MMF = OF*sumMb/sumMc;
        }
        for (Member3 k : memberVector) {
            if (j == k.end && k.Type == 1) {
                k.CMMF = MMF;
            }
            else {
            }
        }
    }
}
```

```

    }
}
}
for (Member3 m : memberVector) {
    if (m.Type == 1){
        if (m.CMMF > 1){
            CapacityFailedColumns.add(m);
        }
        else {
        }
    }
    else {
    }
}
}
public double Mbr1;
public double Mbr2;
public double Mcr1;
public double Mcr2;
public double GammaRd;
public void ComputeDesignShearFromMemberCapacity() {
    for (Member3 m : memberVector) {
        if (m.Type == 0){
            if (DC == 'M'){
                GammaRd = 1.00;
            }
            else if (DC == 'H'){
                GammaRd = 1.20;
            }
            else {
                GammaRd = 0;
            }
        }
        Joint3 [] MemEnds = new Joint3 [2];
        MemEnds[0] = m.start;
        MemEnds[1] = m.end;
        for (Joint3 j : MemEnds) {
            double Mb1span = 0; //Beam1 span moment capacity
            double Mb1supp = 0; //Beam1 support moment capacity
            double Mb2span = 0; //Beam2 span moment capacity
            double Mb2supp = 0; //Beam2 support moment capacity
            double Mc1 = 0; //Column1 moment capacity
            double Mc2 = 0; //Column2 moment capacity
            double sumMc;
            double sumMb;
            for (Member3 k : memberVector) {
                if (j == k.start && k.Type == 0) {
                    Mb1span = Math.abs(k.maxMz) * k.PRRSpan;
                    Mb1supp = Math.abs(k.minMz) * k.PRRSupp;
                }
                else if (j == k.end && k.Type == 0) {
                    Mb2span = Math.abs(k.maxMz) * k.PRRSpan;
                    Mb2supp = Math.abs(k.minMz) * k.PRRSupp;
                }
                else {
                }
            }
        }
        sumMb = Math.max(Mb1span, Mb2span) + Math.max(Mb1supp, Mb2supp);
        for (Member3 k : memberVector) {
            if (j == k.start && k.Type == 1) {

```

```

        Mc1 = Math.abs(k.Mcap);
    }
    else if (j == k.end && k.Type == 1) {
        Mc2 = Math.abs(k.Mcap);
    }
    else {
    }
}
sumMc = Mc1 + Mc2;
if (j == m.start){
    Mbr1 = GammaRd * Math.abs(m.minMz) * m.PRRSupp
        * Math.min(1, sumMc/sumMb);
}
else if (j == m.end){
    Mbr2 = GammaRd * Math.abs(m.minMz) * m.PRRSupp
        * Math.min(1, sumMc/sumMb);
}
else {
}
}
m.Vsdb = ((Mbr1+Mbr2)/m.length) + m.maxVy;
}
else if (m.Type == 1){
    if (DC == 'M'){
        GammaRd = 1.10;
    }
    else if (DC == 'H'){
        GammaRd = 1.30;
    }
    else {
        GammaRd = 0;
    }
    Joint3 [] MemEnds = new Joint3 [2];
    MemEnds[0] = m.start;
    MemEnds[1] = m.end;
    for (Joint3 j : MemEnds) {
        double Mb1span = 0; //Beam1 span moment capacity
        double Mb1supp = 0; //Beam1 support moment capacity
        double Mb2span = 0; //Beam2 span moment capacity
        double Mb2supp = 0; //Beam2 support moment capacity
        double Mc1 = 0; //Column1 moment capacity
        double Mc2 = 0; //Column2 moment capacity
        double sumMc;
        double sumMb;
        for (Member3 k : memberVector) {
            if (j == k.start && k.Type == 0) {
                Mb1span = Math.abs(k.maxMz) * k.PRRSpan;
                Mb1supp = Math.abs(k.minMz) * k.PRRSupp;
            }
            else if (j == k.end && k.Type == 0) {
                Mb2span = Math.abs(k.maxMz) * k.PRRSpan;
                Mb2supp = Math.abs(k.minMz) * k.PRRSupp;
            }
            else {
            }
        }
    }
    sumMb = Math.max(Mb1span, Mb2span) + Math.max(Mb1supp, Mb2supp);
    for (Member3 k : memberVector) {
        if (j == k.start && k.Type == 1) {

```

```

        Mc1 = Math.abs(k.Mcap);
    }
    else if (j == k.end && k.Type == 1) {
        Mc2 = Math.abs(k.Mcap);
    }
    else {
    }
}
sumMc = Mc1 + Mc2;
if (j == m.start){
    Mcr1 = GammaRd * Math.abs(m.Mcap)
        * Math.min(1, sumMb/sumMc);
}
else if (j == m.end){
    Mcr2 = GammaRd * Math.abs(m.Mcap)
        * Math.min(1, sumMb/sumMc);
}
else {
}
}
m.Vsdc = (Mcr1+Mcr2)/m.length;
}
else {
}
}
}
}

public void MemberDesignForShear(){
    ComputeDesignShearFromMemberCapacity();
    BeamDesignForShear();
    ColumnDesignForShear();
    ShearFailedMembers.clear();
    for (Member3 m : memberVector) {
        if (m.SSS == 0){
            ShearFailedMembers.add(m);
        }
        else {
        }
    }
}

public void ResetCapacityCheckResults(){
    UnrestJoList.clear();
    MMF = 0;
    for (Member3 m : memberVector) {
        m.CMMF = 1;
    }
    CapacityFailedColumns.clear();
}

public void ResetShearDesignResults(){
    AswPerSReq = 0; AswPerSmin = 0;
    Vrdmax = 0; theta = 0;
    for (Member3 m : memberVector) {
        m.SSS = 1;
        m.AswPerSb = 0; m.AswPerSc = 0;
        m.Vsdb = 0; m.Vsdc = 0;
    }
}
}

```

A.4 Main Program

```
//AnalyzeDesignCheckRepeat
public int counter1 = 1;
public int counter2 = 1;
public int totCounter = 0;
public void AnalyzeDesignCheckRepeat(){
    EarthquakeLoadComputationInputs();
    for (Member3 membl : memberVector) {
        for (Load3 ii : membl.modiLloads) {
            modifyLoad3(ii , Psi2i);
        }
    }
    do {
        do {
            ComputeMass();
            BaseShearForceComputation();
            EarthquakeLoadStoreyDistribution();
            AnalyzeMember();
            AnalyzeStructure();
            DesignMembers();
            if (FailedMembers.isEmpty()){
                if (DC == 'L'){
                    break;
                }
            }
            else {
                MomentCapacityCheckAtJoint();
                if (CapacityFailedColumns.isEmpty()){
                    counter1++;
                    break;
                }
            }
            else {
                ResetColumnDesignResults();
                ColumnDesign();
                if (FailedColumnsArray.isEmpty()){
                    counter1++;
                    break;
                }
            }
            else{
                ResetEQResults();
                ResetResults();
                ResetDesignResults();
                ResetCapacityCheckResults();

                for (Member3 m: FailedColumnsArray){
                    m.h += 0.025;
                    m.b += 0.025;
                }
                counter1++;
            }
        }
    }
    }
    else {
        ResetEQResults();
        ResetResults();
        ResetDesignResults();
        for (Member3 m : FailedMembers){
            if (m.Type == 0){
```

```

        m.h += 0.025;
    }
    else if (m.Type == 1){
        m.h += 0.025;
        m.b += 0.025;
    }
}
counter1++;
}
}while (!FailedMembers.isEmpty());
for (Member3 m: memberVector){
    if (m.Type == 0){
        ToDiscreteBeamLongReinf(m);
    }
    else {
    }
}
MemberDesignForShear();
if (ShearFailedMembers.isEmpty()){
    counter2++;
    break;
}
else {
    ResetEQResults();
    ResetResults();
    ResetDesignResults();
    ResetCapacityCheckResults();
    ResetShearDesignResults();
    for (Member3 m : ShearFailedMembers){
        if (m.Type == 0){
            m.h += 0.025;
        }
        else if (m.Type == 1){
            m.h += 0.025;
            m.b += 0.025;
        }
    }
    counter2++;
}
}while (!ShearFailedMembers.isEmpty());
for (Member3 m: memberVector){
    ToDiscreteShearReinf(m);
}
totCounter = counter1 + counter2;
}

```