

(MASS) TIMBER

STRUCTURALLY OPTIMIZED TIMBER BUILDINGS

KRISTEN STROBEL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF:

MASTER OF ARCHITECTURE

UNIVERSITY OF WASHINGTON

2016

COMMITTEE:

KATHRINA SIMONEN
ROBERT CORSER
JEFFREY BERMAN

PROGRAM AUTHORIZED TO OFFER DEGREE:

ARCHITECTURE

© COPYRIGHT 2016
KRISTEN STROBEL

UNIVERSITY OF WASHINGTON

ABSTRACT

(MASS) TIMBER: STRUCTURALLY OPTIMIZED TIMBER BUILDINGS

KRISTEN STROBEL

CHAIR OF THE SUPERVISORY COMMITTEE:
KATHRINA SIMONEN
DEPARTMENT OF ARCHITECTURE

This thesis explores the challenges and potential of mass timber as a paradigm shifting technology for the building industry through the application of parametric modeling technology to the design of office buildings. By testing building configurations in three zoning envelopes—low-rise suburban, mid-rise urban, and high-rise urban—optimization strategies for mass timber office buildings were developed. Facades and floor slabs were identified as the primary contributors to building cost and environmental impacts and therefore the easiest targets for optimization. The primary method for optimizing facades is replacing curtain wall with solid cross laminated timber (CLT) walls, this method runs counter to developer driven standards of fully glazed facades making short term adoption of this strategy unlikely without major shifts in building developer and owner expectations. Slabs and floor systems can be optimized through the implementation of novel solutions that take advantage of composite action between glulam, CLT, and concrete elements. Additionally, zoning height incentives could be used to make timber construction competitive with steel and concrete systems despite increased floor-to-floor heights. Finally, future research opportunities and needs, both architectural and technical, are identified.

TABLE OF CONTENTS

INTRODUCTION	12	TAMEDIA	66
PROBLEM STATEMENT	12	C13 BUILDING	68
PROJECT RATIONALE	13	NINA	70
THESIS OVERVIEW	14	ILLWERKE ZENTRUM MONTAFON VANDANS	72
REFERENCES	15	SKUTEVIKSODER 13	74
BACKGROUND RESEARCH	16	EARTH SCIENCES BUILDING	76
TYPE BASED ISSUES AND DRIVERS	16	MÜHLEBACHSTRASSE	78
TIMBER STRUCTURAL SYSTEMS	18	LCT ONE	80
PARAMETRIC OPTIMIZATION	20	ÉDIFICE FONDACTION QUÉBEC	82
LIFE CYCLE ASSESSMENT	20	SEISMIC CODE COMPARISON	84
CODE ISSUES	21	REFERENCES	87
REFERENCES	23	CUSTOM SCRIPT COMPONENTS	88
PARAMETRIC ALGORITHM	24	DESIGN EXAMPLES	152
SOFTWARE TOOLS	24		
PROCESS	26		
LIMITATIONS	34		
LESSONS LEARNED	35		
REFERENCES	36		
RESULTS	38		
GENERAL ANALYSIS	38		
ARCHITECTURAL TEST APPLICATIONS	46		
REFERENCES	51		
CONCLUSIONS	52		
CONCLUSIONS	52		
SUMMARY OF REVIEW COMMENTS	52		
FUTURE WORK	53		
CASE STUDIES	54		
TREET	56		
FRAMEWORK	58		
WOOD INNOVATION AND DESIGN CENTER	60		
STADTWERKE LÜBECK HQ	62		
WOODTEK HQ	64		

LIST OF FIGURES



INTRODUCTION

FIGURE 1. NEW OFFICE SPACE CONSTRUCTED IN THE NEXT DECADE	12
FIGURE 2. TIMBER CONSTRUCTION COULD REDUCE GLOBAL CARBON EMISSIONS BY 14 TO 31%	13

BACKGROUND RESEARCH

FIGURE 3. AMERICAN AND EUROPEAN OFFICE BUILDING FLOOR PLANS	17
FIGURE 4. GRAVITY SYSTEM ELEMENTS	19

PARAMETRIC ALGORITHM

FIGURE 5. STRUCTURAL MODEL VISUALIZED IN RHINO	24
FIGURE 6. GRASSHOPPER ALGORITHM	25
FIGURE 7. PARAMETRIC ALGORITHM FLOW CHART	26
FIGURE 8. ZONING ENVELOPES	27
FIGURE 9. GEOMETRIC PARAMETERS	28
FIGURE 10. LOADING AND DESIGN PARAMETERS	28
FIGURE 11. SLAB DESIGN CRITERIA	28
FIGURE 12. SLAB DESIGN PROCEDURE	29
FIGURE 13. COLUMN DESIGN CRITERIA	29
FIGURE 14. BEAM DESIGN PROCEDURE	30
FIGURE 15. COLUMN DESIGN PROCEDURE	30
FIGURE 16. SHEAR WALL DESIGN PROCEDURE	31
FIGURE 17. CONNECTION DESIGN PROCEDURE	32
FIGURE 18. BUILDING ASSEMBLIES	33
FIGURE 19. ASSEMBLY IMPACTS	33
FIGURE 20. EXISTING/FUTURE BEAM COLUMN LAYOUT	35

RESULTS

FIGURE 21. EXECUTION	39
FIGURE 22. AVERAGE COST AND ENVIRONMENTAL IMPACTS	39
FIGURE 23. COST AND ENVIRONMENTAL IMPACTS, MID-RISE ITERATION	40
FIGURE 24. SHEAR WALL ANALYSIS	40
FIGURE 25. VARYING SHEAR WALL THICKNESS	41
FIGURE 26. 6", 9", AND 12" WALL LAYOUTS, 100' TALL BUILDING	41
FIGURE 27. VARYING FACADE SHEAR WALL LENGTH	41
FIGURE 28. 10', 15', 20', AND 25' WALL LAYOUTS, 100' TALL BUILDING	42

FIGURE 29. COMPARISON OF STEEL AND TIMBER FLOOR SYSTEMS	43
FIGURE 30. COMPOSITE AND NON-COMPOSITE FLOOR SLABS	44
FIGURE 31. COMPOSITE AND NON-COMPOSITE BEAMS	45
FIGURE 32. ZONING HEIGHT INCREASES	45
FIGURE 33. CHARRING	46
FIGURE 34. LOW-RISE TEST BUILDING	47
FIGURE 35. MID-RISE TEST BUILDING	48
FIGURE 36. HIGH-RISE TEST BUILDING	49
FIGURE 37. HIGH-RISE FRAMING PLAN	50

CONCLUSIONS

CASE STUDIES

FIGURE 38. RENDERING OF TREET	56
FIGURE 39. STAIR DETAIL	57
FIGURE 40. EXTERIOR FACADE	57
FIGURE 41. BUILDING SECTION	57
FIGURE 42. PLAN	57
FIGURE 43. RENDERING OF FRAMEWORK	58
FIGURE 44. RENDERING OF FRAMEWORK	59
FIGURE 45. WOOD INNOVATION AND DESIGN CENTER	60
FIGURE 46. TYPICAL FLOOR CONSTRUCTION	61
FIGURE 47. TYPICAL FLOOR PLAN	61
FIGURE 48. LOBBY	61
FIGURE 49. STADTWERKE LÜBECK HEADQUARTERS	62
FIGURE 50. FLOOR PLAN	63
FIGURE 51. COURTYARD	63
FIGURE 52. VIEW OF WOODEN STRUCTURE	63
FIGURE 53. WOODTEK HEADQUARTERS	64
FIGURE 54. SECTION	65
FIGURE 55. COURTYARD	65
FIGURE 56. KITCHEN	65
FIGURE 57. ENTRY	65
FIGURE 58. ROOF	65
FIGURE 59. STAIR	65
FIGURE 60. TAMEDIA OFFICE BUILDING	66

FIGURE 61. VIEW OF STRUCTURE	67
FIGURE 62. SECTION AND PLAN	67
FIGURE 63. VIEW OF UPPER FLOOR	67
FIGURE 64. C13 BUILDING	68
FIGURE 65. C13 BUILDING REAR STAIR DETAIL	68
FIGURE 66. EXPOSED CLT IN FINISHED STRUCTURE	69
FIGURE 67. EXTERIOR VIEW	69
FIGURE 68. STRUCTURE DURING CONSTRUCTION	69
FIGURE 69. NINA HEADQUARTERS	70
FIGURE 70. OFFICE INTERIOR	71
FIGURE 71. EXTERIOR PANELLING DETAIL	71
FIGURE 72. PLAN	71
FIGURE 73. CAFETERIA	71
FIGURE 74. ILLWERKE ZENTRUM MONTAFON VANDANS	72
FIGURE 75. VIEW OF EXTERIOR	73
FIGURE 76. PLAN	73
FIGURE 77. OFFICE	73
FIGURE 78. LOBBY ATRIUM	73
FIGURE 79. SKUTEVIKSODER 13	74
FIGURE 80. SKYLIGHT DETAIL	75
FIGURE 81. CONSTRUCTION	75
FIGURE 82. CONSTRUCTION	75
FIGURE 83. PANEL ERECTION	75
FIGURE 84. EARTH SCIENCES BUILDING	76
FIGURE 85. SECTION	77
FIGURE 86. MAIN ATRIUM	77
FIGURE 87. VIEW OF EXTERIOR	77
FIGURE 88. MÜHLEBACHSTRASSE	78
FIGURE 89. STREET FACADE	79
FIGURE 90. SIDING DETAIL	79
FIGURE 91. SECTION	79
FIGURE 92. INTERIOR	79
FIGURE 93. PATIO DETAIL	79
FIGURE 94. LCT ONE	80
FIGURE 95. EXTERIOR ELEVATION	81
FIGURE 96. OFFICE	81
FIGURE 97. PLAN	81
FIGURE 98. CONCRETE-TIMBER COMPOSITE FLOOR DECK ERECTION	81
FIGURE 99. INTERIOR	81
FIGURE 100. ÉDIFICE FONDATION QUÉBEC	82
FIGURE 101. CONFERENCE ROOM	83
FIGURE 102. OFFICE CORRIDOR	83

SEISMIC CODE COMPARISON

FIGURE 103. ELASTIC DESIGN RESPONSE SPECTRA	84
FIGURE 104. SHEAR FORCE BY LEVEL	86

CUSTOM SCRIPT COMPONENTS

DESIGN EXAMPLES

TABLE OF TABLES



INTRODUCTION

BACKGROUND RESEARCH

TABLE 1. IBC FIRE RATING REQUIREMENTS 22

PARAMETRIC ALGORITHM

TABLE 2. CORE LAYOUT 27

TABLE 3. ASSEMBLY IMPACTS 34

RESULTS

TABLE 4. PARAMETRIC INPUTS 38

CONCLUSIONS

CASE STUDIES

SEISMIC CODE COMPARISON

TABLE 5. SEISMIC DESIGN CALCULATIONS 85

CUSTOM SCRIPT COMPONENTS

DESIGN EXAMPLES

INTRODUCTION

PROBLEM STATEMENT

As of the year 2012, there were over 16 billion square feet of office space in the United States accounting for approximately 18% of all commercial space. Of the commercial building stock 20% had been constructed in the past decade.^[1] Extrapolating from this data, there will be approximately four billion new square feet of office space constructed in the next decade. If built following the status quo, these buildings will contribute 2.3 billion metric tons of greenhouse gas emissions (CO₂e) to the environment due to construction and operations over the course of a 50-year lifespan.^[2]



Figure 1. New office space constructed in the next decade in the US will be enough to replace all office space in downtown Seattle nearly 100 times.^[3]

Greenhouse gas emissions are the primary contributor to climate change, with studies showing that “atmospheric concentrations are...unprecedented in at least the last 800,000 years”.^[4] Studies have shown that “substantial emissions reductions over the next few decades can reduce climate risks in the 21st century and beyond, increase prospects for effective adaptation, reduce the costs and challenges of mitigation in the longer term and contribute to climate-resilient pathways for sustainable development”.^[5] Seven percent of all energy globally is used by buildings in the United States; therefore, the construction of high efficiency buildings represents a significant opportunity for diverting carbon from the environment.^[6]

One way to reduce the carbon emissions associated with buildings is to use materials with lower carbon intensity, such as wood. Wood sequesters carbon from the environment and is renewable when harvested from sustainably managed forests. Research has shown that transitioning from steel and

concrete construction to timber construction could reduce global carbon emissions by 14 to 31 percent.^[7] Among many scenarios, optimized timber construction is the most promising using only 38% of available sustainable forest resources to offset steel and concrete construction. Currently there are significant research and construction efforts to realize tall mass-timber buildings. Examples in North America include the newly constructed Wood Innovation and Design Center in Prince George, BC,^[8] the Bullitt Center in Seattle, WA,^[9] as well as research proposals by leading architects and engineers including: Michael Green,^[10] SOM,^[11] and Mahlum.^[12]



While these projects represent a significant investment in bringing to market many years of research, there still exist significant opportunities to explore wood structures to better understand how they are different from more traditional construction methods and what structural forms lead to the greatest efficiencies.

Figure 2. Timber construction could reduce global carbon emissions by 14 to 31%

PROJECT RATIONALE

Studies project that nearly 90% of the US population will live in urban areas by the year 2050.^[13] In response to this influx of population, cities will need to grow increasingly dense. Towers—traditionally one of the most materially intensive forms of construction—rely on robust steel and concrete structural systems to support their height.^[14] Timber structures pose a promising alternative, supporting density at a lower carbon footprint.^[15] The majority of existing timber structures are located in Northern Europe. These buildings are primarily multi-family housing and due to the design and construction methods feature a large amount of structural redundancy. As timber construction is adopted in more locations, it is crucial that it becomes more optimized in order to reduce its costs and enable it to more effectively compete with steel and concrete structures.

Historically, building optimization has occurred through trial and error with each new building representing a step forward in engineering and architectural knowledge. Emerging digital technologies allow designers

to quickly test a multitude of configurations during design to allow each building to consider a greater range of design alternatives. Thereby advancing the state of practice more quickly.

This project tests both opportunities for mass timber construction and iterative parametric design to develop optimization guidelines for mass timber office buildings in Seattle.

THESIS OVERVIEW

A parametric algorithm was developed in Grasshopper for Rhino that allows for the rapid comparison of many different structural configurations. Given a series of user inputs, the algorithm generates 3-D geometry, sizes all pertinent structural members, and calculates output material quantities. Based on the execution of the algorithm for a range of inputs, general optimization guidelines for timber buildings are developed.

Results from the analysis show that the most effective optimization strategies are those that target material usage in the facade and the floor slabs. In analyzing test cases, these two assemblies form approximately 75% of the total impact of the building.

For the building facade, the primary challenge is balancing daylighting, thermal performance, and the layout of CLT shear walls. Analysis shows that moving away from the existing model for office building construction, a completely transparent facade, to a model where CLT shear walls are located along the facade has myriad benefits. Buildings with increased opacity to the facade cost less, can be designed to meet daylighting demands, have improved thermal performance, and free up the structural layout of shear walls. The true challenge to this model is not technical challenges but instead the willingness of building owners and developers to accept buildings that are not fully glazed.

The optimization of floor systems relies on two primary factors, decreasing the quantity of material in the slabs and reducing the overall depth of the floor system. Comparison of timber systems to steel systems found that at best timber systems have equal weight to steel and are twice as deep. Composite floor systems have a moderate effect, reducing slab depth and beam depth by about 20%. To allow timber to compete effectively with steel and concrete structures, technologies to reduce floor depth will need to be developed, or zoning incentives in the form of increased height will need to be provided for timber construction.

While this thesis provides an overview of key issues, it is also limited and leaves significant room for further research. Key aspects for future implementation include in depth studies of floor layout to improve system efficiency, refinement of considered load cases, continued architectural application of key findings, integration of thermal and daylighting analyses, further research into cost impacts of designing timber members for charring, and development of novel structural systems to improve floor system efficiency.

REFERENCES

- [1] US Energy Information Administration. (2014). *2012 Commercial Building Energy Consumption Survey* [Data file]. Retrieved from <http://www.eia.gov/consumption/commercial/data/2012/>
- [2] US Department of Energy. (2011). *Buildings Energy Data Book* [Data file]. Retrieved from: <http://buildingsdatabook.eren.doe.gov/default.aspx>
- [3] Metropolitan Improvement District and Downtown Seattle Association. (2015). *Downtown Office Space*. Seattle, WA: Downtown Seattle Association.
- [4] Intergovernmental Panel on Climate Change. (2014). *Climate Change 2014 Synthesis Report Summary for Policymakers*. (pp. 4) Switzerland: IPCC.
- [5] IPCC, 2014, pp. 17
- [6] US DOE, 2011, Chapter 1
- [7] Oliver, CD., Nassar, NT., Lippke, BR., & McCarter, JB. (2014). Carbon, Fossil Fuel, and Biodiversity Mitigation with Wood and Forests. *Journal of Sustainable Forestry*, 33(3), 248-275.
- [8] Michael Green Architects. (2015). *Wood Innovation and Design Center*. Retrieved from: <http://mg-architecture.ca/work/wood-innovation-design-center/>
- [9] Bullitt Foundation. (2013). *Bullitt Center*. Retrieved from: <http://www.bullittcenter.org/>
- [10] Michael Green. (2012). *The Case for Tall Wood Buildings*. British Columbia, Canada: MGB.
- [11] Skidmore, Owings, & Merrill. (2013). *Timber Tower Research Project*. Chicago, IL: SOM.
- [12] Mahlum. (2014). *CLT Feasibility Study: A Study of Alternative Construction Methods in the Pacific Northwest*. Seattle, WA: Mahlum.
- [13] UN Department of Economic and Social Affairs. (2014). *World Urbanization Prospects, the 2014 Revision*. Retrieved from: <http://esa.un.org/unpd/wup/Country-Profiles/>
- [14] SOM, 2013, pp. 2
- [15] SOM, 2013, pp. 40
- [16] US General Services Administration. (2013). *Federal Real Property Profile*. Retrieved from: <http://www.gsa.gov/portal/content/102880>
- [17] US General Services Administration. (2015). *Sustainable Design*. Retrieved from: <http://www.gsa.gov/portal/content/104462>

BACKGROUND RESEARCH

This thesis lies at the intersection of a number of established and emerging fields in architecture and structural engineering. An overview of key theoretical issues follows; starting with building type and precedents, then exploring structural issues, assessing developments in environmental performance, and finally providing an analysis of code issues and obstacles to implementation.

TYPE BASED ISSUES AND DRIVERS

More than any other building type, the office building is conventionally dominated by the plan. Its efficiency, its marketability, as well as the corporate culture of the client, unfold primarily in plan. This is not to say that the section has no role in the design of the office building; but in the end, it is the plan multiplied vertically, that betrays its true workings. The corner office, the cellular offices, the open plan: these spatial divisions may evolve with lightweight partitions, but they are anchored by the permanent positioning of a building's infrastructures. In that sense, the ultimate limitations and opportunities of an office building manifest in the disposition of these services in plan.^[1]

The plan of a modern office building is carefully orchestrated in response to a series of rigorous requirements. Commercial building developers aim to maximize leasable floor space and future adaptability while minimizing construction costs.

The maximum allowable floor area for a given site is dictated by zoning requirements, usually through an allowable floor area ratio (FAR). FAR dictates the maximum allowable space that can be constructed on any site as a function of the total square footage of the site. This in addition to height limitations along with other zoning prescriptions dictate the maximum envelope a building can fill on any given site.

Once the overall envelope is determined, architects test layouts against interior space requirements. The efficiency of interior space for an office building is driven by lease depth (typically 40 to 50 feet in the US, 20 to 25 feet in the EU)^[2] as well as the ability of the structure to accommodate the module of furnishings (typically five feet in the US).^[2] Secondary criteria for the suitability of office space include access to daylight and views and,

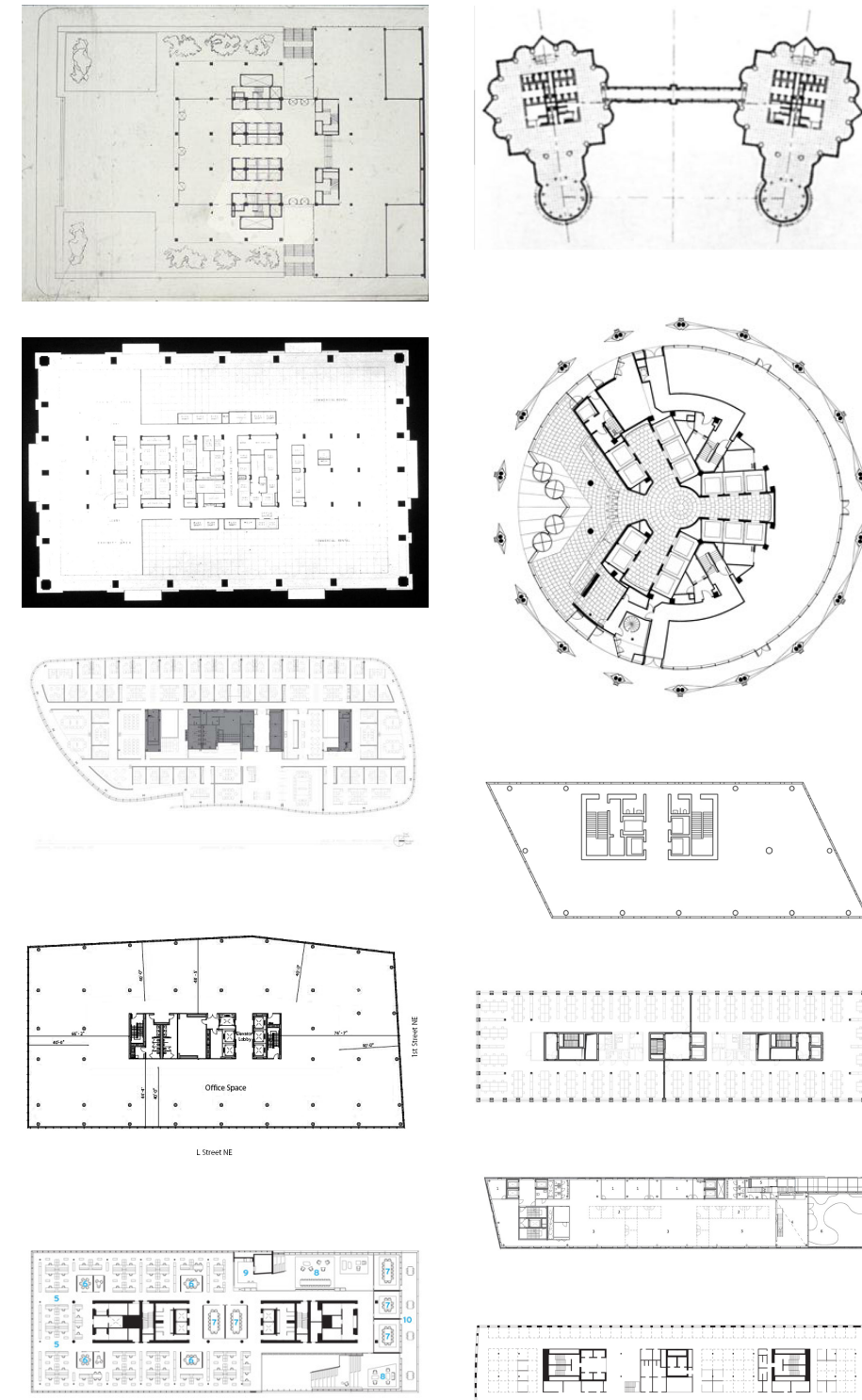


Figure 3. American and European Office Building Floor Plans

Left Column (from Top):

Seagram Building (1958) (from artstor.org)

John Hancock Center (1969) (from artstor.org)

Centennial Tower (2009) (from archrecord.construction.com)

1100 First Street (2009) (from archrecord.construction.com)

Building 335 (2012) (from archrecord.construction.com)

Right Column (from Top):

Petronas Towers (1996) (from pcparch.com)

30 St Mary Axe (2004) (from fosterandpartners.com)

Turninn (2010) (from PK Arkitektur)

Herostrasse 12 (2014) (from Max Dudler Architects)

DNB C-Building (2012) (from Dark Arkitektur)

The Carve (2014) (from A-Lab)

for buildings with operable windows, natural ventilation. The standard for Class A office space in the US is a fully glazed facade allowing views and daylight from all sides. Associated with all of these variables is the location of the core.

The location of the core must meet requirements for fire, egress, and lateral force resistance but also allow for flexibility in the subdivision of floors to accommodate multiple office layouts. Additionally, designs must accommodate changes to tenants and working environments over time in response to new technologies. These requirements in the United States often lead to rectangular office buildings with a central core as the market standard.^[2]

Other key elements affecting the efficiency and economic viability of office buildings include: column spacing and floor-to-floor heights. Column spacing and floor-to-floor height are opposing criteria, with optimal solutions balancing the demands of each. Longer spans (wider column spacing) is valued as it allows a greater degree of flexibility for the layout of furnishings and spaces within the building. However, longer spans increase the depth of the building floor system thus increasing floor-to-floor heights. As floor-to-floor heights become increasingly large, fewer floors fit within the zoning envelope thus decreasing the overall economic viability of a project. The optimal relationship between these two factors for steel and concrete buildings is typically a column spacing of 30 to 40 feet.

TIMBER STRUCTURAL SYSTEMS

While timber buildings pre-date the development of modern structural types methods of building with wood have largely been forgotten in favor of steel and concrete construction. Due to concerns about the environmental impact of building out of steel and concrete coupled with advances in timber material and manufacturing technologies, there is a renewed interest in exploring how new wood technologies can fit into the modern building ecosystem. There have been a number of tall wood buildings constructed over the past decade as well as research efforts intended to move wood construction forward.

Two primary reports target the development of tall timber structures for North America. The first research report developed by Green^[3] proposed a mass timber steel composite system for a hypothetical residential building in Vancouver BC. The proposed structure was analyzed for its ability to withstand gravity, wind, and seismic forces at a schematic level. Additionally, the proposal treated issues associated with codes, systems, and architectural viability. The proposed structure was tested for its application to 12, 20, and 30 story residential buildings. Key structural issues identified include the need for more intensive structural testing and modeling to ensure that the proposed system behaves as designed.

The second study, by SOM^[4], outlined the conceptual design of a mass timber concrete hybrid structural system for a 42-story residential tower in

Chicago. The building has a centrally located core and a 9'-0" floor to floor dimension. The structural system for the proposed building consists of timber floors, columns, and shear walls. Connections are made between structural elements by steel reinforcing epoxy bonded into the wood and cast into concrete joints. Overall the building is 70% timber and 30% concrete by volume. Key issues identified for the proposed structural system include: uplift in the foundation caused by the lightness of the structure, as well as the need to test the proposed connection type to ensure it performs as designed.

Appendix A contains key information and photos of mid- to high-rise case study projects. These case studies demonstrate the current state of practice for timber design and construction. The case study projects are primarily located in Europe and Canada. Of note is the high degree of redundancy and material inefficient inherent in the panelized construction of many of these buildings. Also significant is that the majority of these examples are built in locations without significant seismic risks.

From these case studies, two primary approaches to gravity systems for timber buildings emerge. The first is a panel-based system where the building has load bearing CLT walls that support CLT floor panels. This construction method is suited to buildings with a large number of interior partitions, typically housing. The second approach is a frame approach where glulam beams and columns support CLT floor slabs. This thesis will implement the second approach, a sketch of primary elements is shown in Figure 4.

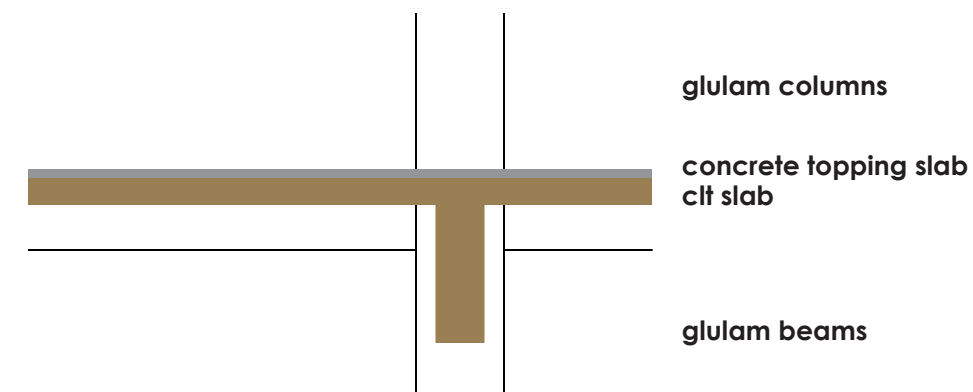


Figure 4. Gravity System Elements

A significant challenge to tall mass timber buildings in high seismic regions is the design of CLT shear walls. A number of researchers have developed systems to enhance the performance of CLT shear wall systems. Dolan et al^[5] propose the introduction of viscous dampers at multiple levels in a 10-story CLT building to reduce story shears and displacements, Buchanan et al^[6] propose post-tensioned timber connections for seismic resistance, and Ganey^[7] studied the design of rocking CLT walls. While these emerging technologies represent a significant opportunity for higher performing, more materially efficient systems, they all require significant research and testing before they are incorporated as code accepted timber seismic systems. This thesis will implement CLT shear walls using standard connections and tie downs, which will be included in the next cycle of structural design codes^[8], to understand the potential for wide spread near-term implementation of fully timber buildings in seismic zones.

PARAMETRIC OPTIMIZATION

Parametric optimization is a general category describing a series of methods for finding optimal solutions to problems with multiple input variables. At the most rudimentary level, a brute force method can be used to optimize a problem. In a brute force method, the problem is executed over all possible combinations of parameters within the solution space to provide a full set of possible solutions. By solving a problem over the entire solution space, optimum solutions and the effects of different input variables can be understood. While a number of more intelligent algorithms exist^[9], that allow multi-variable problems to be optimized by selectively seeding solutions and iterating towards the optimum thus reducing computational time, this thesis will use a brute force method. The ability to understand when and why the models were under-performing was one reason for choosing a brute force method. Additionally, in this instance the time to iterate through all possible solutions of interest was not prohibitively computationally expensive.

LIFE CYCLE ASSESSMENT

Life cycle assessment (LCA) is a method for accounting the environmental impacts of a material or process.^[10] To calculate the environmental impact of a building over its life span this typically requires accounting for raw material extraction, manufacturing, transportation, construction, operational energy, repairs and replacement, demolition, and disposal. There has been a recent push—led by the inclusion of LCA in LEED v4^[11]—to perform LCA during building design in order to better understand the environmental impacts of construction.

For this thesis, equivalent global warming potential will be used as one measure of the efficiency of building alternatives. Using LCA to measure buildings' material efficiency helps to further test whole building LCA providing benchmark data for timber buildings. A brief overview of the LCA methodology for this project follows.^[12]

GOAL

This LCA is a study of the impact of the shell and structure of mass timber buildings, developed to understand the differences in performance of parametrically iterated designs. The LCA is intended for structural engineers and architects, to serve as a baseline for the carbon intensity of mass timber buildings.

SCOPE

For each building iteration, the LCA includes impacts from glulam beams and columns (including steel connections), floor assemblies, roof assemblies, core wall assemblies, and facade wall assemblies. The analysis does not include foundations, finishes, mechanical, electrical, and plumbing equipment, material repair and replacement during the life cycle of the building, or operational energy. The impacts of materials due to production, construction, and disposal at end of life are included while credits from carbon sequestration and recycling are excluded. If credits from carbon sequestration and recycling are included, the impact of timber products

can be negative (carbon sequestered). Since the goal of this analysis is to minimize material usage, including timber as a material that increases the impact of the building rather than decreases it aligns with this goal.

In order to simplify the analysis, only GWP was used to assess the environmental impact of the buildings. Analyzing other impact categories (e.g. acidification potential, eutrophication, etc.) might guide different conclusions about minimizing the environmental impacts of buildings.

The LCA work done for this thesis has not been critically reviewed and therefore should only be used for increased understanding of the impacts of timber construction not to make definitive conclusions.

INVENTORY

Data extracted from Athena Impact Estimator^[13] are used to calculate the impacts of each of the assemblies. Structural material quantities are estimated using the structural sizing algorithm. Material quantities for facades are estimated based on the area of the facades using Athena assembly data for the quantity of glass and aluminum per square foot of curtain wall.

For further detail about materials included in each assembly and their impacts see Chapter 03. For discussion of results and conclusions see Chapter 04.

CODE ISSUES

While the technology exists for wide scale implementation of high efficiency tall timber buildings, there are still significant obstacles from a regulatory and code standpoint. Chief among these are concerns related to the fire and seismic performance of timber.

Although flammable, mass timber is naturally fire resistant once a charred layer has formed, allowing structural members to continue carrying load in a fire. Some European codes allow the calculation of this effect as part of a strategy for designing mass timber buildings that meet life safety objectives. Through the use of formulas that calculate the charring rate of timber and the required fire rating, structural members are oversized to ensure sufficient load capacity to prevent collapse in the event of a fire.

American codes rely on a height limitation to govern the use of wood in buildings, Table 1 outlines fire ratings and height limits for the heights of buildings considered for this project.^[14] This is largely due to concerns about the flammable nature of wood and a lack of incorporation of current scientific understanding of the performance of wood members during a fire. A performance based approach that demonstrates code equivalent performance carried out in consultation with the local jurisdiction is likely the best approach for overcoming current code restrictions.

Table 1. IBC Fire Rating Requirements

Construction Type	Structural Materials	Maximum Height	Required Primary Frame Fire Rating
Type 1A	Fire Resistive Non-Combustible	UL	3 hr
Type 1B	Fire Resistive Non-Combustible	160 ft	2 hr
Type 4	Heavy Timber	65 ft	1 hr

For this thesis, the effect of including a one hour fire rating for timber members is examined. Both the International Building Code (IBC)^[14] and National Design Specification (NDS)^[15] include equations for calculating charring depths and resulting member capacities. For buildings above 65-feet in height this is insufficient as the sole fire design approach. As such additional fire protection methods such as fireproofing structural assemblies with gypsum wallboard will be necessary for meeting the code-required minimum fire ratings.

The lateral system for tall wood buildings is the other primary obstacle to implementation in seismic zones. Only a few buildings have been designed in seismic regions (Cenni di Cambiamento (Italy) and Framework (Portland) see Appendix A) both required significant testing work to validate the design and performance of lateral force systems. While there is a significant amount of research around the development of novel systems for the use of CLT as a lateral force resisting system, American structural design provisions as defined by ASCE 7^[16] require all wood systems to meet performance based design criteria with analytical calculations and test data to be submitted to the authority having jurisdiction for approval.

REFERENCES

- [1] Picon, A. (2013). A-Typical Plan: Projects and Essays on Identity, Flexibility and Atmosphere in the Office Building. Kuo, J. (Ed.). (pp. 21) Zurich, Switzerland: Park Books.
- [2] Kohn, A., & Katz, P. (2002). Building Type Basics for Office Buildings. Kliment, S. (Ed.). New York: J. Wiley & Sons.
- [3] Green (2012)
- [4] SOM (2013)
- [5] Dolan, J.D., Bordry, V., Pei, S., & Van de Lindt, J. (2014). Tall Cross Laminated Timber Building: Design and Performance. *Structures Congress 2014*.
- [6] Buchanan, A., Palermo, A., Carradine, D., & Pampanin, S. (2011). Post-tensioned timber frame buildings. *The Structural Engineer*. 89(17). 24-30.
- [7] Ganey, R. (2015). *Seismic Design and Testing of Rocking Cross Laminated Timber Walls*. Master's thesis, University of Washington, Seattle, Washington.
- [8] American Society of Civil Engineers. (2016). Minimum Design Loads for Buildings and Other Structures. Reston, VA.
- [9] Rutten, D. (2010). Evolutionary Principles Applied to Problem Solving. Retrieved from: <http://www.grasshopper3d.com/profiles/blogs/evolutionary-principles>
- [10] International Organization for Standardization. (2006). Environmental Management - Life Cycle Assessment - Principles and Framework. Geneva, Switzerland.
- [11] US Green Building Council. (2015). LEED. Retrieved from: <http://www.usgbc.org/leed>
- [12] Simonen, K. (2014). Life Cycle Assessment. Smith, RE. (Ed.). New York: Routledge.
- [13] Athena Sustainable Materials Institute. (2015). Athena Impact Estimator for Buildings 5.1.01 [Computer software]. Ottawa, Canada: Athena Institute.
- [14] International Code Council. (2015). International Building Code. Country Club Hills, IL: International Code Council.
- [15] American Wood Council. (2015). National Design Specification for Wood Construction. Leesburg, VA.
- [16] American Society of Civil Engineers. (2010). Minimum Design Loads for Buildings and Other Structures. Reston, VA.

PARAMETRIC ALGORITHM

As the primary method for understanding the material implications of basic architectural decisions (floor-to-floor height, column spacing) a parametric algorithm was developed within Grasshopper for Rhinoceros 5. The algorithm generates building geometry based on input parameters and sizes all pertinent structural elements. The various iterations are evaluated with the goal of minimizing material usage, with material quantities calculated based on the structural design and used to generate cost and environmental impacts for each iteration.

The choice to build a customized algorithm on top of existing software tools rather than using more established timber design programs was driven by the desire to be able to iterate geometry and add members as the building structure changed as well as to have full control and understanding of design assumptions. Apart from the generation of geometry and checks of members, the algorithm uses existing software tools.

SOFTWARE TOOLS

A number of existing software tools and plug-ins for Grasshopper were combined with sections of custom code to generate and analyze each iteration of the structure. A brief overview of tools used and their role in the algorithm follows.

RHINOCEROS 5

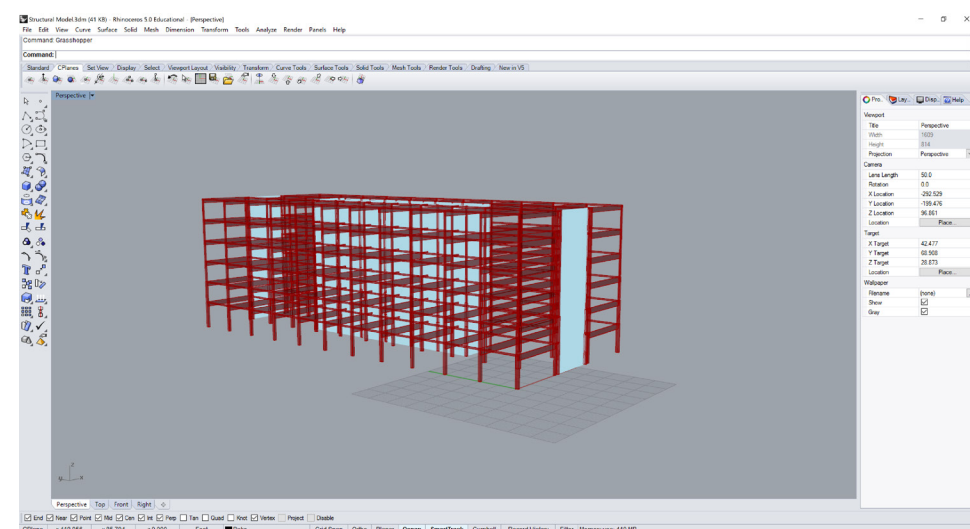


Figure 5. Structural Model Visualized in Rhino

Rhinoceros 5^[1] (Rhino) is a 3D-CAD environment. For the purposes of this analysis, Rhino served primarily as a visualization environment where input data and code could be graphically checked to ensure accuracy of the algorithm. The iteration of the Grasshopper algorithm was automated by a Python script in Rhino allowing iteration of the Grasshopper algorithm without manual intervention.

GRASSHOPPER

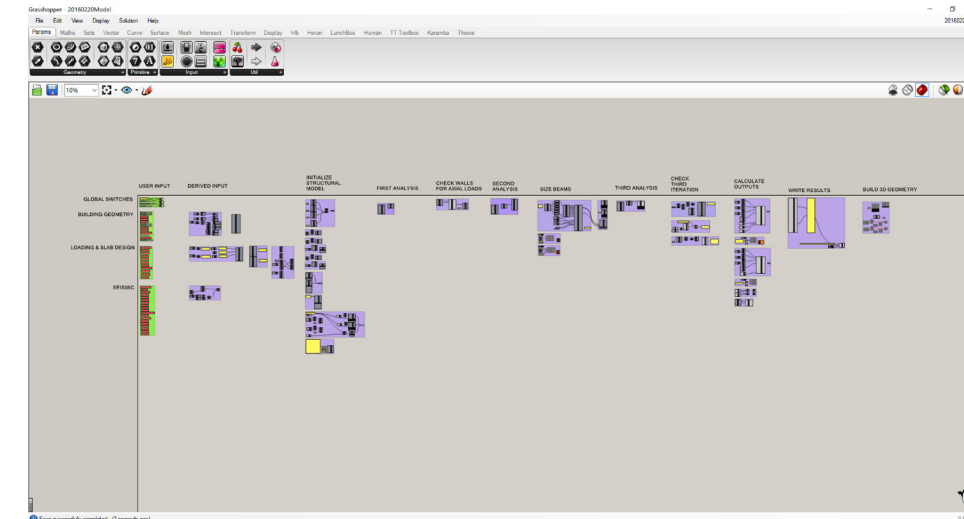


Figure 6. Grasshopper Algorithm

Grasshopper^[2] is a graphical algorithm editor for Rhino. Grasshopper contains a suite of components that allows geometry to be created and manipulated in Rhino. Grasshopper also supports custom C# components that allow the user to program specialized functions into Grasshopper. User input, generation of geometry, and calculation of outputs were all handled in Grasshopper using a mixture of stock components and custom C# scripts. For a full overview of custom script components, see Appendix C. Data from the analysis is automatically written to a text file at the end of each iteration to record both the inputs and corresponding results.

KARAMBA

Karamba^[3] is a finite element analysis (FEA) plug-in for Grasshopper. Based on the input of geometry and other parameters that define the structure (cross sections, materials, fixities, loads) Karamba calculates forces and deflections for the structure. Demands from Karamba were used to design structural members.

LUNCHBOX

LunchBox^[4] is a software tool that enables the transfer of data between Grasshopper and Microsoft Excel. Static data tables with cross section sizes and properties for CLT slabs and walls and glulam beams and columns and cost and environmental impact data were imported into grasshopper and subsequently internalized into the grasshopper script. This allows data tables to be updated as required before the iteration is performed and subsequently kept static over the course of the analysis. Since the transfer of data between grasshopper and excel is time intensive, minimizing the transfer of data between the two programs during analysis was key to minimizing total analysis time.

PROCESS

The primary processes in the parametric algorithm are shown in Figure 7. The following sections provide a detailed description of each of the sections and their implementation. Worked structural design examples for each of the elements are shown in Appendix D. A combination of building design codes and guidelines were used as appropriate: IBC^[5], ASCE 7^[6], NDS^[7], the CLT Handbook^[8], and ACI 318^[9].

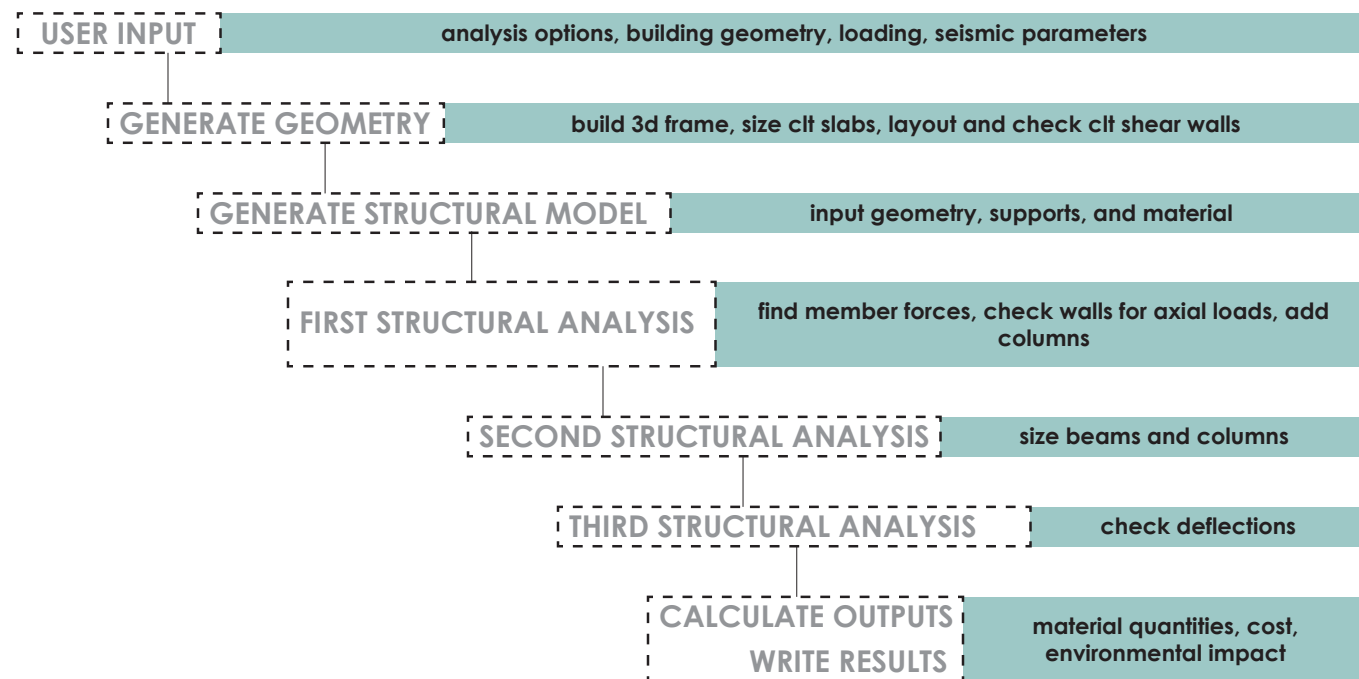


Figure 7. Parametric Algorithm Flow Chart

INPUTS

The algorithm generates analysis based on a number of input parameters that automatically initialize different geometric variations and structural design options. A list of all user-input values follows:

- Design Options
 - Include 60-minute fire rating for beams, columns, and slabs
 - Include Timber-Concrete Composite Slabs
 - Include Composite Beam-Slab

- Geometry Inputs
 - Bay Length (X and Y)
 - Maximum Building Dimensions (X and Y)
 - Maximum Building Height
 - First Floor Height
 - Typical Floor to Floor Height
 - Core Dimensions (X and Y)
 - Number of Interior Core Walls
 - Length of Facade Wall Segments
 - CLT Shear Wall Thickness
 - Concrete Topping Slab Thickness
- Loading Inputs
 - Dead Loads (Finishes, MEP)
 - Live Loads (Occupancy, Partitions)
 - Seismic (Importance Factor (I_e), Ground Accelerations (S_{DS} , S_{D1}), Response Modification Coefficient (R), Deflection Amplification Factor (C_d), Building Period Coefficient (C_t), Distribution Exponent (k))

Results are based on analysis for three primary scenarios: low-rise suburban, mid-rise urban, and high-rise urban.

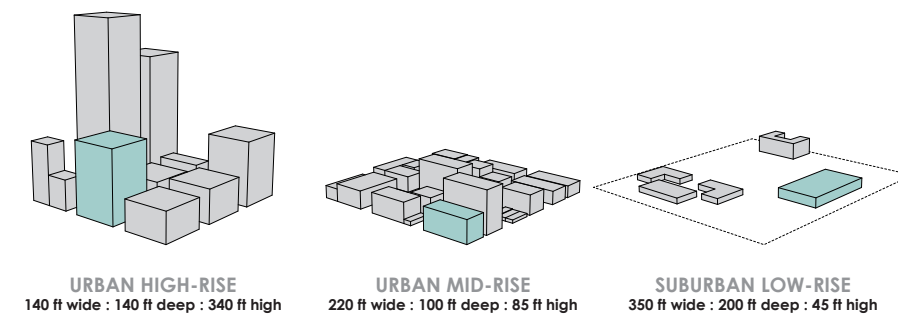


Figure 8. Zoning Envelopes

Based on the input zoning envelope and an input core layout varying geometry is generated based off the parameters shown in Figure 9. Core layouts were specified using rules of thumb for core area as a percent of floor area by building height. For each of the layouts, five interior core walls were included in the x-direction to account for walls between stairs, restrooms, and elevators in the core. The floor plan and section are laid out such that actual overall dimensions vary based on the number of grids and floors that divide evenly into the maximum dimensions.

Zoning Envelope	% Floor Area	X-Dimension	Y-Dimension	# Interior Walls
Low-Rise	5%	20'	35'	5
Mid-Rise	10%	20'	110'	5
High-Rise	20%	40'	80'	5

Table 2. Core Layout

At the element level, a number of parameters direct the structural design of each of the elements in the building. Figure 10 shows primary criteria for each of the structural elements.

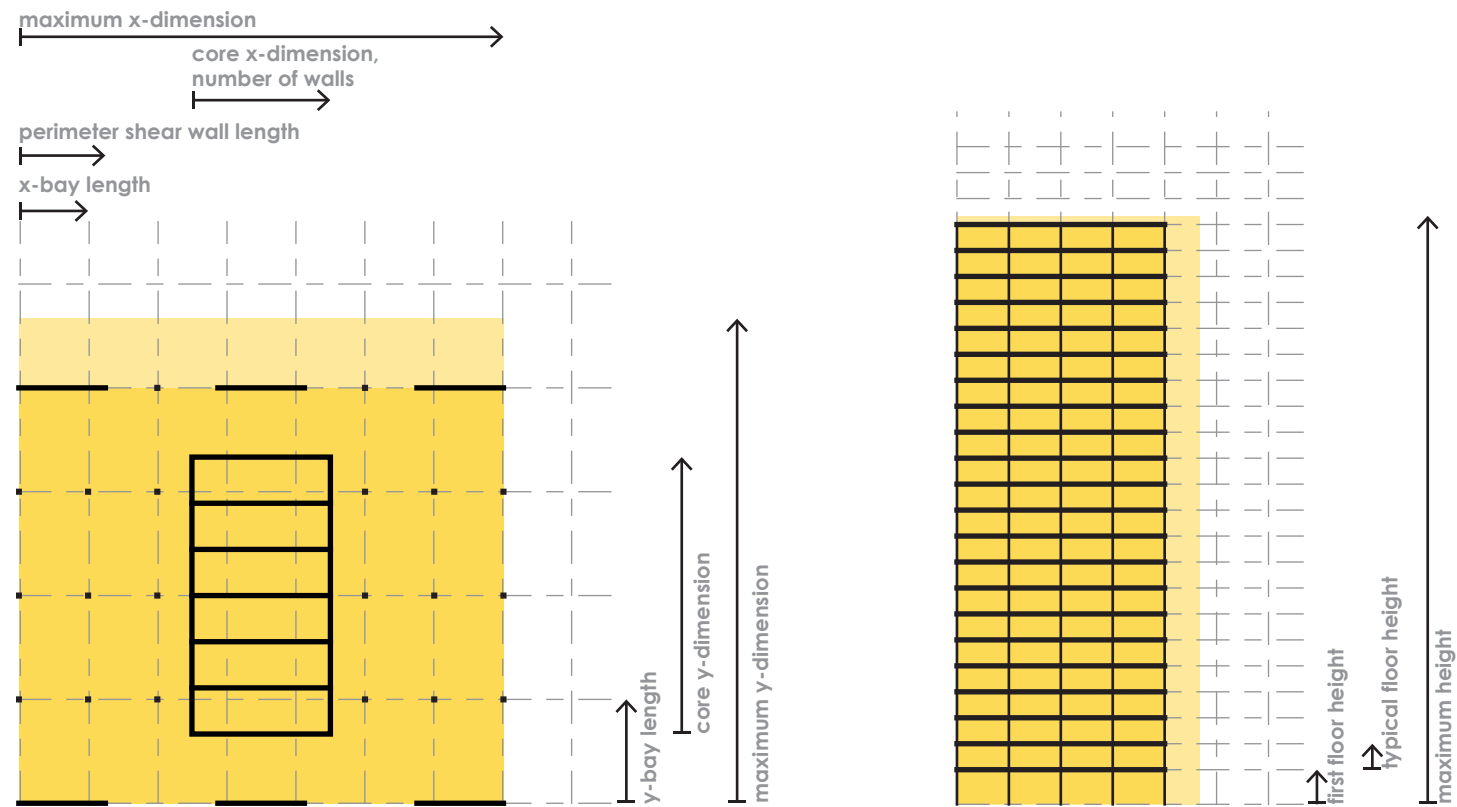


Figure 9. Geometric Parameters

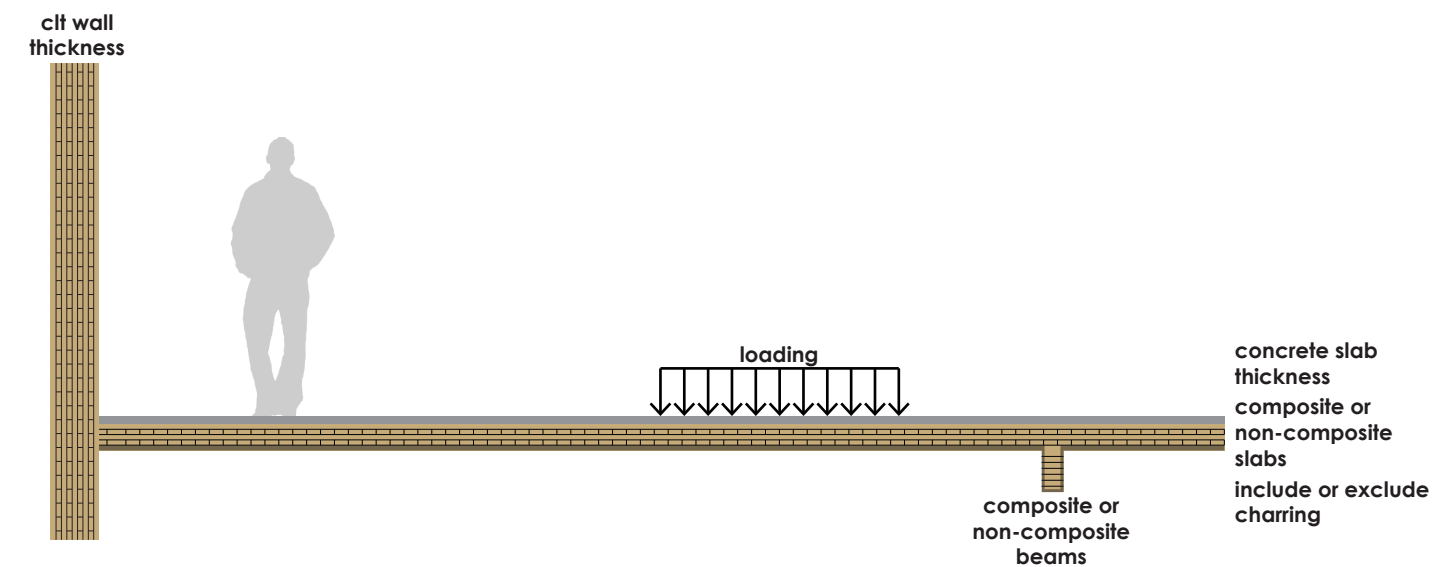


Figure 10. Loading and Design Parameters

SLABS

Based on loading, concrete topping slab thickness, and span length, the thickness of CLT slabs was calculated. Two distinct, but similar design procedures were used for the design of composite and non-composite slabs as shown in Figure 12. In both cases slabs were designed for strengths, deflections, vibrations, and a 60 minute fire rating.

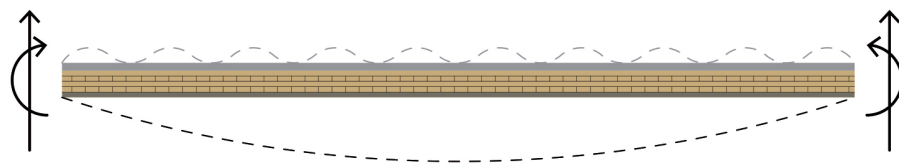
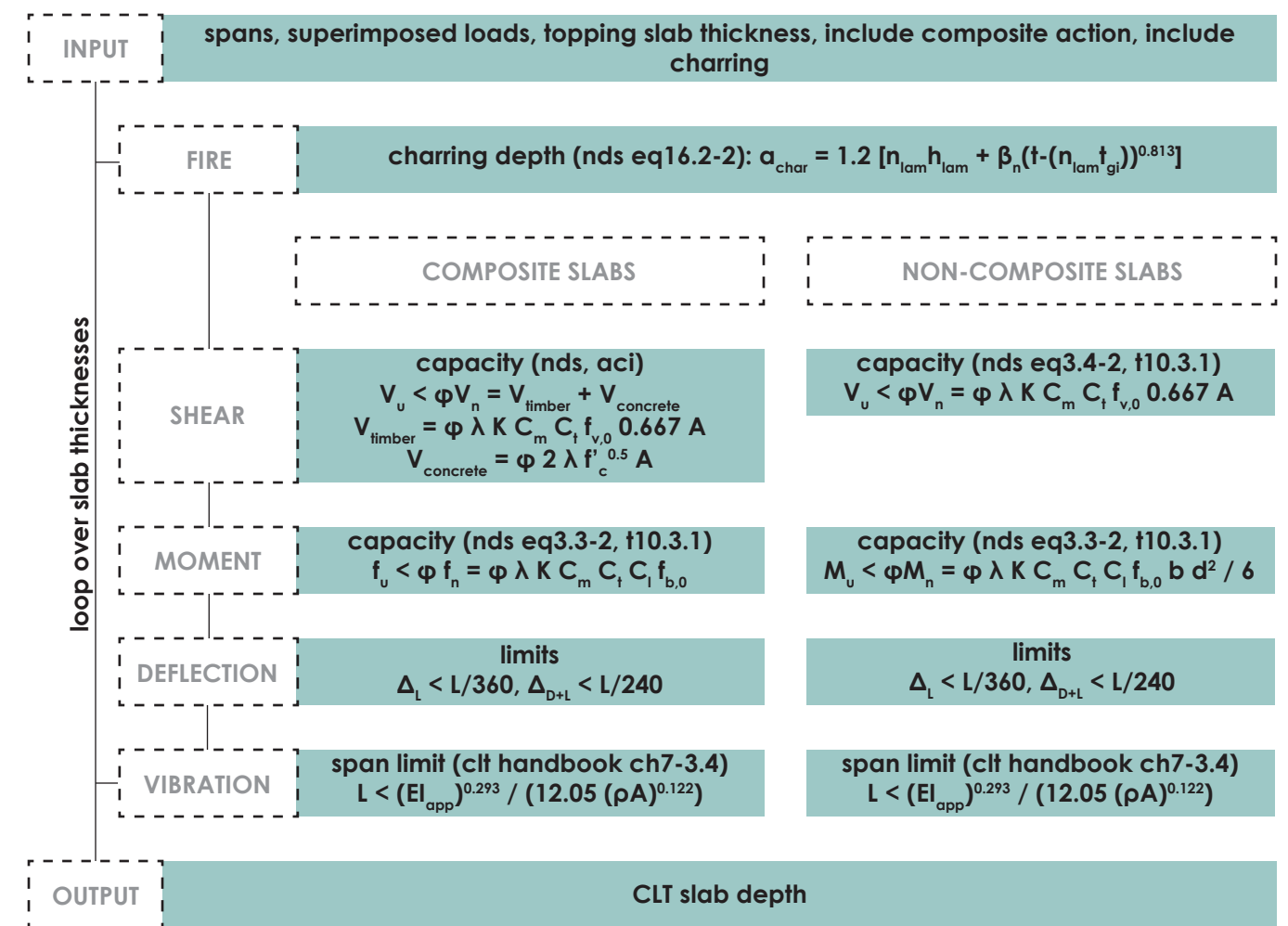


Figure 11. Slab Design Criteria Strengths, Deflections, Vibrations, and Charring



For composite slabs, transformed section properties and densities were used to increase the strength and stiffness for deflection and vibration design. This procedure assumes a connection between the CLT and concrete can develop the full demands.

Figure 12. Slab Design Procedure

Where fire is considered in the design of the timber slabs, the reduced section is used to check the slab for shear and moment capacity while the full section is used for deflection and vibration design. If slabs are not designed for charring, slabs are assumed to be fireproofed.

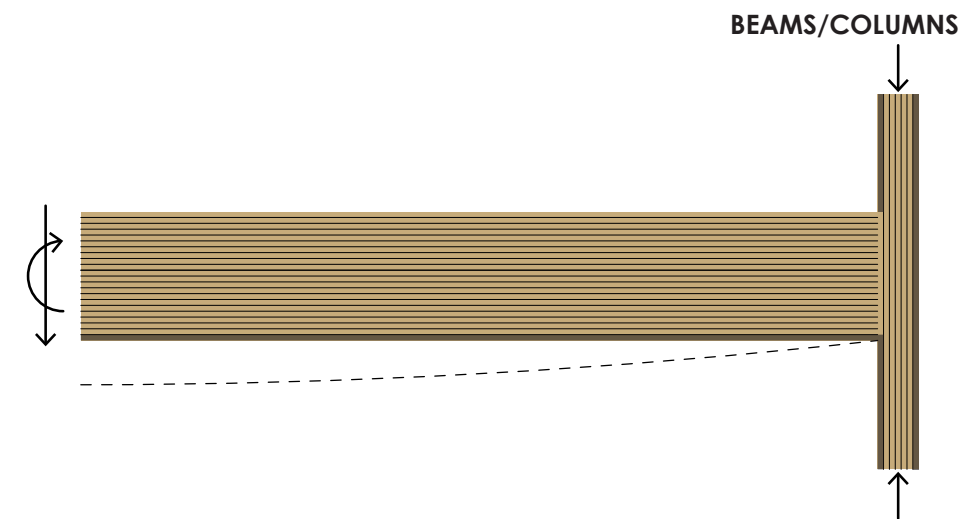


Figure 13. Column Design Criteria Strengths, Deflections, and Charring

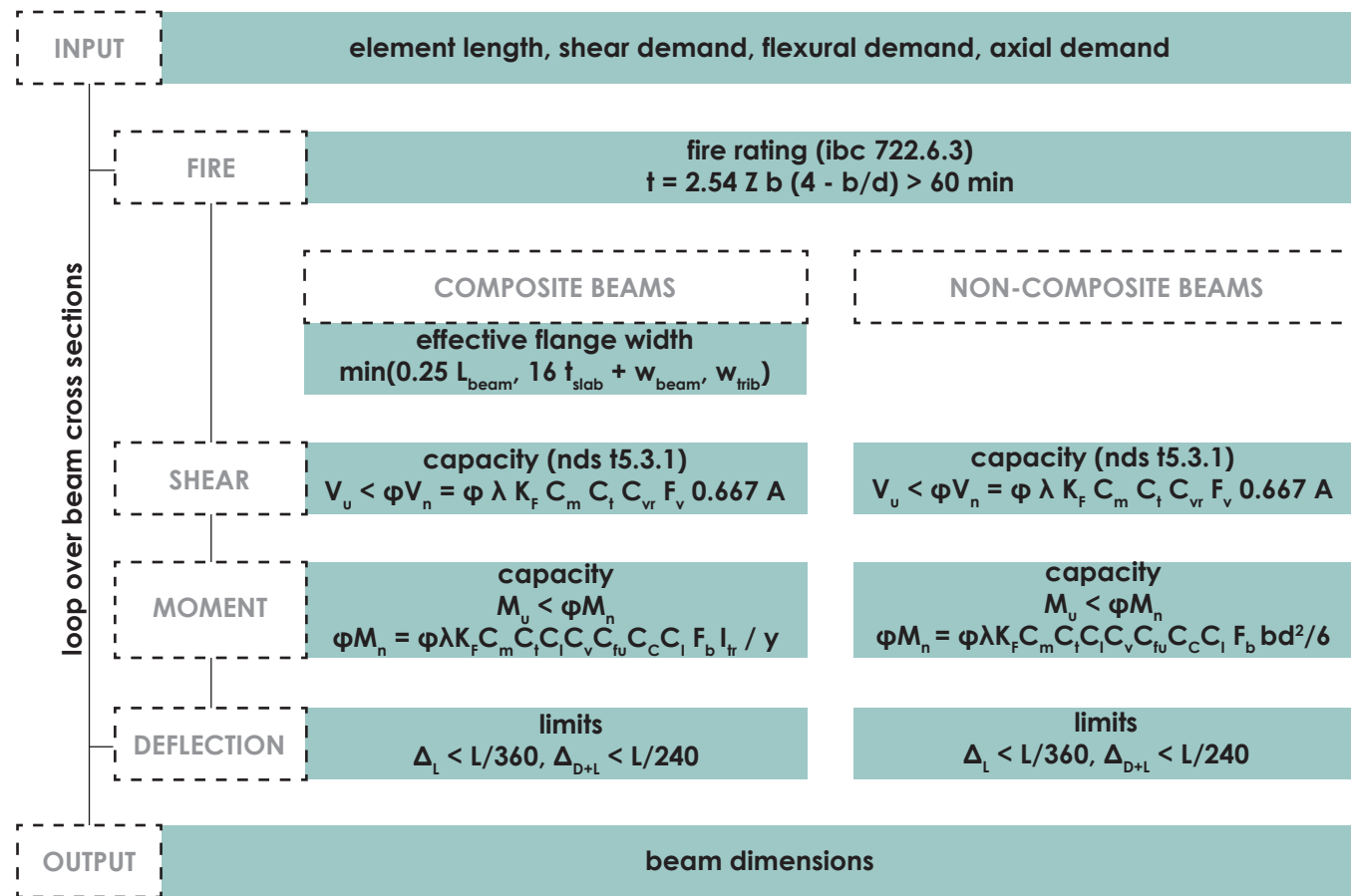


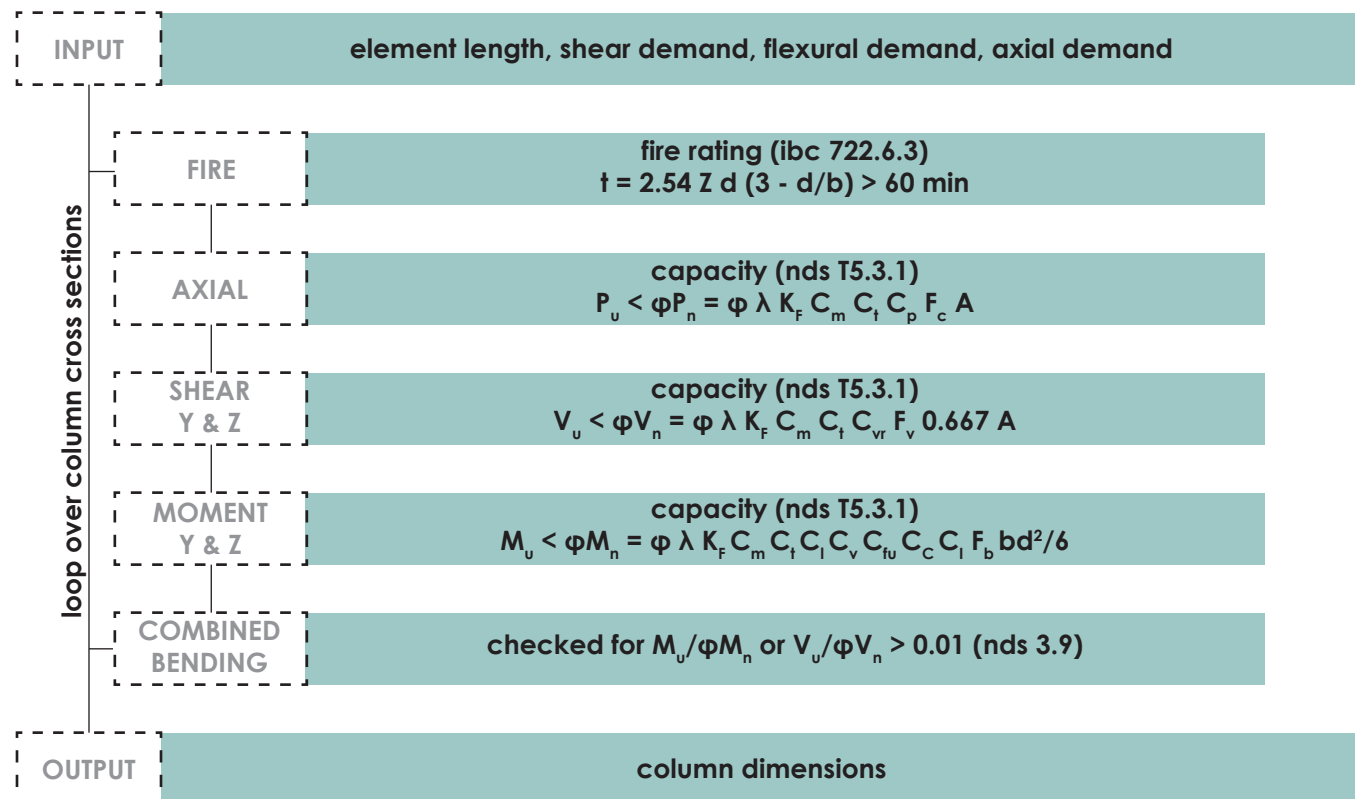
Figure 14. Beam Design

Procedure

From inputs of element demands and lengths, beam and column cross sections are selected for strength, deflections, and a 60 minute fire rating.

Figure 15. Column Design

Procedure



Beams are designed to either act compositely with CLT slabs or non-compositely. For composite beams, an effective flange width and corresponding effective section are used in checking the strength and deflections of the members.

For both beams and columns, fire design criteria from the IBC to determine the fire rating of members. Fire calculations use the demand-capacity ratio of the member, a measure of how close to its ultimate load carrying capacity a member is, to calculate a corresponding time rating for each member. For this analysis, beam and column dimensions were increased until members meet a 60-minute fire rating in addition to passing checks for strengths and deflections.

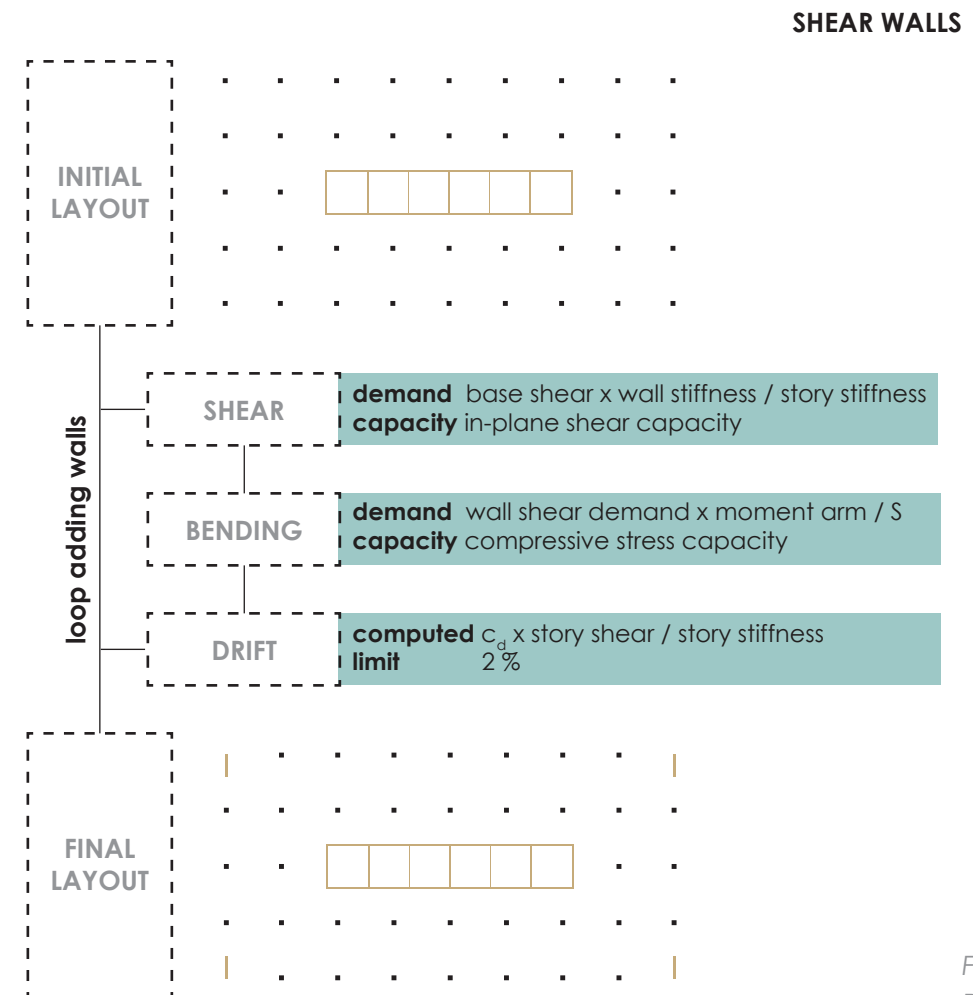


Figure 16. Shear Wall Design Procedure

Shear walls were designed to resist seismic and gravity loads. Fire was not considered for shear walls as they are assumed to be fireproofed with gypsum wallboard to achieve increased fire ratings at the core for fire and life safety. A simplified design procedure was developed that accounts for primary limit states; however, as limited design guidance exists, this procedure is likely limited and will benefit from further testing and development of consensus driven design guidelines. A number of assumptions were made to simplify the shear wall design procedure. Among these, rigid diaphragms were assumed (lateral forces distribute to walls by stiffness) and shear walls were assumed to act independently in each direction.

An overview of the shear wall design procedure is shown in Figure 16, for a more complete example see Appendix D. The algorithm starts by checking the core walls for shear, bending stress, and drifts. If the core walls are adequate for resisting lateral forces, the algorithm exits. If they do not pass all of the checks in either direction, a wall is added in the failing direction and the calculation is rerun, continuing to add walls until a passing configuration is found. Shear walls are checked for combined forces from bending and axial loads. If combined forces exceed wall capacity, columns are added to take the gravity loads.

CONNECTIONS

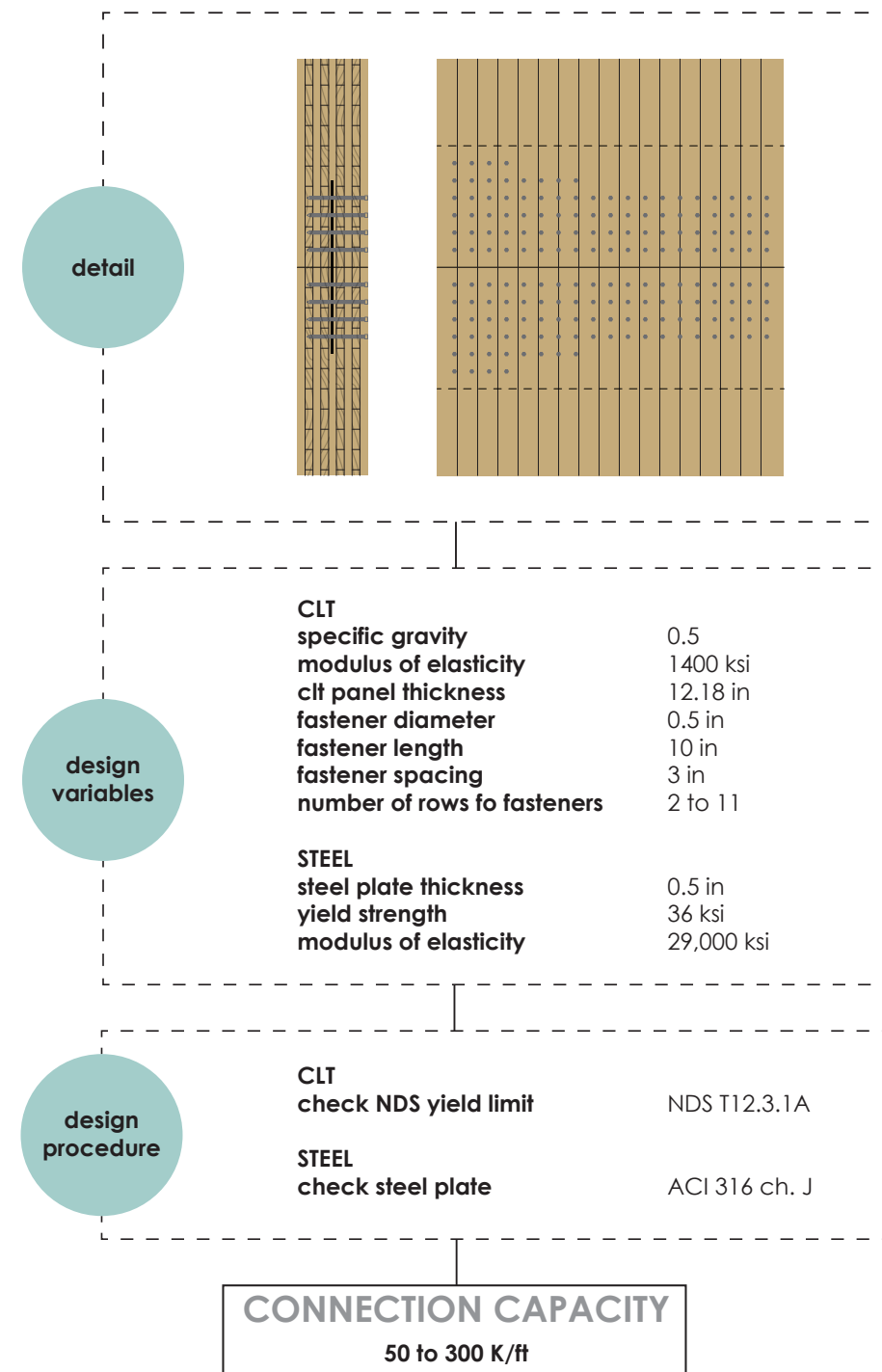


Figure 17. Connection Design Procedure

A single shear wall connection was designed and checked for its ability to resist combined shear and bending from seismic forces at the base of the wall. The connection was designed on a per foot basis looking at the last foot of wall. The connection was checked for an equivalent point force checking limit states for the CLT panel, bolts, and steel plate. Figure 17 has an overview of the connection design procedure and resulting capacity.

OUTPUTS

Based on the results of the structural design, the sizes and material quantities of all relevant structural and facade elements in the building are known. Figure 18 shows the building assemblies used for calculating costs and environmental impacts. Assemblies for floor slabs, beams and columns, roofs, facades, and core walls are used to estimate cost and environmental impacts for the shell and structure of the building.

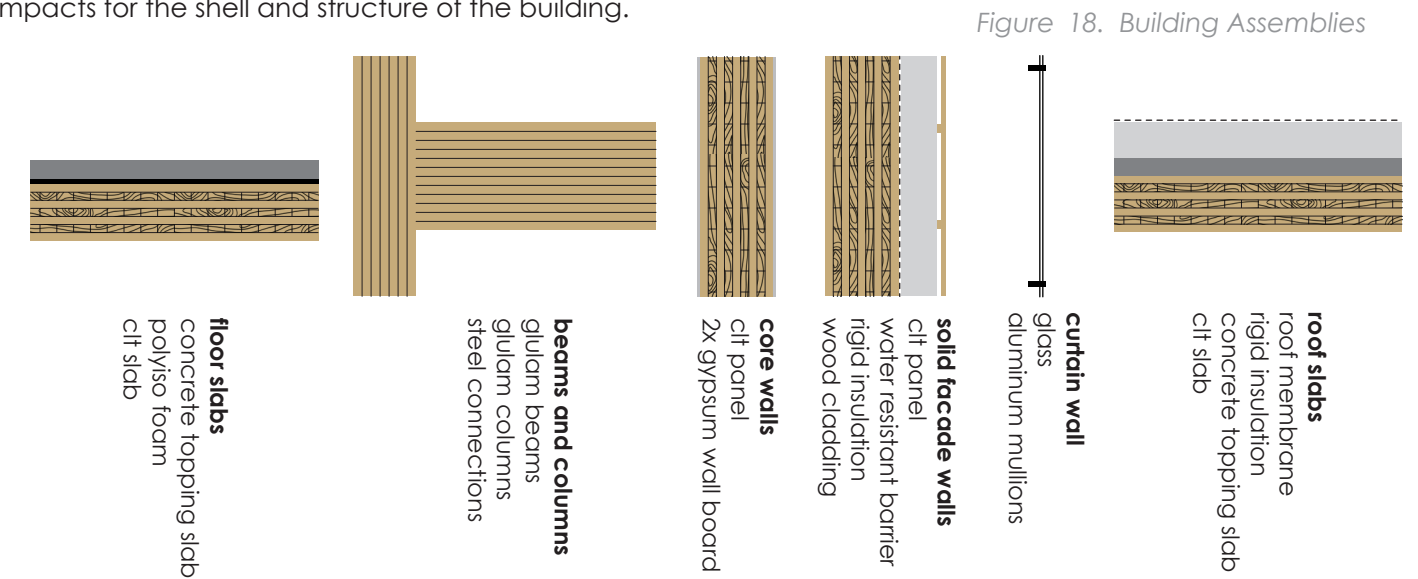


Figure 18. Building Assemblies

Figure 19 shows the cost and environmental impact per unit of the different assemblies. Cost data is derived from a mixture of RS Means^[10] data and cost estimates generated by Hovhannisyanyan^[11]. Environmental impact data is based on data from the Athena Impact Estimator database^[12]. Environmental impacts are calculated excluding credits for carbon sequestration and end of life recycling. If carbon sequestration is included, the total life cycle carbon impact for wood products is negative (more carbon is absorbed during growth than is emitted during manufacturing).

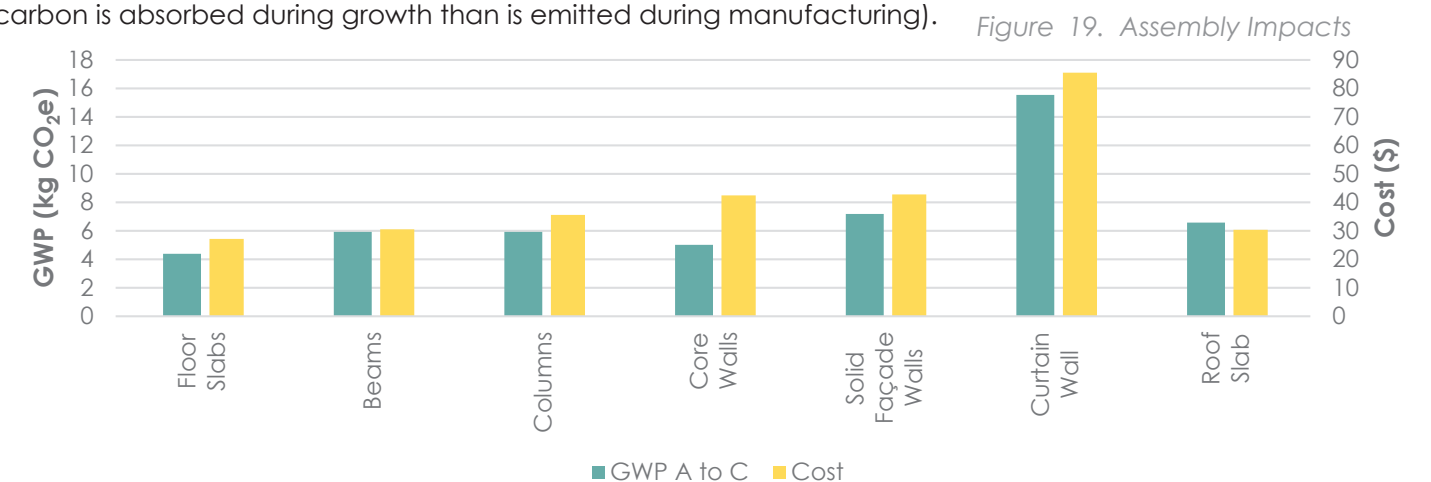


Figure 19. Assembly Impacts

Assembly	Quantity	Material	Cost	GWP
Floor Slabs	ft ²	Concrete Topping Slab	\$6.75/ft ²	0.85 kg CO ₂ e/in-ft ²
		Polyiso Foam	\$2.71/ft ²	0.22 kg CO ₂ e/ft ²
		CLT Slab	\$2.58/in-ft ²	0.36 kg CO ₂ e/in-ft ²
		Steel Connections	1% CLT Cost	0.06 kg CO ₂ e/in-ft ² CLT
Beams	ft ³	Glulam Beams	\$30.24/ft ³	5.17 kg CO ₂ e/ft ³
		Steel Connections	1% Glulam Cost	0.75 kg CO ₂ e/ft ³
Columns	ft ³	Glulam Columns	\$35.27/ft ³	5.17 kg CO ₂ e/ft ³
		Steel Connections	1% Glulam Cost	0.75 kg CO ₂ e/ft ³
Core Walls	ft ²	CLT Panel	\$2.58/in-ft ²	0.36 kg CO ₂ e/in-ft ²
		Gypsum Board	\$11/ft ²	0.64 kg CO ₂ e/ft ²
		Steel Connections	1% CLT Cost	0.06 kg CO ₂ e/in-ft ² CLT
Facade Wall	ft ²	CLT Panel	\$2.58/in-ft ²	0.36 kg CO ₂ e/in-ft ²
		Water Resistant Barrier	\$2.66/ft ²	0.04 kg CO ₂ e/ft ²
		6" Rigid Insulation	\$4.80/ft ²	2.38 kg CO ₂ e/ft ²
		Cladding	\$3.88/ft ²	0.384 kg CO ₂ e/ft ²
		Steel Connections	1% CLT Cost	0.06 kg CO ₂ e/in-ft ² CLT
Curtain Wall	ft ²	Glass	\$85.50/ft ²	6.5 kg CO ₂ e/ft ²
		Aluminum Mullions		9.04 kg CO ₂ e/ft ²
Roof Slab	ft ²	Roof Membrane	\$2.66/ft ²	0.04 kg CO ₂ e/ft ²
		6" Rigid Insulation	\$3.23/ft ²	2.38 kg CO ₂ e/ft ²
		Concrete Topping Slab	\$6.75/ft ²	0.85 kg CO ₂ e/in-ft ²
		CLT Slab	\$2.58/in-ft ²	4.32 kg CO ₂ e/ft ³
		Steel Connections	1% CLT Cost	0.06 kg CO ₂ e/in-ft ² CLT

Table 3. Assembly Impacts

Thus if carbon sequestration is included, the structure with the least impacts would be the one that uses the most wood. Since the aim of this study is to minimize material usage, excluding credits so that the environmental impact of timber reflects the cost of harvesting, manufacturing, and transporting the products, makes more materially intensive buildings have larger impacts. The similarity in scale between cost and global warming potential means that the two measures of efficiency can be used interchangeably to understand the overall impact of a given alternative.

LIMITATIONS

A number of items fell outside the scope of the current exploration. A brief overview of limitations of the current algorithm follows. A number of simplified loading assumptions were made: wind loading was omitted, floor live loads were assumed to be uniform at 50 psf excluding corridor loading, and a number of simplifying assumptions were made in the distribution of seismic forces (rigid diaphragms, simplified accidental torsion).

Additionally, a significant limitation in the layout of beam and column elements is that beam and grid columns match (See Figure 20). This limits the range of floor layouts that can be explored and likely omits some floor layouts that may be more efficient.

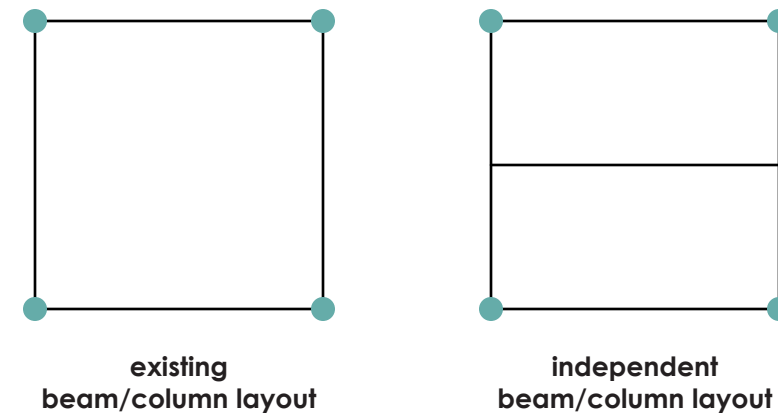


Figure 20. Existing/Future Beam Column Layout

Finally, the analysis of only a 60-minute fire rating for beam, column, and slab elements is insufficient to develop the full fire-resistance as required by codes for buildings greater than 65 feet in height.

LESSONS LEARNED

In the process of creating the custom algorithm, I ran into a number of challenges. One major challenge was that the built-in components in Grasshopper and its plug-ins often become highly inefficient and limiting to work with when implementing complicated procedures. To overcome this, I wrote custom C# code to simplify and streamline complicated procedures that would be difficult to implement using stock Grasshopper components. The custom C# scripts are included in Appendix C.

I also encountered difficulty transferring data into and out of Grasshopper. Although many plug-ins, including Lunchbox, provide components to read from and write to Excel, these components are time-intensive to execute slowing down the overall execution of the code. As a solution, I ended up internalizing data from excel into the grasshopper script and appending results to text files significantly speeding up the execution of the code.

One final lesson learned involved rethinking the order and method of problem solving to use available information to solve problems out of the traditional order and to find unknown solutions. While engineering judgment is typically used to determine the starting point for member sizes, when approaching problems with variable inputs code has to be flexible enough to adjust for all possible variations.

REFERENCES

- [1] Robert McNeel & Associates (2015). Rhinoceros 5 3D Commercial Release 2015-8-10 [Computer software]. Seattle, WA: McNeel.
- [2] Rutten, D. (2014). Grasshopper 3D v0.9.0076 [Computer software]. Seattle, WA: McNeel.
- [3] Preisinger, C. (2013). Linking Structure and Parametric Geometry. *Archit Design*, 83: 110-113.
- [4] Miller, N. (2015). LunchBox for Grasshopper v2015.7.20 [Computer software]. Omaha, NE: Proving Ground.
- [5] International Code Council. (2015). International Building Code. Country Club Hills, IL: International Code Council.
- [6] American Society of Civil Engineers. (2010). Minimum Design Loads for Buildings and Other Structures. Reston, VA.
- [7] American Wood Council. (2015). National Design Specification for Wood Construction. Leesburg, VA.
- [8] FP Innovations, & Binational Softwood Lumber Council. (2013). CLT Handbook - US Edition. Pointe-Claire, Quebec. Farmington Hills, MI: American Concrete Institute.
- [9] American Concrete Institute. (2011). Building Code Requirements for Structural Concrete (ACI 318-11).
- [10] RS Means Company. (2013). Building Construction Cost Data. Kingston, MA.
- [11] Hovhannisyan, M. (2015). *Wood Cityscapes: Mass Timber Office Buildings*. Master's thesis, University of Washington, Seattle, Washington.
- [12] Athena Sustainable Materials Institute. (2015). Athena Impact Estimator for Buildings 5.1.01 [Computer software]. Ottawa, Canada: Athena Institute.

RESULTS

Two primary implementations of the parametric algorithm were executed to understand the influence of different variables and different possible building configurations. The first was a general analysis looking at the broad scale interactions of primary variables. The second implementation tested the adaptability of the algorithm for custom geometry input.

GENERAL ANALYSIS

The first analysis was used to understand the behavior of the buildings as a whole and to guide further study.

EXECUTION

Analysis of the model for varying parametric inputs was performed using a brute force iteration method. Within each zoning envelope, parameters were varied in the ranges shown in Table 4.

Parameter	Range
Include Charring	True, False
Composite Slabs	True, False
Composite Beams	True, False
X-Bay Length	20' to 40' in 5' increments
Y-Bay Length	20' to 40' in 5' increments
First Floor Height	15' to 20' in 1' increments
Typical Floor Height	11' to 15' in 1' increments
Concrete Slab Thickness	2" to 3" in 0.5" increments
CLT Wall Thickness	12"
Length of Facade Walls	25'

In total 18,000 iterations were executed for each of the zoning envelopes. The run time for each model varied based on the number of elements with low-rise buildings having an iteration time of seven seconds, mid-rise buildings having an iteration time of three seconds, and high-rise buildings having an iteration time of ten seconds. About a third of the iterations resulted in successful buildings. About half of the buildings were not viable as floor depth were too deep to maintain an eight foot minimum clear height from top of floor to bottom of beam. About a quarter of buildings were not viable as manufactured depths of CLT were insufficient to span the required span lengths. As beams were only located along grid lines, at large column spaces slabs span up to 40 feet. The rest of the unsuccessful iterations were due to iteration times being too short for the model to completely execute.

Table 4. Parametric Inputs

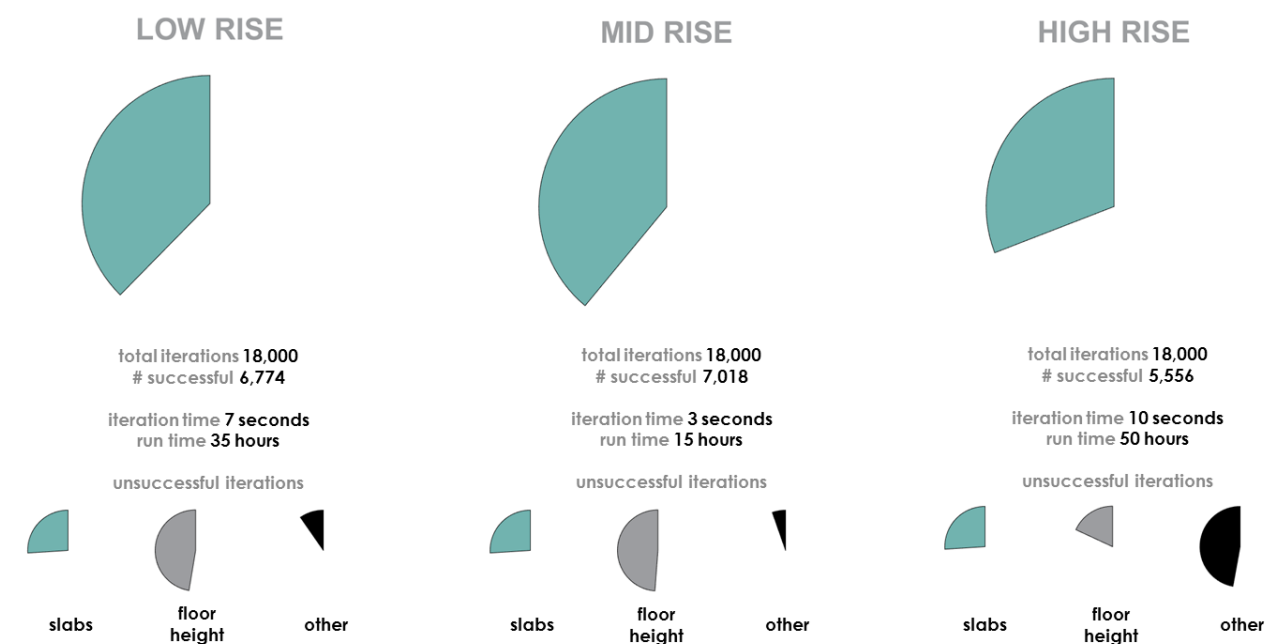
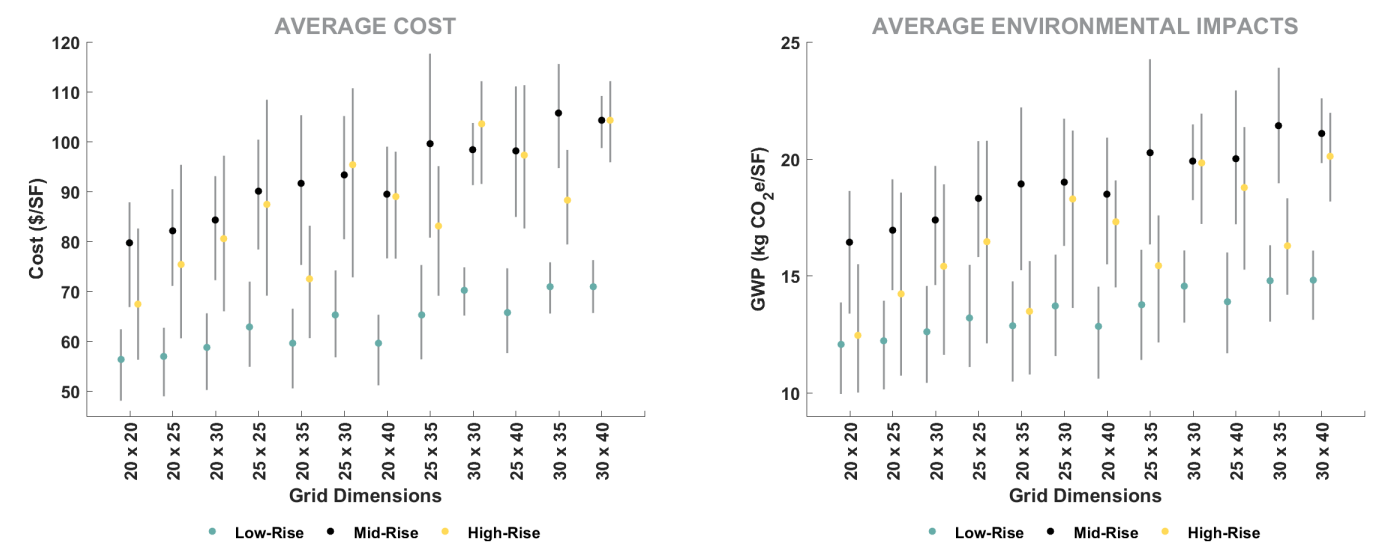


Figure 21. Execution

GENERAL FINDINGS

Based on the results from each successful iteration, average, maximum, and minimum costs and environmental impacts per square foot of the building were calculated as shown in Figure 22.

Figure 22. Average Cost and Environmental Impacts



These results show that mid- and high-rise buildings are more materially intensive than low-rise buildings. And buildings with larger grid areas, and therefore longer spans, are also more material intensive. Differences between the different zoning envelopes likely arise from the ratio of facade to floor area. While increasing impacts with spans are likely due to the increased amount of material in the floor slabs. The effect of these two variables can be seen by looking at the breakdown of cost and environmental impact per square foot for a single mid-rise iteration, as shown in Figure 23.

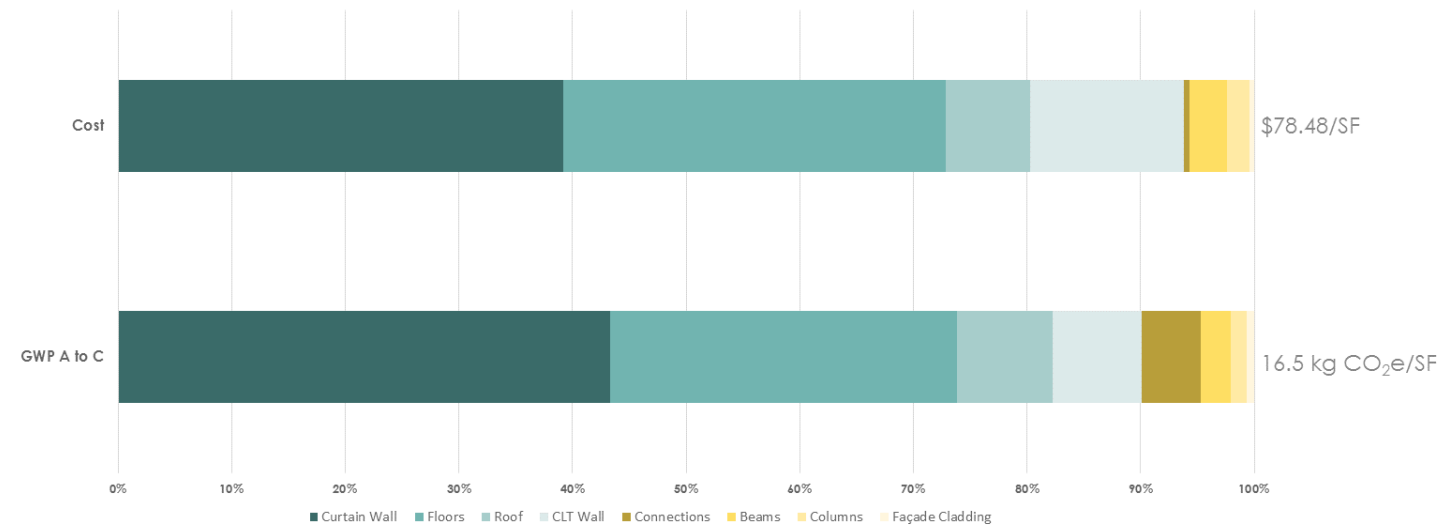


Figure 23. Cost and Environmental Impacts, Mid-Rise Iteration

The graph shows that the curtain wall and floor slabs contribute roughly 75 percent of the total cost and global warming potential of the building. Thus differences between iterations is primarily driven by differences between these two assemblies. Further analysis and strategies for optimizing timber buildings will focus on these two assemblies.

SHEAR WALLS

In order to isolate and understand factors affecting shear wall layout a simplified analysis was done using the floor plan and variables shown in Figure 24. As primary variables, the building height, floor-to-floor heights, length of shear walls along the facade, and wall thicknesses were varied. A total of 23,040 iterations were completed resulting in a thorough understanding of shear wall layout for the given simplified 100 foot by 100 foot floor plan with a 20 foot by 20 foot core.

Figure 24. Shear Wall Analysis

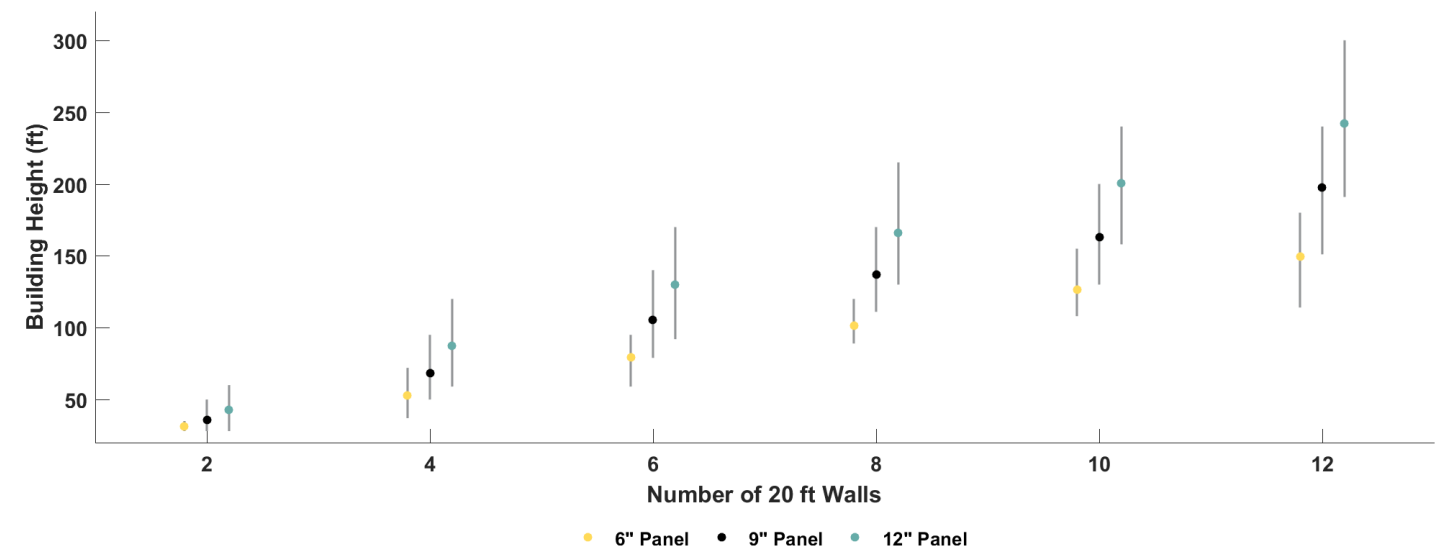
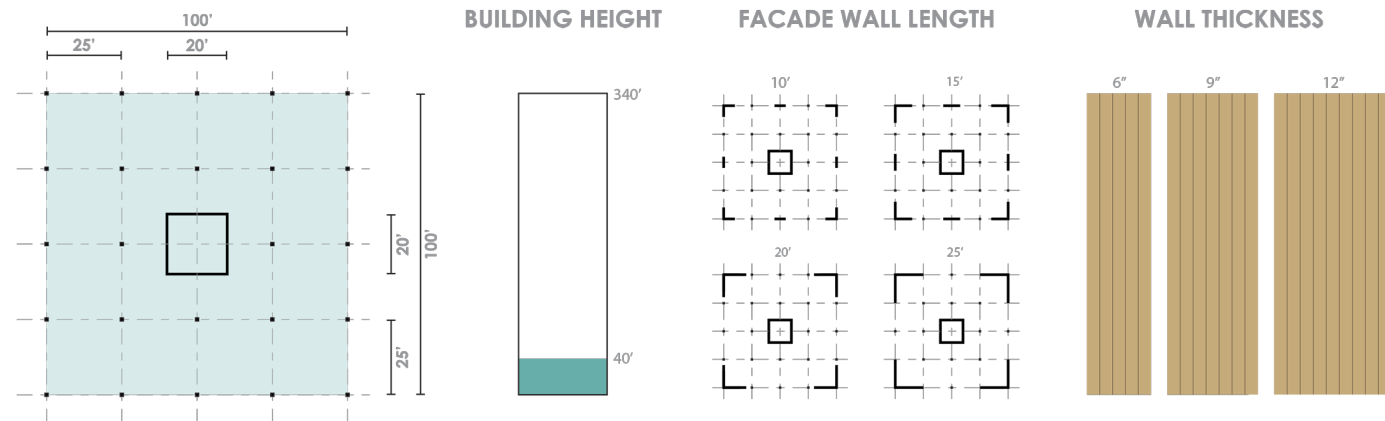


Figure 25. Varying Shear Wall Thickness

Figure 25. Varying Shear Wall Thickness

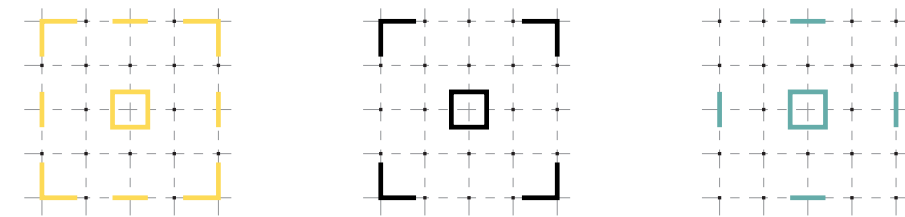


Figure 26. 6", 9", and 12" Wall Layouts, 100' Tall Building

While the configuration with 12 inch CLT panels is closest to the industry norm, fully glazed facades, the 6 inch panel configuration has significant benefits if owners and developers can be convinced that buildings with opaque walls along the facade will perform well and can be sold.

Key benefits include reduced facade costs (32% reduction, \$1 million) and improved thermal performance. From a daylighting perspective, All three alternatives have glazing ratios at or above 40 percent. This is in line with prescriptive requirements in the Seattle Energy Code^[1] that dictate a maximum glazed ratio of 40 percent unless the performance of the facade is tested using advanced energy modeling tools.

Figure 27. Varying Facade Shear Wall Length

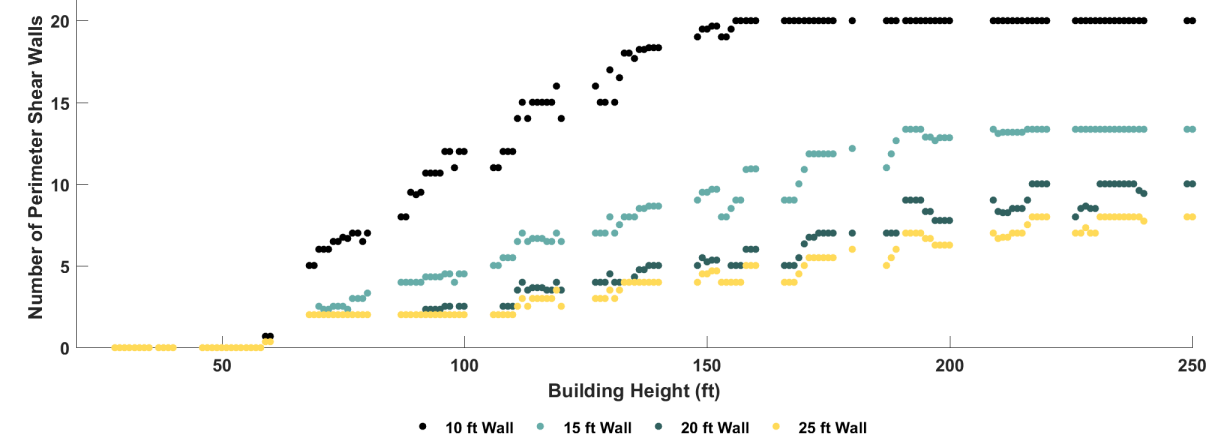


Figure 27 shows the effects of instead varying the length of walls along the perimeter of the building using 12 inch CLT panels. The results are illustrated for a 100 foot tall building in Figure 28, showing that as the shear walls become shorter more are needed to resist forces. This is due to tensile and compressive bending forces at the ends of the wall increasing as walls become shorter. Despite this effect, all four test layouts still maintain glazing ratios at or above 40 percent. With the building with 10 foot long walls having the lowest facade costs (20% reduction from building with 20' walls).

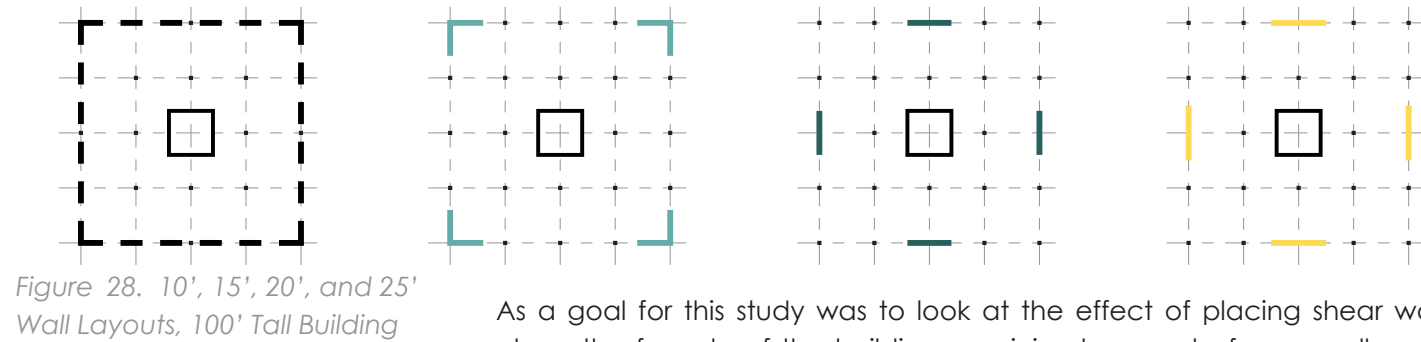


Figure 28. 10', 15', 20', and 25' Wall Layouts, 100' Tall Building

As a goal for this study was to look at the effect of placing shear walls along the facade of the building, a minimal amount of core walls were used. In practice, the developer driven standard of completely glazed facades would likely lead to more robust core layouts eliminating the need for shear walls along the perimeter of the building for low- and mid-rise buildings. For high-rise buildings, the increased demands on shear walls in the majority of cases necessitate shear walls along the facade. In this case, the design work to implement more advanced timber systems (e.g. post-tensioned rocking CLT walls) or a hybrid system with concrete shear walls and timber gravity system are more likely to fit within the demand for fully glazed facades.

As the architecture, engineering, and construction industry moves into the future, finding ways to shift developer and building occupant expectations away from fully glazed facades would help to improve the performance and efficiency of buildings. While this thesis presents no clear path to this market transformation, some factors that may provide incentives are: building regulations further incentivizing opacity along the facade of buildings, economics tied to the increasing costs of energy and carbon, and test-cases showing that buildings do not have to be fully glazed to provide pleasant and marketable working environments.

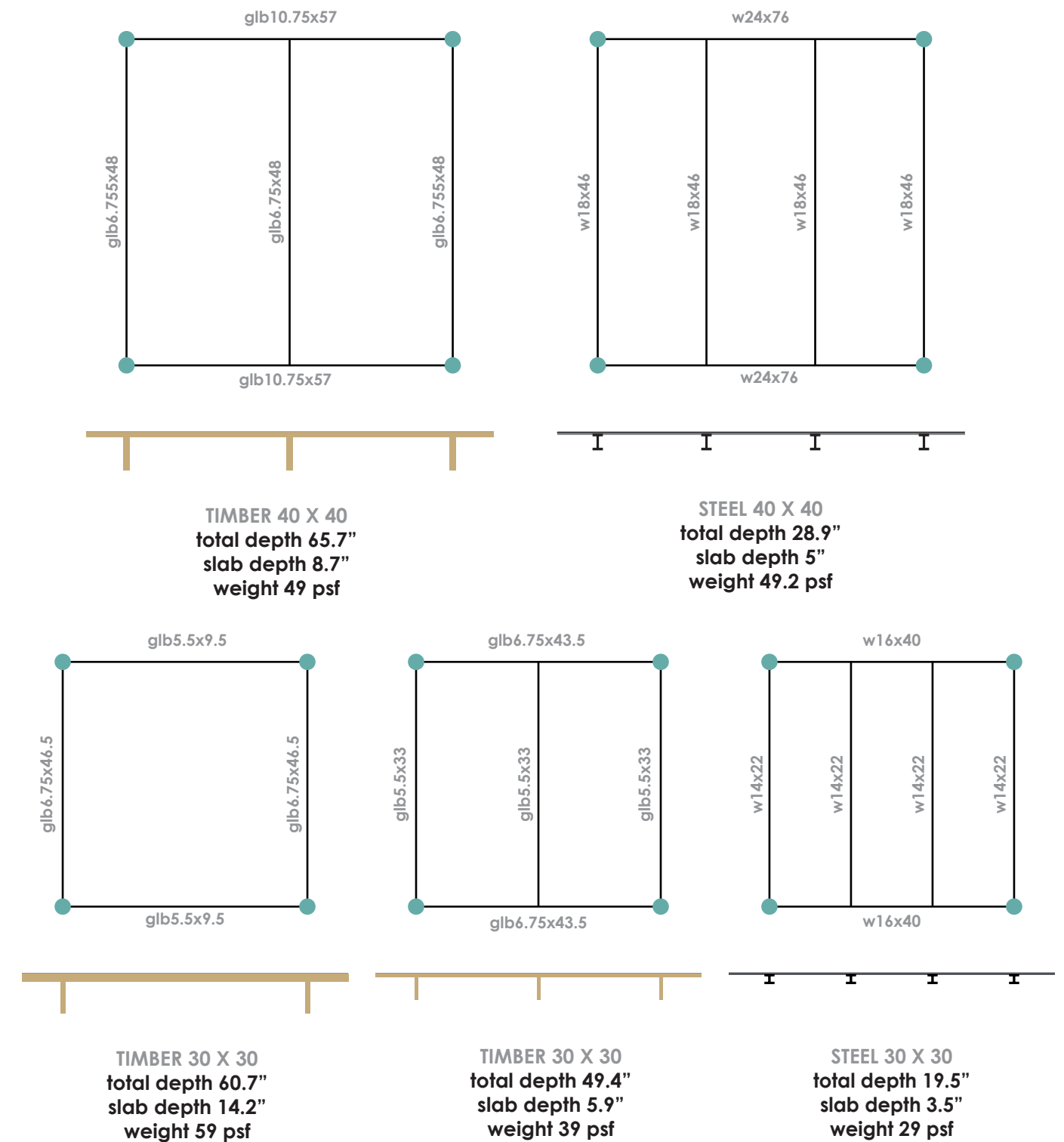
FLOOR SYSTEMS

As the second most materially intensive assembly in timber buildings, optimized slabs and floor systems will have a significant effect on the overall performance of timber buildings. Two primary factors affect the efficiency and suitability of timber floor systems: the total quantity of material in the system (driven by slab depth) and the overall floor system depth (driven by beam depth).

To be considered as a viable alternative to steel and concrete systems, timber floor systems will need to show some competitive advantages to existing systems. To understand the strengths and weaknesses of timber floor systems, a comparison to typical steel floor systems was done. Figure 29

shows depths and weights for 30 foot square and 40 foot square grids for steel and timber systems.

Figure 29. Comparison of Steel and Timber Floor Systems

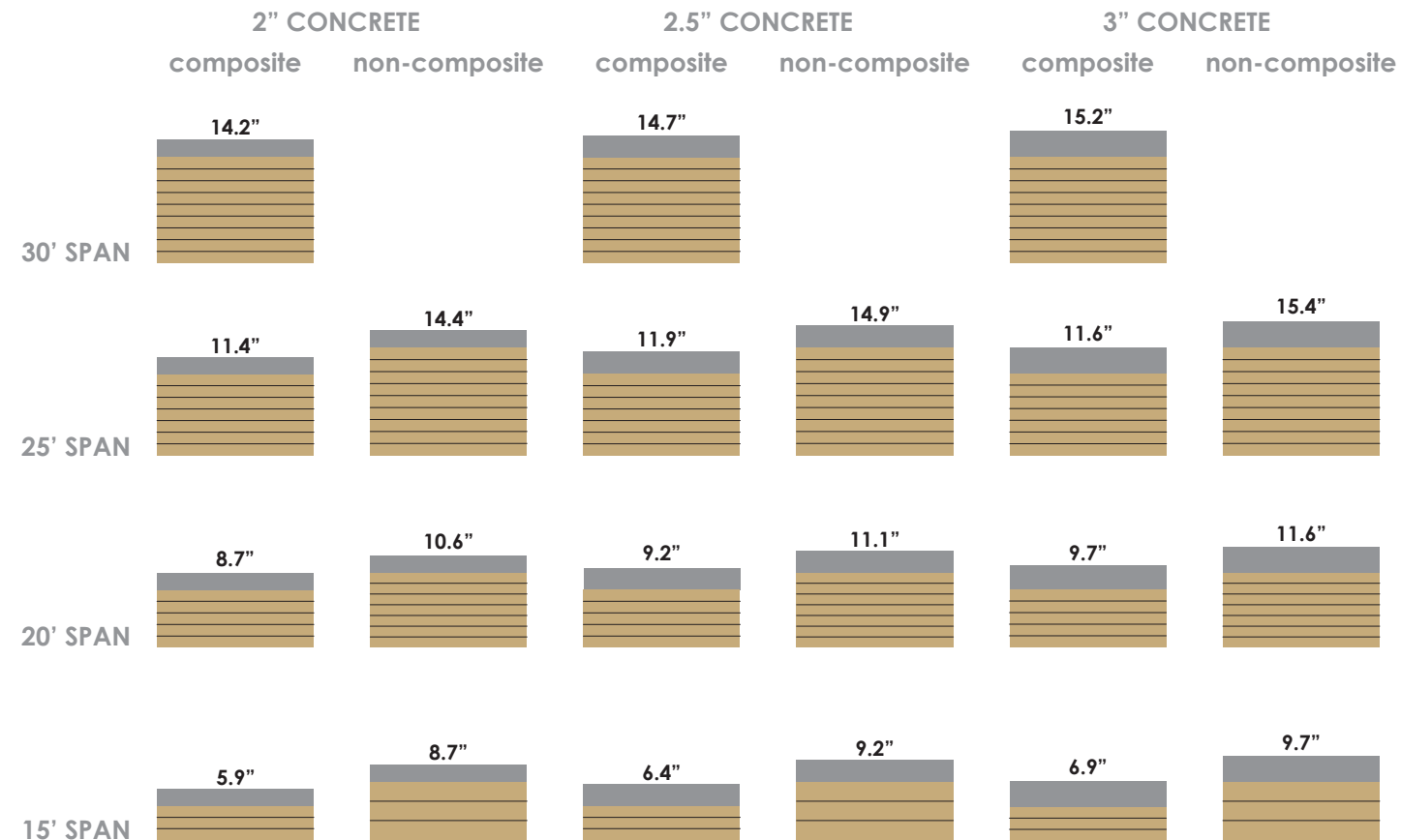


At the outset of research, one hypothesis was that timber structural systems would prove competitive by reducing building weight thereby reducing demands on foundations and lateral systems. In preliminary testing, this

hypothesis was found to be false with timber and steel systems having similar weights in the best cases and timber systems weighing significantly more in the worst cases. Beyond this, timber systems also have total system depths roughly double those of steel systems. This puts timber at a competitive disadvantage as deeper floor systems equate to greater floor-to-floor heights reducing the number of floors that can be constructed within a given zoning envelope and increasing the amount of facade that needs to be constructed per square foot of built area.

Several strategies were tested to understand pathways for improving the performance of timber floor systems. The first strategy for reducing the material intensity and cost of floor slabs was having the concrete topping slab act compositely with the floor slab. Standard CLT slab construction requires a topping slab to minimize acoustic transmission through the floor and to help with vibrations. By using this layer structurally, minimal decreases in total slab thickness are possible. More important to minimize slab depth is shortening spans. While the full effects of adding beams to shorten spans were not studied, it is likely more efficient to add beams reducing slab thickness. Errors in the code, found after completion of results, were oversizing the thickness of composite CLT slabs by one to two inches. Corrected thicknesses are shown in Figure 29 and Figure 30; however all other results include this error. The design of non-composite slabs was not affected by this error.

Figure 30. Composite and Non-Composite Floor Slabs



While slabs contribute the majority of the weight and material of the floor system, the beams contribute the majority of the depth of the system. Figure 31 shows the overall floor system depth for varying grid dimensions.

The total system depth varies between three and six feet with corresponding minimum floor-to-floor heights varying between 11 and 14 feet. Composite beams have a modest effect on the floor depth, reducing depths by approximately 10 inches. The overall depth is still significant as compared to steel and concrete systems.

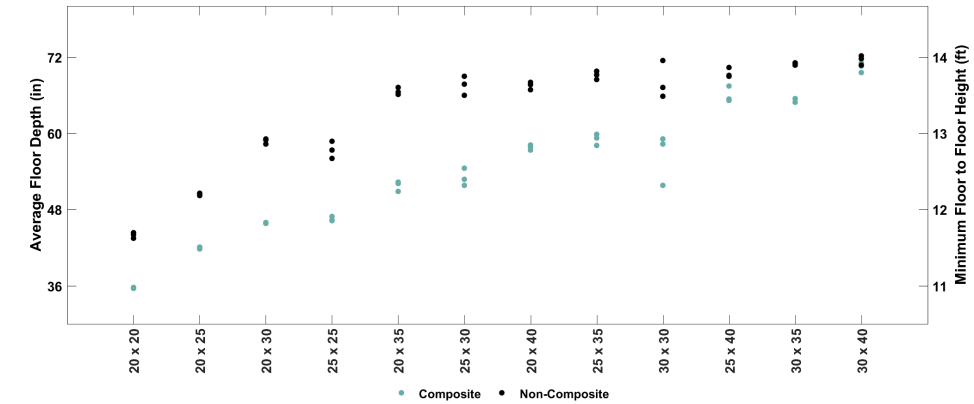


Figure 31. Composite and Non-Composite Beams

There are two possible pathways to overcoming the depth of timber systems. The first is the development of novel timber floor systems that are more efficient. This is likely a less accessible pathway in the short term as it will require investment in research and development. The second, is to provide zoning height incentives for timber buildings allowing developers to build the same number of floors. Zoning height incentives are a more viable short term solution to push timber construction. A survey of buildings in King County^[2] serves as guidance for the required height increases for timber systems in each of the three zoning envelopes (See Figure 32). The required height increase varies from 15 to 50 percent depending on the chosen floor system.

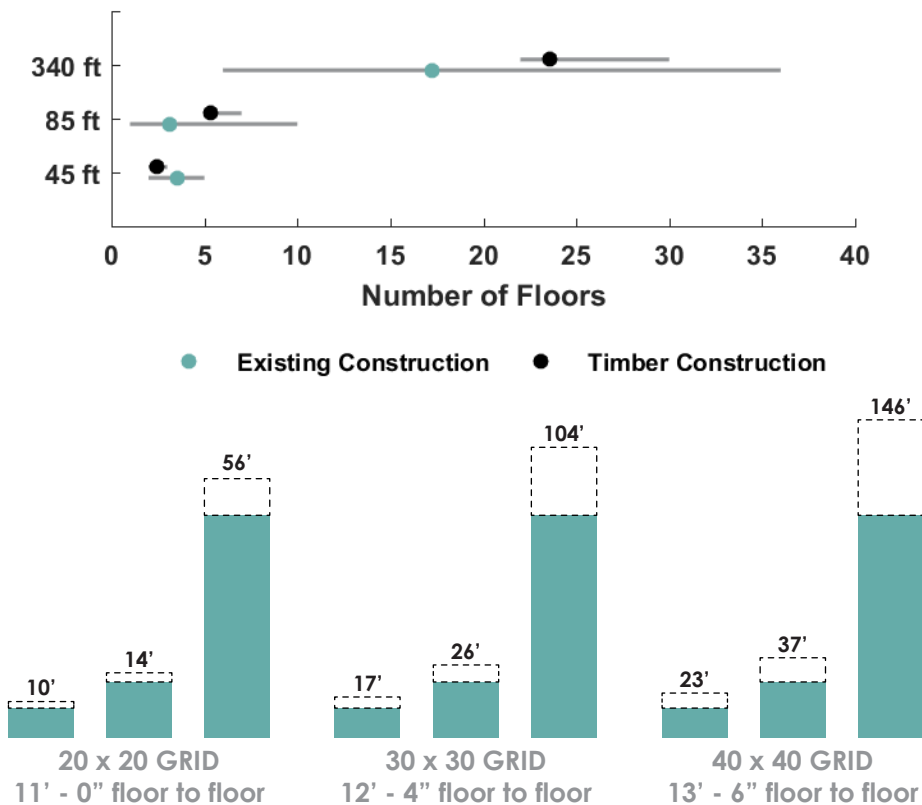


Figure 32. Zoning Height Increases

CHARRING

The effect of fire and strategies for mitigating the flammability of timber are an ongoing area of study and are one of the main deterrents to the widespread construction of tall mass timber buildings. To understand the impact of including a 60-minute fire rating for timber members, structure material quantities were calculated for structures including and excluding charring. Columns had the most significant material increase with an increase of about 3 percent as they are strength controlled. Beams had a smaller material increase of 1 percent as beam design is controlled by deflections and they are therefore initially oversized for strength. Charring has no effect on slabs as they are oversized for vibrations.

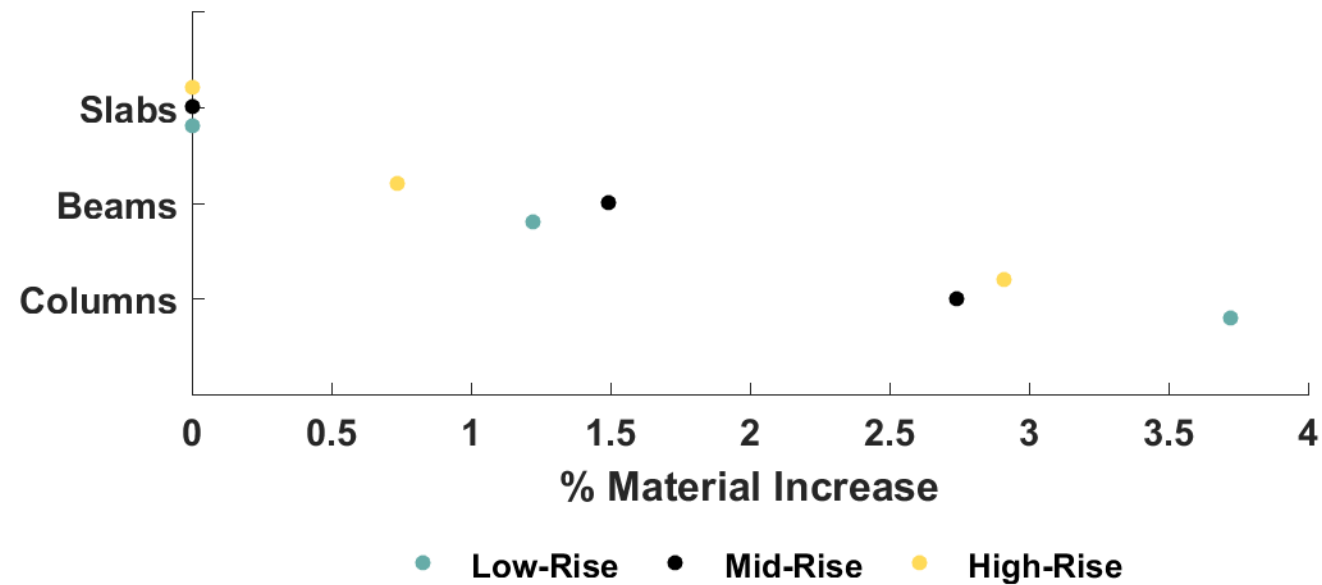


Figure 33. Charring

While this study shows that a 60-minute fire rating has a minimal effect on material quantities, it omits the full impact of alternate fireproofing methods and therefore does not provide a comprehensive understanding of the full costs. Additionally, for high-rise buildings a 60-minute fire rating is probably insufficient for the life-safety of building occupants.

ARCHITECTURAL TEST APPLICATIONS

To test the algorithm as an architectural design tool, the algorithm was adapted to take user input geometry. A basic plan and section for a low-rise (Figure 34), mid-rise (Figure 35), and high-rise (Figure 36) test buildings were developed. Based on these test cases a couple of key design issues for shear walls were uncovered. In order to get core layouts to work, longer walls leading to squarer core layouts are more effective. Additionally, matching the length of all walls in each direction is more efficient than having a mixture of longer and shorter walls.

The revised algorithm provides a quick way to test the efficiency of user specified architectural layouts against the benchmark generated through the iterative analysis.

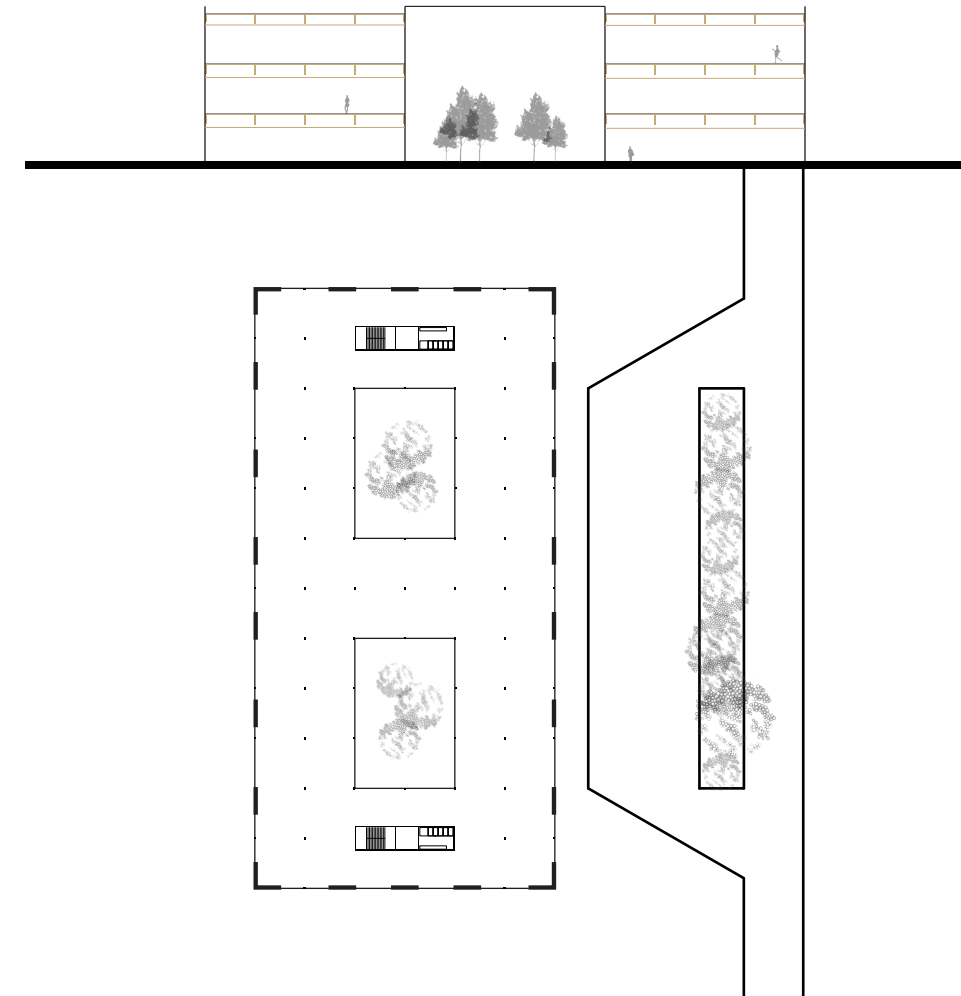
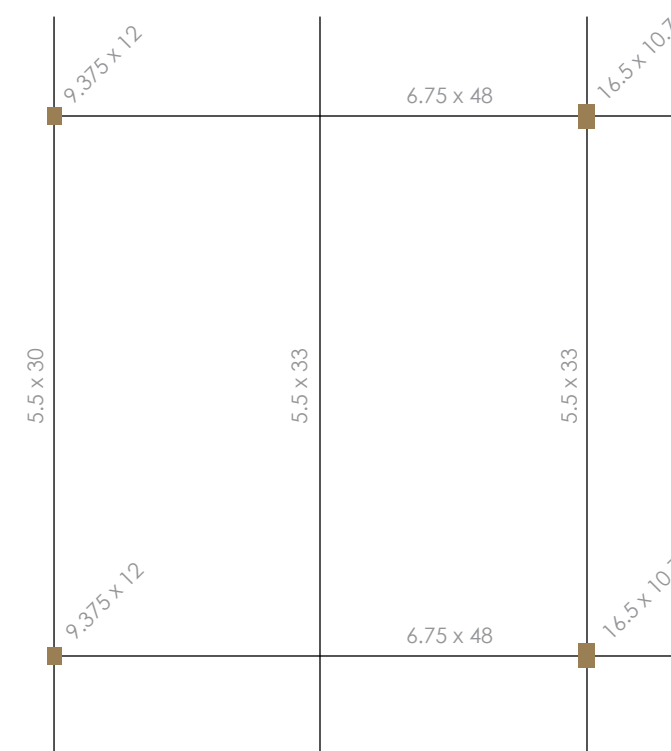
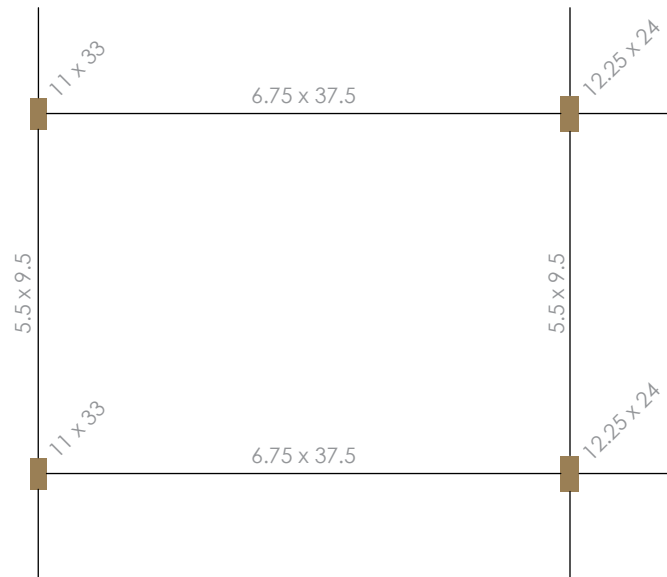


Figure 34. Low-Rise Test Building



grid 30' x 30'
slab 2" concrete over 5.125" CLT
cost \$50.62/sf
gwp 10.79 kg CO₂e/sf

Figure 35. Mid-Rise Test Building



grid 30' x 20'
slab 2" concrete over 12.18" CLT
cost \$93.51/sf
gwp 18.05 kg CO₂e/sf

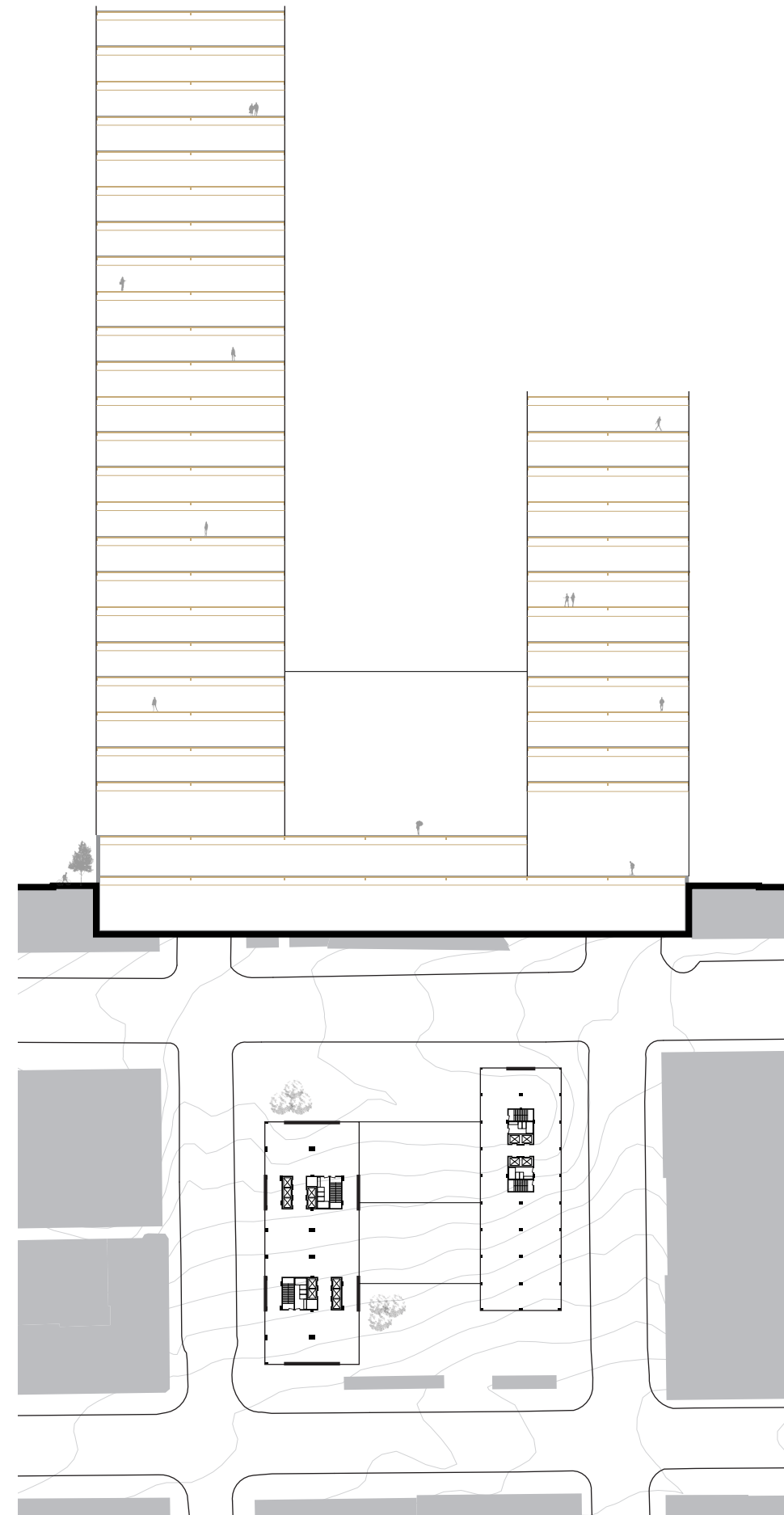
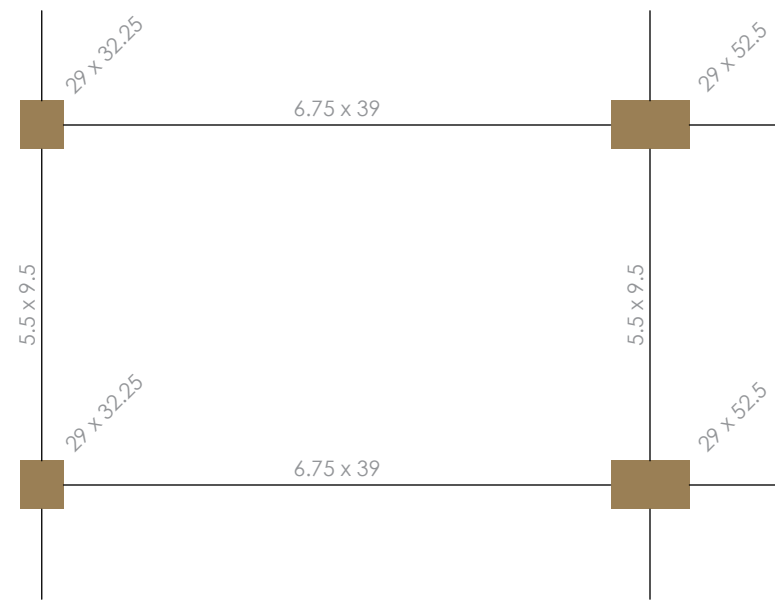


Figure 36. High-Rise Test Building



340' TOWER

grid 35' x 20'
slab 2" concrete over 6.875" CLT
cost \$94.38/sf
gwp 17.83 kg CO₂e/sf

REFERENCES

- [1] City of Seattle. (2014). *Seattle Energy Code*. Country Club Hills, IL.: International Code Council.
- [2] King County Department of Assessment. (2013). *Assessments Data* [CSV file]. Retrieved from <http://info.kingcounty.gov/assessor/DataDownload/default.aspx>.

Figure 37. High-Rise Framing Plan

CONCLUSIONS

This thesis has explored opportunities for optimizing mass timber buildings through the development and execution of a custom parametric algorithm.

- Facades and floor systems contribute roughly 75 percent of the impacts of the building, optimization efforts focused on these two systems will have the greatest impact.
- Facades can be optimized by reducing the amount of glazing, one method for reducing the area of glazing is placing shear walls along the perimeter of the building.
 - While it is possible for this strategy to work with overall building performance goals, it is unlikely to meet developer standards for Class A office space.
- Optimized floor systems are the best near-term option for improving the efficiency of timber structures. There are two main optimization strategies:
 - Development of innovative composite floor systems, that reduce system depth and material usage.
 - Providing zoning height allowances to offset floor-to-floor height increases driven by the depth of timber floor systems.
- The algorithm provides a way to test the viability of timber structures and provides schematic level sizing of structural members.
 - This could allow architects to quickly test the implications of building with timber and understand whether or not it is a competitive alternative for a given project.
- Data from this study also serves to benchmark timber construction establishing the general range for the cost and environmental impact of timber construction.

SUMMARY OF REVIEW COMMENTS

As a part of this work a final review with guest reviewers was held, a brief summary of comments from this review follows. Reviewers were: Scott Crawford, Architect-LMN Architects, Brett Mozden, Engineer-Swenson Say Faget, Susan Jones, Architect-Atelier Jones, Joe Mayo, Architect-Mahlum, and John Stanton, Civil Engineering Professor-University of Washington.

Comments from the review focused on how the further development of the program could provide comparisons that would help understand the

tipping points where timber becomes attractive as compared to steel and concrete systems. These might involve other variables (such as daylighting) that help to assess architectural criteria, or modifying the program to directly compare to other systems such as steel, concrete, or light frame timber construction. They also discussed moving away from how timber buildings are supposed to look and instead focusing on how timber technologies can work within existing norms for office construction. While the time frame for this thesis was limited, these suggestions provide guidance for how this work can be adapted to further serve practicing architects and engineers.

FUTURE WORK

While the current study provides a comprehensive overview of key structural issues affecting timber construction, there are a number of potential future additions that would serve to make the tool more robust and more useful in practice.

- A study of independent beam-column grids to find systems optimized for weight and depth.
- Refinement of loading included in the model—adding wind loading, corridor loading, and refining seismic loading/design
- Research and development of timber composite floor assemblies
- Further comparison of timber systems with steel and concrete systems
- Addition of thermal and daylighting analysis to the algorithm
- Continued architectural application and exploration of timber structures.
- Further study of the cost and material impacts of different fire performance strategies (charring vs fireproofing)

CASE STUDIES

A

The following sections provide an overview of existing and proposed mass timber buildings around the world. The buildings selected represent the state of practice for mass timber office buildings four stories or more in height. A number of tall (eight or more stories) residential buildings are also included.

TREET



Figure 38. Rendering of Treet (from ARTEC AS)

LOCATION: BERGEN, NORWAY
COMPLETION: 2016
TYPE: HOUSING
AREA: 3780 SQ. M
HEIGHT: 14 STORIES
DEVELOPER: BOB EIENDOMSUTVIKLING
ARCHITECT: ARTEC AS
STRUCTURAL ENGINEER: SWECO
STRUCTURE OVERVIEW: GLULAM FRAME SUPPORTING PREFABRICATED UNITS

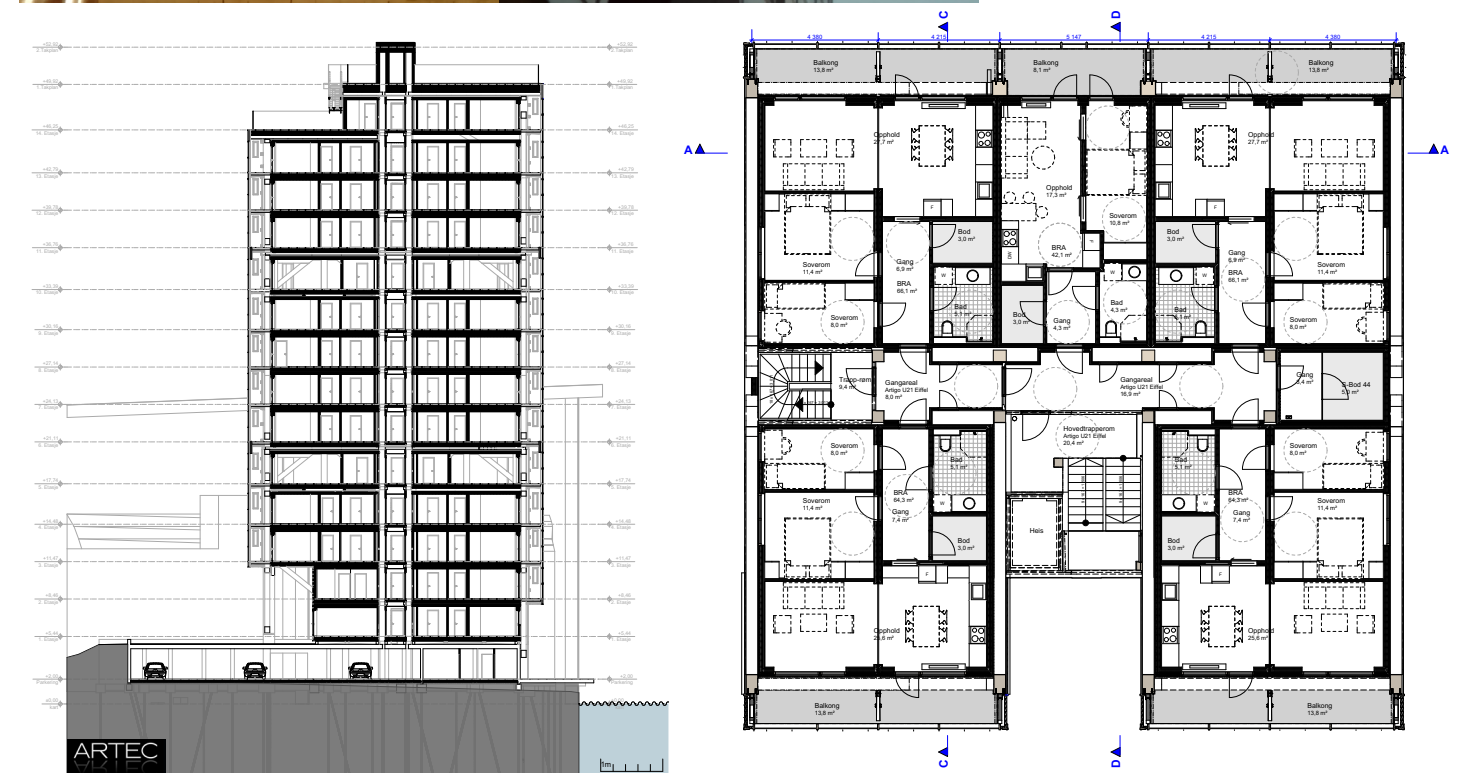


Figure 39. Stair Detail (Upper Left) (from author)

Figure 40. Exterior Facade (Upper Right) (from author)

Figure 41. Building Section (Lower Left) (from ARTEC AS)

Figure 42. Plan (Lower Right) (from ARTEC AS)



ARTEC

FRAMEWORK



Figure 43. Rendering of Framework
(from LEVER Architecture)

LOCATION:	PORTLAND, OR
COMPLETION:	2016
TYPE:	MIXED USE
HEIGHT:	12 STORIES
DEVELOPER:	PROJECT
ARCHITECT:	LEVER ARCHITECTURE
STRUCTURAL ENGINEER:	KPFF
STRUCTURE OVERVIEW:	GLULAM BEAMS AND COLUMNS WITH CLT DECKING AND CORE WALLS



Figure 44. Rendering of Framework
(from LEVER Architecture)

WOOD INNOVATION AND DESIGN CENTER



Figure 45. Wood Innovation and Design Center (from Michael Green Architecture)

LOCATION: PRINCE GEORGE, BC, CANADA
COMPLETION: 2014
TYPE: OFFICE AND LABORATORY
AREA: 4820 SQ. M
HEIGHT: 8 STORIES
OWNER: UNIVERSITY OF BRITISH COLUMBIA
ARCHITECT: MICHAEL GREEN ARCHITECTURE
STRUCTURAL ENGINEER: EQUILIBRIUM CONSULTING INC
STRUCTURE OVERVIEW: CLT FLOOR AND WALL PANELS WITH GLULAM BEAMS AND COLUMNS, NO CONCRETE ABOVE FOUNDATION



Figure 46. Typical Floor Construction (Left) (from Michael Green Architecture)

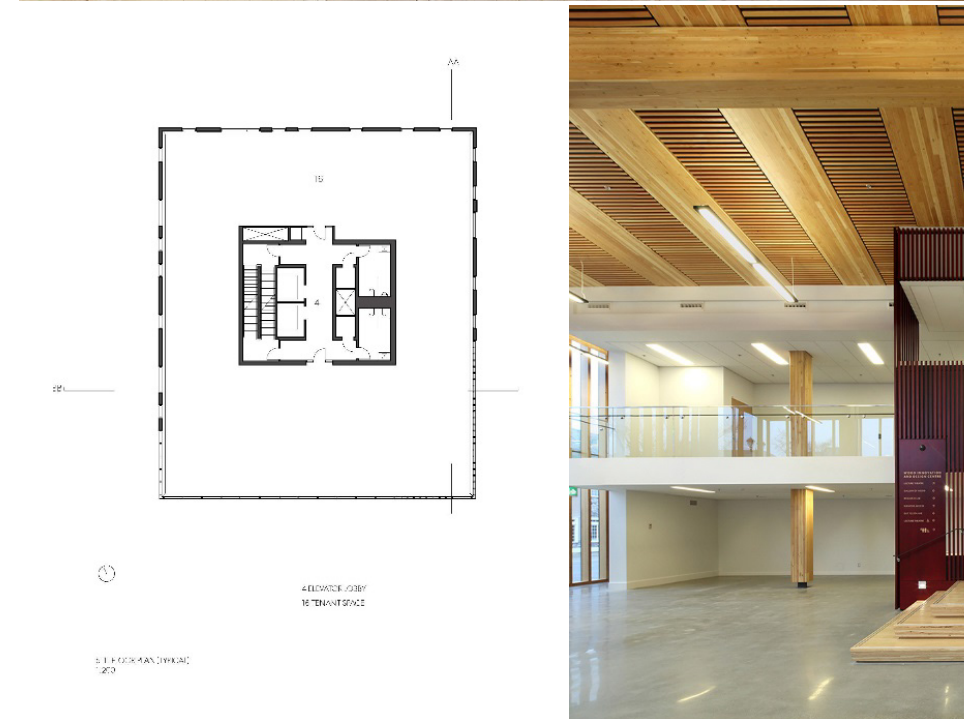


Figure 47. Typical Floor Plan (Lower Left) (from Michael Green Architecture)

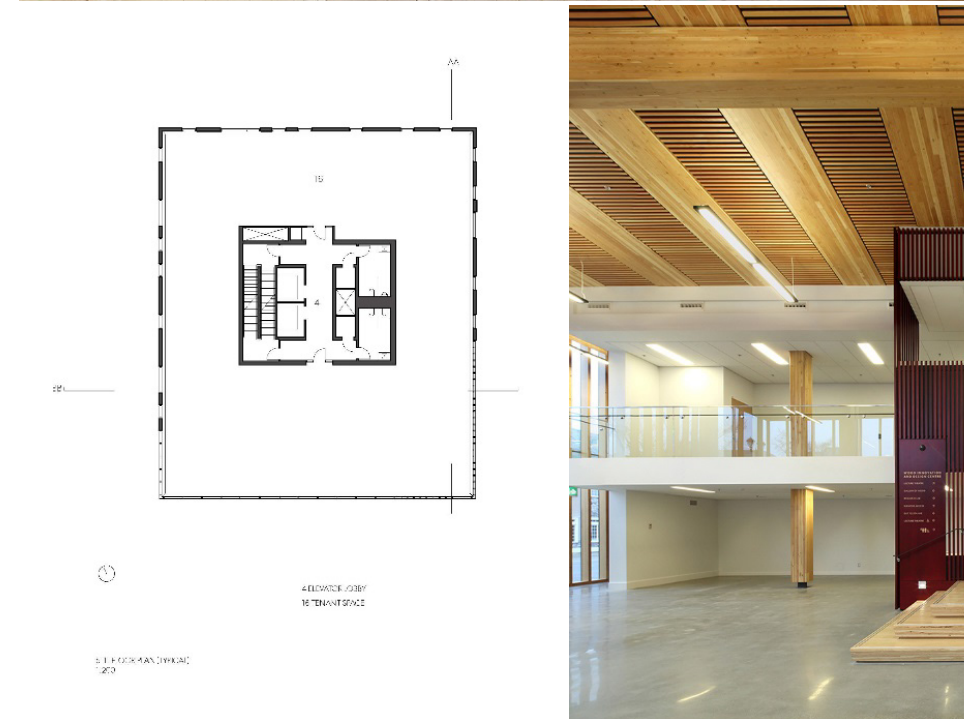


Figure 48. Lobby (Lower Right) (From Michael Green Architecture)

STADTWERKE LÜBECK HQ



Figure 49. Stadtwerke Lübeck Headquarters (from FpbrAG)

LOCATION:	LÜBECK, GERMANY
COMPLETION:	2014
TYPE:	OFFICE
AREA:	13,856 SQ. M
HEIGHT:	4 STORIES
OWNER:	STADTWERKE LÜBECK GMBH
ARCHITECT:	KLEIN ARCHITEKTEN
WOOD USE:	2,438 CU. M
STRUCTURE OVERVIEW:	GLULAM BEAMS AND COLUMNS WITH CLT FLOOR DECKS

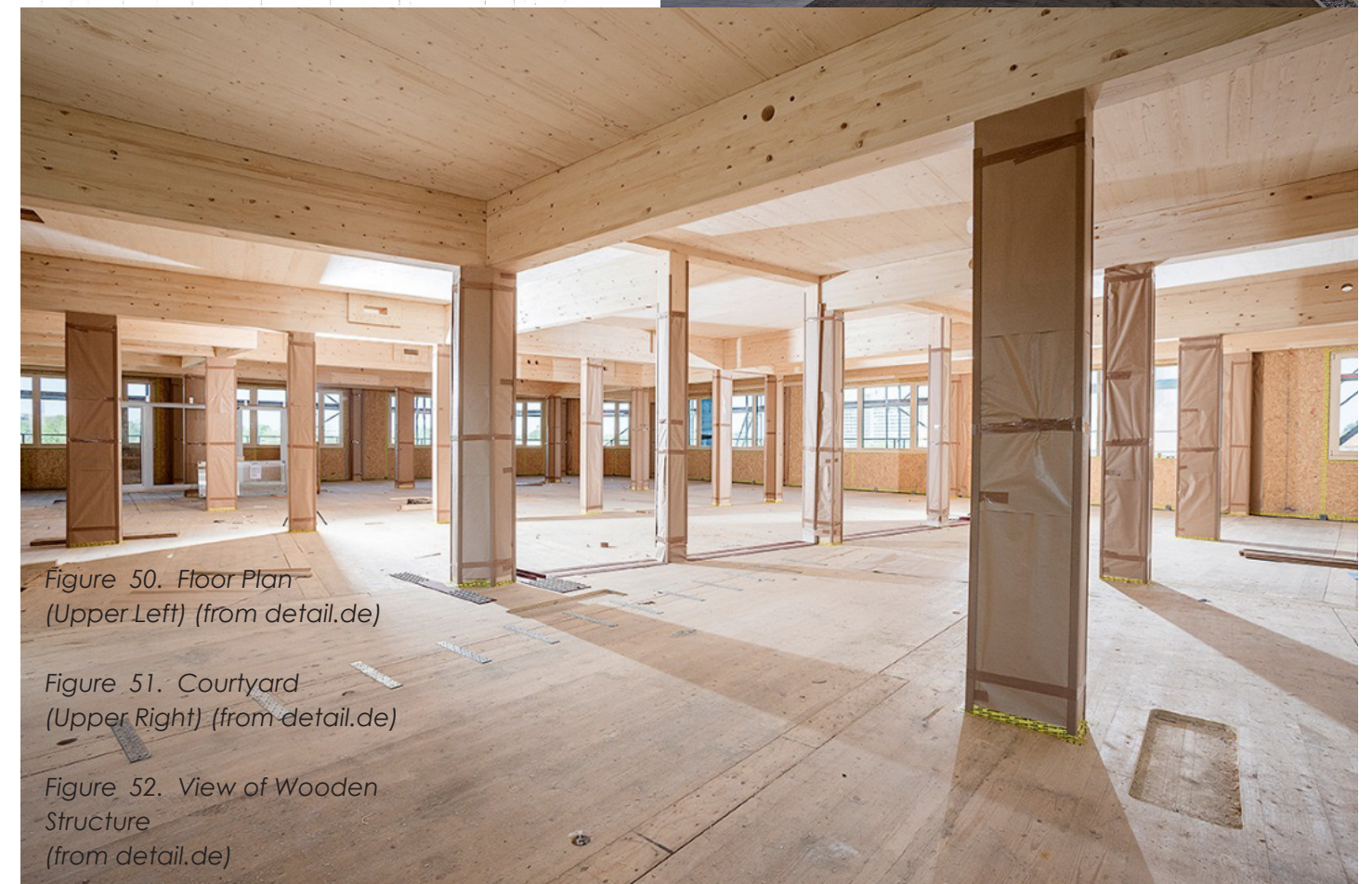


Figure 50. Floor Plan (Upper Left) (from detail.de)

Figure 51. Courtyard (Upper Right) (from detail.de)

Figure 52. View of Wooden Structure (from detail.de)

WOODTEK HQ



Figure 53. Woodtek Headquarters
(from Figure x Lee Kuo-Min Studio)

LOCATION:	TAIWAN
COMPLETION:	2014
TYPE:	OFFICE
AREA:	600 SQ. M
HEIGHT:	5 STORIES
OWNER:	WOODTEK
ARCHITECT:	ORIGIN
STRUCTURE OVERVIEW:	CLT STRUCTURE

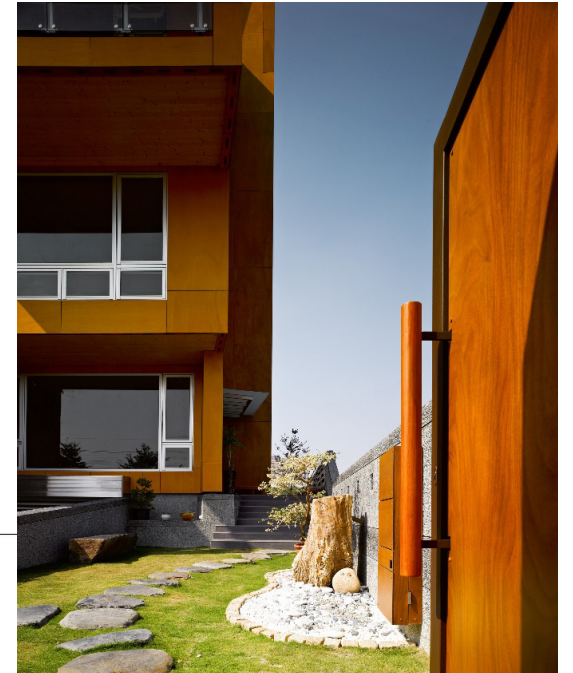
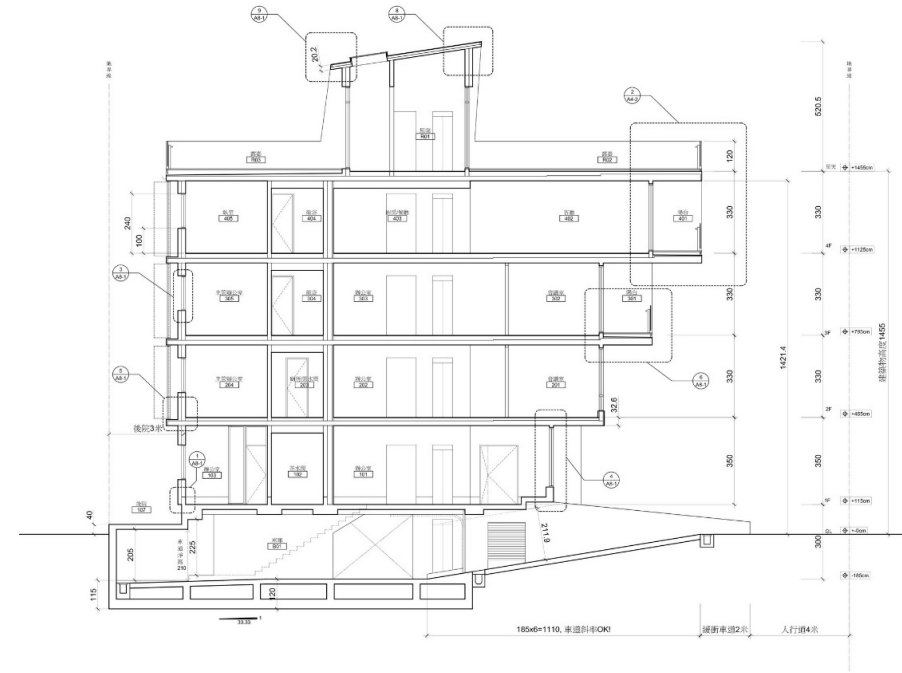


Figure 54. Section (Upper Left) (from Origin)

Figure 55. Courtyard (Upper Right) (from Figure x Lee Kuo-Min Studio)



Figure 56. Kitchen (Middle) (from Figure x Lee Kuo-Min Studio)

Figure 57. Entry (Lower Left) (from Figure x Lee Kuo-Min Studio)



Figure 58. Roof (Lower Center) (from Figure x Lee Kuo-Min Studio)

Figure 59. Stair (Lower Right) (from Figure x Lee Kuo-Min Studio)

TAMEDIA



Figure 60. Tamedia Office Building
(from Shigeru Ban Architects)

LOCATION:	ZURICH, SWITZERLAND
COMPLETION:	2013
TYPE:	OFFICE
AREA:	8,900 SQ. M
HEIGHT:	7 STORIES
OWNER:	TAMEDIA
ARCHITECT:	SHIGERU BAN ARCHITECTS
STRUCTURAL ENGINEER:	CREATION HOLZ GMBH
WOOD USE:	2,000 CU. M
STRUCTURE OVERVIEW:	GLULAM BEAMS AND COLUMNS WITH CLT FLOOR DECKS



Figure 61. View of Structure (Top) (from Shigeru Ban Architects)

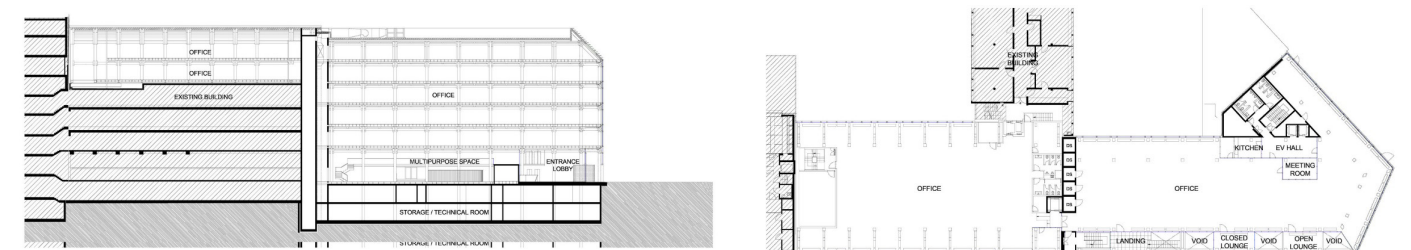
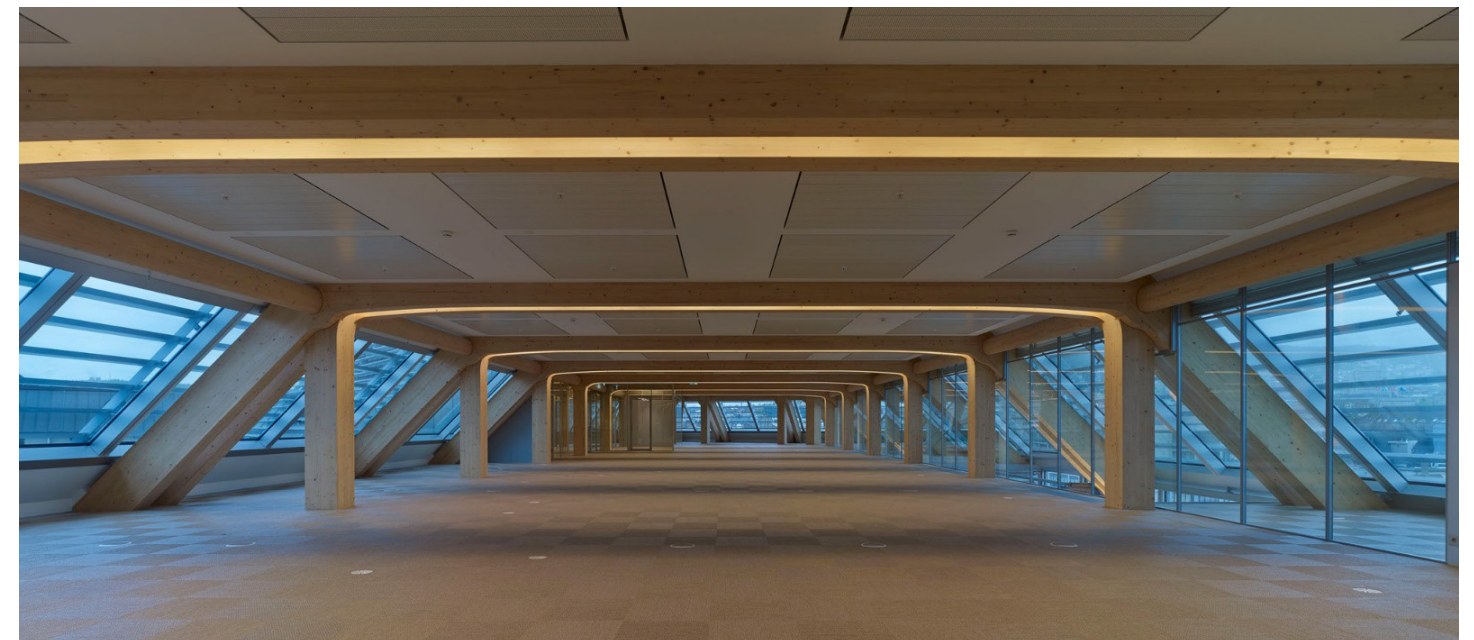


Figure 62. Section and Plan (Middle) (from Shigeru Ban Architects)

Figure 63. View of Upper Floor (Bottom) (from Shigeru Ban Architects)



C13 BUILDING



Figure 64. C13 Building (Left) (from Kaden Klingbeil)

Figure 65. C13 Building Rear Stair Detail (Right) (from Kaden Klingbeil)

LOCATION: BERLIN, GERMANY
COMPLETION: 2013
TYPE: FAMILY, EDUCATIONAL, AND HEALTH CENTER
AREA: 2,350 SQ. M
HEIGHT: 7 STORIES
OWNER: STIFTUNG FÜR CHRISTLICHE BILDUNG
ARCHITECT: KADEN KLINGBEIL ARCHITEKTEN
STRUCTURAL ENGINEER: PIRMIN JUNG
STRUCTURE OVERVIEW: STEEL BEAMS WITH CLT FLOOR DECKS, CONCRETE AND GLULAM COLUMNS, CLT PARTITION WALLS



Figure 66. Exposed CLT in Finished Structure (Left) (from Kaden Klingbeil)

Figure 67. Exterior View (Lower Left) (from Kaden Klingbeil)

Figure 68. Structure During Construction (Lower Right) (from Kaden Klingbeil)



NINA



Figure 69. NINA Headquarters
(from Pir II AS)

LOCATION: TRONDHEIM, NORWAY
COMPLETION: 2013
TYPE: OFFICE AND LABORATORY
AREA: 8,374 SQ. M
HEIGHT: 5 STORIES
OWNER: NINA
ARCHITECT: PIR II AS
STRUCTURE OVERVIEW: STEEL STRUCTURE WITH CONCRETE CORE AND 170 MM CLT DECK

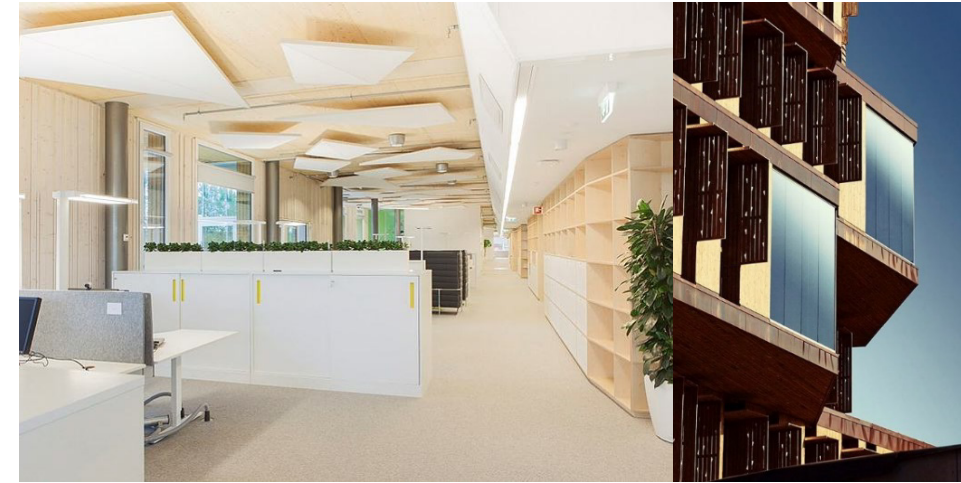
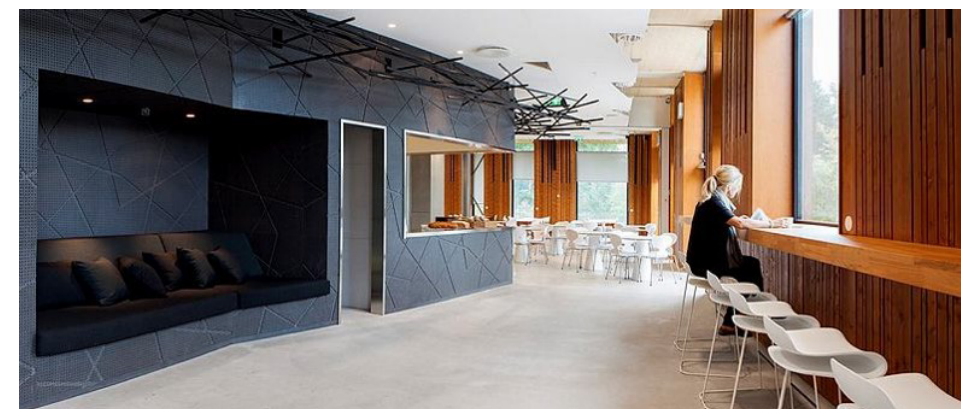
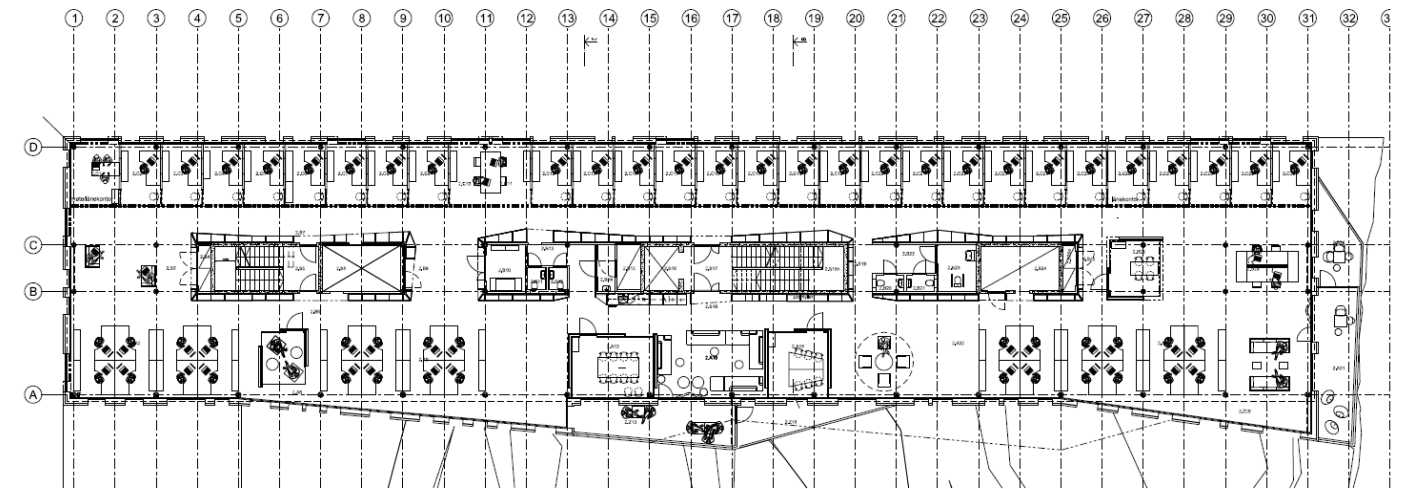


Figure 70. Office Interior
(Top Left) (from Pir II AS)

Figure 71. Exterior Panelling
Detail
(Top Right) (from Pir II AS)

Figure 72. Plan
(Middle) (from Pir II AS)

Figure 73. Cafeteria
(Bottom) (from Pir II AS)



ILLWERKE ZENTRUM MONTAFON VANDANS



Figure 74. Illwerke Zentrum Montafon Vandans (from Hermann Kaufmann)

LOCATION:	VANDANS, AUSTRIA
COMPLETION:	2013
TYPE:	OFFICE
AREA:	10,000 SQ. M
HEIGHT:	5 STORIES
OWNER:	VORARLBERGER ILLWERKE AG
ARCHITECT:	HERMANN KAUFMANN
STRUCTURAL ENGINEER:	MERZ KLEY PARTNER
STRUCTURE OVERVIEW:	HYBRID CONCRETE-CLT SLAB SYSTEM WITH CONCRETE AND STEEL COLUMNS



Figure 75. View of Exterior (Left) (from Hermann Kaufmann)



Figure 76. Plan (Middle) (from Hermann Kaufmann)

Figure 77. Office (Lower Left) (from Hermann Kaufmann)

Figure 78. Lobby Atrium (Lower Right) (from Hermann Kaufmann)



SKUTEVIKSBODER 13

X



Figure 79. Skuteviksboder 13
(from Tret teknisk)

LOCATION: BERGEN, NORWAY
COMPLETION: 2013
TYPE: OFFICE
AREA: 1,300 SQ. M
HEIGHT: 4 STORIES + LOFT
OWNER: JAN HARALD HELMICH PEDERSEN
ARCHITECT: BERTRAM BROCHMAN
WOOD USE: 334 CU. M CLT
STRUCTURE OVERVIEW: CLT WALLS AND FLOORS, WALLS ARE 95 AND 140 MM THICK, FLOORS ARE 165 MM THICK



Figure 80. Skylight Detail
(Upper Left) (from OBAS)

Figure 81. Construction
(Upper Right) (from OBAS)

Figure 82. Construction
(Lower Left) (from OBAS)

Figure 83. Panel Erection
(Lower Right) (from OBAS)

EARTH SCIENCES BUILDING



Figure 84. Earth Sciences Building (from Martin Tessler)

LOCATION:	VANCOUVER, BC, CANADA
COMPLETION:	2012
TYPE:	ACADEMIC
AREA:	15,794 SQ. M
HEIGHT:	5 STORIES
OWNER:	UNIVERSITY OF BRITISH COLUMBIA
ARCHITECT:	PERKINS AND WILL
STRUCTURAL ENGINEER:	EQUILIBRIUM CONSULTING
STRUCTURE OVERVIEW:	GLULAM BEAMS AND COLUMNS WITH CLT FLOOR DECKS IN CLASSROOM WING, CONCRETE IN LABORATORY WING

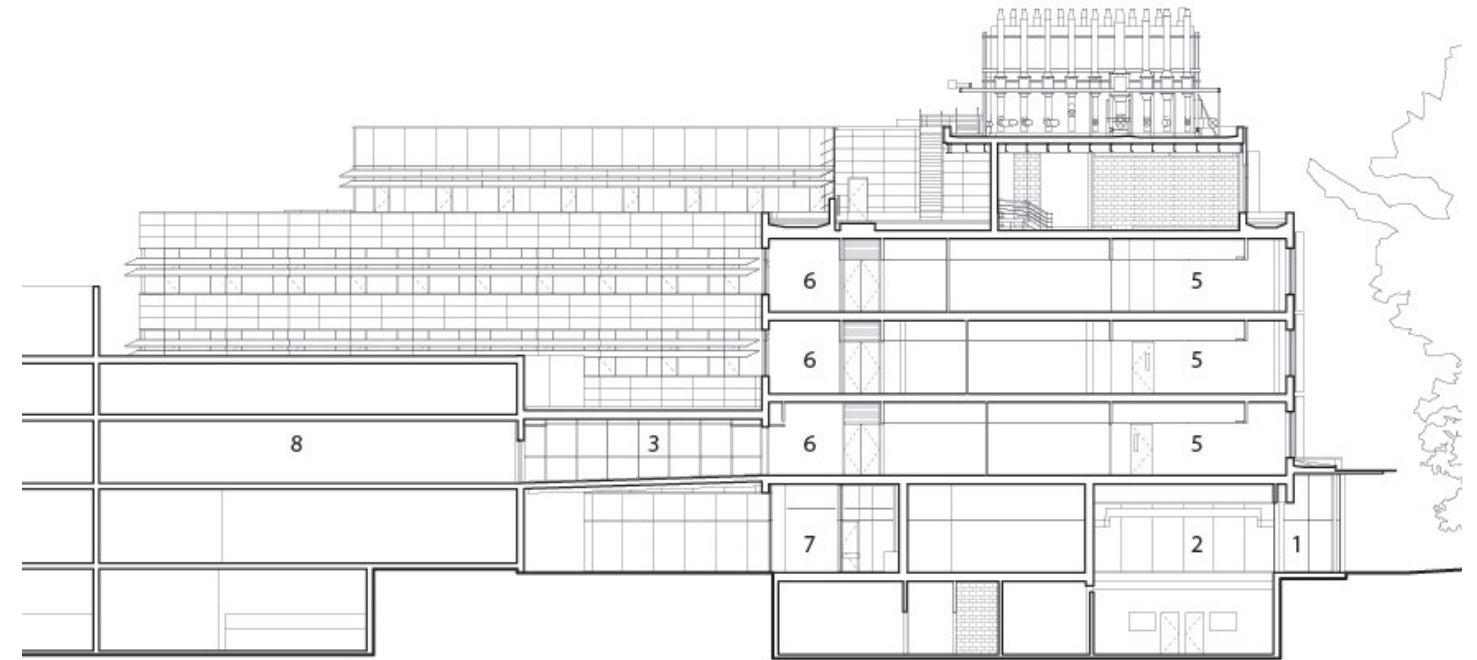


Figure 85. Section (Above) (from Perkins and Will)



Figure 86. Main Atrium (Lower Left) (from Martin Tessler)



Figure 87. View of Exterior (Lower Right) (from Martin Tessler)

MÜHLEBACHSTRASSE



Figure 88. Mühlebachstrasse
(from Kämpen Für Architektur)

LOCATION: ZÜRICH, SWITZERLAND
COMPLETION: 2012
TYPE: OFFICE
HEIGHT: 6 STORIES
ARCHITECT: KÄMPEN FÜR ARCHITEKTURE
STRUCTURE OVERVIEW: HYBRID CLT-CONCRETE FLOOR PANELS WITH CLT WALL PANELS, GLULAM COLUMNS, AND LVL BEAMS

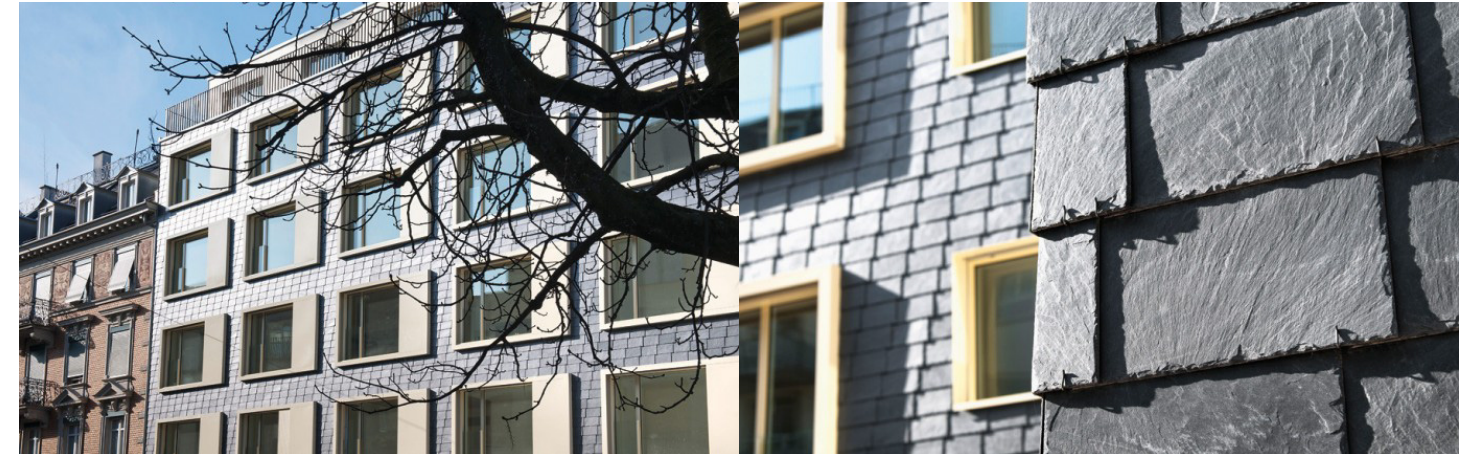


Figure 89. Street Facade
(Top Left) (from KF Arch)

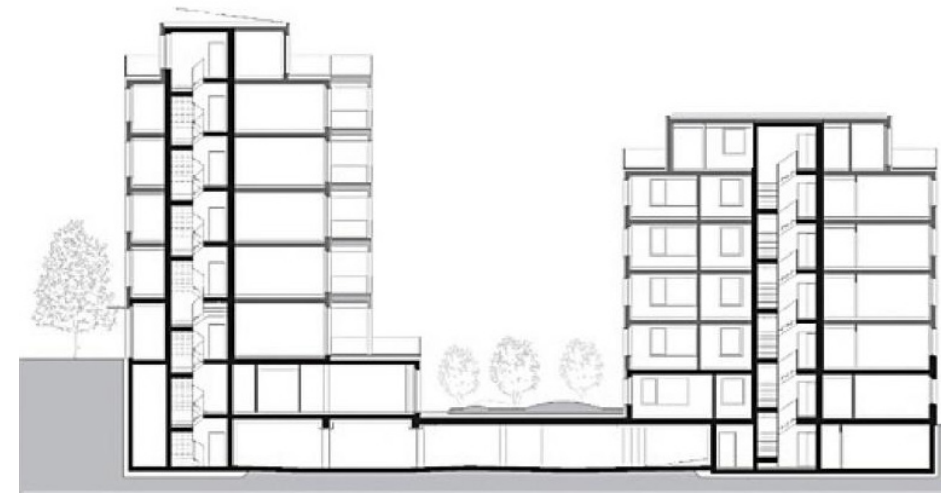


Figure 90. Siding Detail
(Top Right) (from KF Arch)

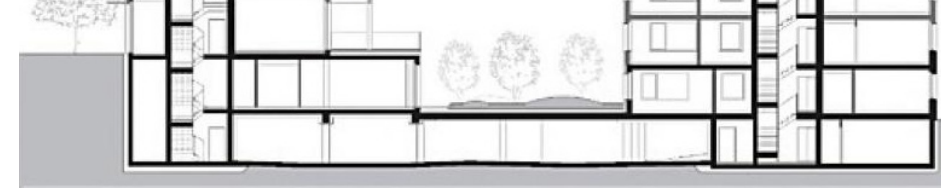


Figure 91. Section
(Left) (from KF Arch)

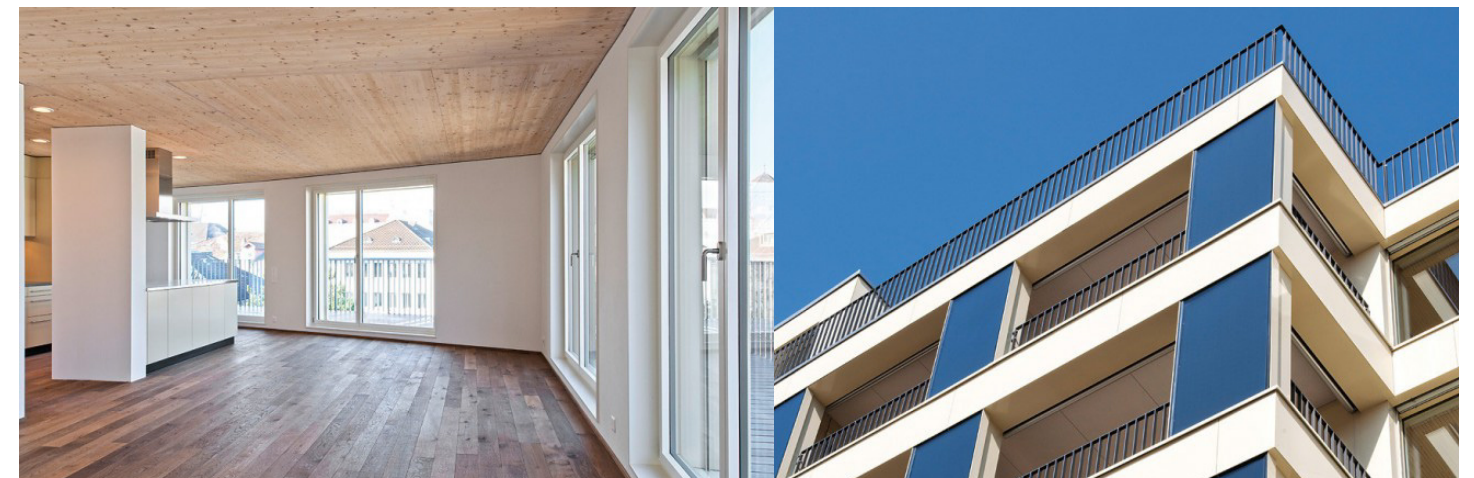


Figure 92. Interior
(Lower Left) (from KF Arch)



Figure 93. Patio Detail
(Lower Right) (from KF Arch)

LCT ONE



Figure 94. LCT One
(from Hermann Kaufmann)

LOCATION:	DORNBIRN, AUSTRIA
COMPLETION:	2012
TYPE:	OFFICE
AREA:	2,319 SQ. M
HEIGHT:	8 STORIES
OWNER:	CREE GMBH
ARCHITECT:	HERMANN KAUFMANN ZT GMBH
STRUCTURAL ENGINEER:	ARUP
STRUCTURE OVERVIEW:	PRIMARY STRUCTURAL SYSTEM GLULAM TIMBER AND CONCRETE, HYBRID GLULAM- CONCRETE FLOOR DECK WITH CONCRETE CORE



Figure 95. Exterior Elevation
(Upper Left) (from Hermann
Kaufmann)

Figure 96. Office
(Upper Right) (from Hermann
Kaufmann)

Figure 97. Plan
(Middle) (from Hermann
Kaufmann)



Figure 98. Concrete-Timber
Composite Floor Deck Erection
(Lower Left) (from Hermann
Kaufmann)

Figure 99. Interior
(Lower Right) (from Hermann
Kaufmann)



ÉDIFICE FONDACTION QUÉBEC

xiv



Figure 100. Édifice Fondation Québec (from GHA)

LOCATION:	QUÉBEC, FRANCE
COMPLETION:	2010
TYPE:	OFFICE
AREA:	6,000 SQ. M
HEIGHT:	6 STORIES
OWNER:	FONDACTION CSN
ARCHITECT:	GILLES HUOT ARCHITECTE
STRUCTURAL ENGINEER:	BES
STRUCTURE OVERVIEW:	ALL TIMBER STRUCTURE WITH GLULAM BEAMS AND COLUMNS AND TIMBER DECKING



Figure 101. Conference Room (Above) (from GHA)



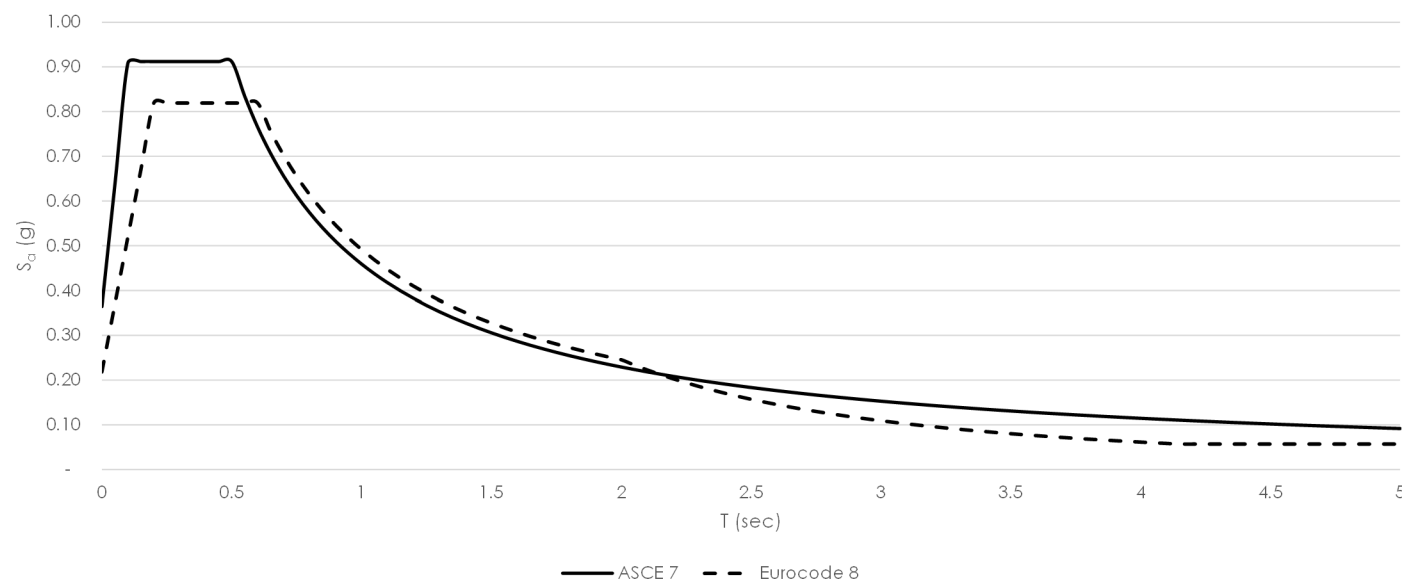
Figure 102. Office Corridor (Left) (from GHA)

SEISMIC CODE COMPARISON

This section contains a comparison of the calculation of base shears using the equivalent lateral force method for a simplified prototypical building in Seattle, Washington. Code provisions in the United States (ASCE 7-10^[1]) and Europe (Eurocode 8^[2]) are compared to determine if there are significant differences between the codes making the structural design of tall timber buildings more favorable on one continent as compared to the other. The key finding is that results for this case from the two building codes are fairly similar, the building designed to Eurocode 8 has an 11 percent reduction in base shear as compared to the building designed to ASCE 7. A description of key differences in the codes as well as a discussion of challenges follows.

The first challenge to comparing the codes was in defining a consistent seismic hazard. While ASCE 7 defines the response spectra by specifying short period and one second spectral accelerations which are then used to scale the response spectrum curve, Eurocode 8 uses the peak ground acceleration to define the response spectrum curve. For this case all values are taken from USGS data^{[3][4]}. Figure 103 shows the elastic design response spectra for the seismic hazard in Seattle for both codes. Under Eurocode 8, short period structures see a ten percent reduction in accelerations as compared to ASCE 7. In this case, since estimated periods fall outside this range (ASCE 7 - 0.75 sec, Eurocode 8 - 0.77 sec) the spectral acceleration should not be a primary point of variation between the two codes.

Figure 103. Elastic Design Response Spectra



ASCE 7-10			Eurocode 1998		
General Parameters					
Parameter	Value	Reference	Parameter	Value	Reference
Risk Category	II	[T1.5-1]			
l_e	1	[T1.5-2]			
Number of Stories	11				
Ground Floor Height	4.6 m				
Typical Floor to Floor	3.4 m				
Building Height	38.1 m		H	38.1 m	
Floor Area	557 m ²				
Total Area	6132 m ²				
Seismic Parameters					
Parameter	Value	Reference	Parameter	Value	Reference
Site Class	C	[11.4.2]	Ground Type	C	[T3.1]
S_s	1.37 g		S	1.15	[T3.2]
S_1	0.53 g		Y_1	1	[3.2.2.1]
S_{DS}	0.91 g		a_{gR}	0.29 g	[3.2.2.1]
S_{D1}	0.46 g		a_g	2.80 g	[3.2.2.1]
T_0	0.10 s		T_B	0.2 s	[T3.2]
T_s	0.50 s		T_C	0.6 s	[T3.2]
T_L	6 s		T_D	2 s	[T3.2]
F_a	1	[T11.4-2]	ξ	5%	[3.2.2.2]
F_v	1.3	[T11.4-2]	η	1	[EQ3.6]
Seismic Design Category	D	[T11.6-1]	β	0.2	[3.2.2.5]
F_{PGA}	1		S_d	2.10 g	[3.2.2.5]
Structural Parameters					
Parameter	Value	Reference	Parameter	Value	Reference
System Type	CLT				
R	3	[11.4.2]	q	3	[T8.1]
C_d	2		λ	0.85	[4.3.3.2.2]
Ω_0	1.4				
C_t	0.0488	[T12.8-2]	C_t	0.05	[4.3.3.2.2]
α	0.75				
T_a	0.75 s	[12.8.2.1]	T_1	0.77 s	[EQ4.6]
Equivalent Lateral Force Procedure					
Parameter	Value	Reference	Parameter	Value	Reference
C_s	0.204	[12.8.1.1]			
W	18,195 kN		m	1,854,700 kg	
V	3,720 kN		F_b	3,305 kN	[EQ4.5]

Table 5. Seismic Design Calculations (yellow values indicate input parameters, white values indicate calculated parameters)

Table 5 shows the calculations for both codes, values that are related between the two codes are shown on the same line. Both codes use an adjustment factor to implicitly account for energy dissipation through inelastic deformation. While CLT shear walls are not currently a code accepted system and therefore R, C_d , and Ω_0 values are not provided in the code, Eurocode 8 gives a q-value of three for timber structures with a high capacity to dissipate energy. This value has also been suggested as an appropriate R-value based on preliminary research. Therefore to maintain consistency, R and q are taken equal to three for purposes of this comparison. This value is not a given and has a linear effect on the total base shear of the building, significantly impacting the structural design.

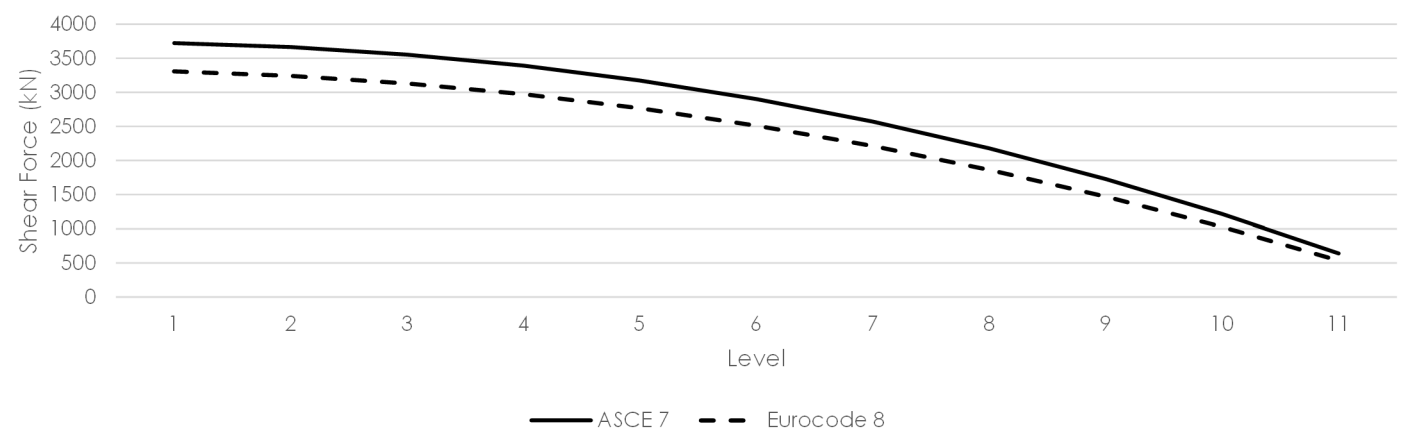


Figure 104. Shear force by level

Based on the analysis, the total base shear for the building designed to ASCE 7 is 3,720 kN and to Eurocode 8 is 3,305 kN. The total shear force by level is shown in Figure 104, both codes provide a similar procedure for distributing shear force by seismic mass at a level therefore producing similar distributions of force over the building.

While based on this analysis both codes provide similar results suggesting that neither code provides significantly more favorable conditions for the design of timber buildings in seismic regions, several key assumptions were made could significantly alter this conclusion.

The first is code acceptance of CLT shear wall systems. While Eurocode explicitly allows the use of CLT as a lateral force resisting system, ASCE 7 does not include CLT shear walls as a code approved system. This means that any buildings using CLT shear walls must prove code equivalent performance and are subject to increased testing, more extensive design procedures, and peer review. This significantly increases the cost of design and discourages the use of CLT as a lateral force resisting system. Future inclusion of CLT shear wall systems will considerably mitigate many of the current challenges to design.

The second assumption is related to R- and q-values. While a q-value of three is currently permitted based on Eurocode provisions, no R-value is currently defined in ASCE 7. While this is an area of ongoing research, comparison of R- and q-values for other systems shows that the two codes do not always adopt similar values and this can have a significant effect on base shear calculated using the two codes. Since preliminary research has shown good inelastic behavior for CLT systems^[5], an R-value of three does not seem unreasonable; however, establishment of these values will be a key step in the development of CLT lateral systems.

REFERENCES

- [1] American Society of Civil Engineers. (2010). ASCE 7-10: Minimum design loads for buildings and other structures. Reston, VA: American Society of Civil Engineers.
- [2] European Committee for Standardization. (2004). Eurocode 8: Design of structures for earthquake resistance. Brussels, Belgium: European Committee for Standardization.
- [3] US Department of the Interior, & US Geological Survey. (2013). 2008 interactive deaggregations. Retrieved from <http://geohazards.usgs.gov/deaggint/2008/>
- [4] US Department of the Interior, & US Geological Survey. (2014). US seismic design maps. Retrieved from <http://earthquake.usgs.gov/designmaps/us/application.php?>
- [5] Van de Lindt, John W., Rammer, D., Popovski, M., Line, P., Pei, S., & Pryor, S. E. (2013). Lateral design of cross-laminated timber buildings. In R. O. Hamburger, J. D. Dolan, E. Karacabeyli, B. Douglas, T. D. Skaggs & A. Ceccotti (Eds.), CLT handbook (US ed., pp. 119). Pointe-Claire, QC: FP Innovations.

CUSTOM SCRIPT COMPONENTS

C

Appendix C contains a copy of custom C# scripted components written for grasshopper. A series of components were used to build the geometry in grasshopper and to perform the structural design of timber elements. For access to full grasshopper algorithm, contact author.

```

1  using Rhino.Geometry;
2  using Grasshopper.Kernel;
3  using System;
4  using System.Linq;
5  using System.Collections.Generic;
6
7  namespace Geometry3D
8  {
9      public class Geometry3DComponent : GH_Component
10     {
11         /// LOCATES COMPONENT IN GRASSHOPPER PANEL
12         public Geometry3DComponent()
13             : base("Geometry3D", "G3D",
14                 "Build Frame and Lay Out Shear Walls",
15                 "Thesis", "Geometry")
16         {
17         }
18
19         /// REGISTER INPUT PARAMETERS
20         protected override void
21         RegisterInputParams(GH_Component.GH_InputParamManager pManager)
22         {
23             pManager.AddNumberParameter("nx", "nx", "Number of X
24             Gridlines", GH_ParamAccess.item, 2);
25             pManager.AddNumberParameter("ny", "ny", "Number of Y
26             Gridlines", GH_ParamAccess.item, 2);
27             pManager.AddNumberParameter("nz", "nz", "Number of Levels",
28             GH_ParamAccess.item, 1);
29             pManager.AddNumberParameter("sx", "sx", "X-Bay Length",
30             GH_ParamAccess.item, 20.0);
31             pManager.AddNumberParameter("sy", "sy", "Y-Bay Length",
32             GH_ParamAccess.item, 20.0);
33             pManager.AddNumberParameter("sz0", "sz0", "Ground Level Floor
34             to Floor Height", GH_ParamAccess.item, 15.0);
35             pManager.AddNumberParameter("sz1", "sz1", "Typical Floor to
36             Floor Height", GH_ParamAccess.item, 11.0);
37             pManager.AddNumberParameter("cx", "cx", "Core X-Dimension",
38             GH_ParamAccess.item, 20.0);
39             pManager.AddNumberParameter("cy", "cy", "Core Y-Dimension",
40             GH_ParamAccess.item, 20.0);
41             pManager.AddNumberParameter("cox", "cox", "Core Offset X",
42             GH_ParamAccess.item, 10.0);
43             pManager.AddNumberParameter("coy", "coy", "Core Offset Y",
44             GH_ParamAccess.item, 10.0);
45             pManager.AddNumberParameter("nCoreInt", "nCoreInt", "Number of
46             Interior Core Walls", GH_ParamAccess.item, 4);
47             pManager.AddNumberParameter("tWall", "tWall", "CLT Shear Wall
48             Thickness (in)", GH_ParamAccess.item, 12);
49             pManager.AddNumberParameter("Veq", "Veq", "Seismic Base Shear",
50             GH_ParamAccess.item, 1000);
51             pManager.AddNumberParameter("Cd", "Cd", "ASCE 7 Cd Parameter",
52             GH_ParamAccess.item, 3.2);
53             pManager.AddNumberParameter("k", "k", "ASCE 7 k Parameter",
54             GH_ParamAccess.item, 1.2);
55             pManager.AddNumberParameter("EqMassTyp", "EqMassTyp",
56             "Earthquake Mass on Typical Floor", GH_ParamAccess.item, 75);
57             pManager.AddNumberParameter("EqMassRoof", "EqMassRoof",
58             "Earthquake Mass on Roof", GH_ParamAccess.item, 50);
59         }
60     }
61 }

```

```

40     pManager.AddNumberParameter("LSeg", "LSeg", "Length of CLT
41     Shear Walls at Facade", GH_ParamAccess.item, 10);
42 }
43
44 /// REGISTER OUTPUT PARAMETERS
45 protected override void
46 RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
47 {
48     pManager.AddCurveParameter("X Beams", "BX", "Beams in
49     X-Direction", GH_ParamAccess.list);
50     pManager.AddCurveParameter("Y Beams", "BY", "Beams in
51     Y-Direction", GH_ParamAccess.list);
52     pManager.AddCurveParameter("columns", "C", "Columns",
53     GH_ParamAccess.list);
54     pManager.AddMeshParameter("FloorSlabs", "FloorSlabs", "Floor
55     Slab Meshes", GH_ParamAccess.list);
56     pManager.AddMeshParameter("RoofSlab", "RoofSlab", "Roof Slab
57     Meshes", GH_ParamAccess.item);
58     pManager.AddMeshParameter("CLTWallCore", "CLTWallCore", "CLT
59     Core Walls", GH_ParamAccess.list);
60     pManager.AddMeshParameter("CLTWallFac", "CLTWallFac", "CLT
61     Facade Walls", GH_ParamAccess.list);
62     pManager.AddCurveParameter("CLTWallLoc", "CLTWallLoc", "Lines
63     with Location of All CLT Walls", GH_ParamAccess.list);
64     pManager.AddCurveParameter("WallCol", "WallCol", "Columns at
65     CLT Walls", GH_ParamAccess.list);
66     pManager.AddNumberParameter("VWall", "VWall", "Base Shear in
67     Wall (K)", GH_ParamAccess.list);
68     pManager.AddNumberParameter("MWall", "MWall", "Moment at base
69     of Wall (K-ft)", GH_ParamAccess.list);
70 }
71
72 protected override void SolveInstance(IGH_DataAccess DA)
73 {
74     ///-----
75     ///-----
76     /// RETRIEVE INPUT VARIABLES
77     ///
78     ///-----
79     ///
80     ///-----
81
82     double nx = 0;
83     double ny = 0;
84     double nz = 0;
85     double sx = 0.0;
86     double sy = 0.0;
87     double sz0 = 0.0;
88     double sz1 = 0.0;
89     double cx = 0.0;
90     double cy = 0.0;
91     double cox = 0.0;
92     double coy = 0.0;

```

```

77     double nIntCore = 4;
78     double tWall = 12;
79     double baseshear = 1000;
80     double Cd = 3.2;
81     double ASCEkEq = 1.2;
82     double EMassTyp = 75;
83     double EMassRoof = 30;
84     double LSeg = 10;
85
86     if (!DA.GetData(0, ref nx)) return;
87     if (!DA.GetData(1, ref ny)) return;
88     if (!DA.GetData(2, ref nz)) return;
89     if (!DA.GetData(3, ref sx)) return;
90     if (!DA.GetData(4, ref sy)) return;
91     if (!DA.GetData(5, ref sz0)) return;
92     if (!DA.GetData(6, ref sz1)) return;
93     if (!DA.GetData(7, ref cx)) return;
94     if (!DA.GetData(8, ref cy)) return;
95     if (!DA.GetData(9, ref cox)) return;
96     if (!DA.GetData(10, ref coy)) return;
97     if (!DA.GetData(11, ref nIntCore)) return;
98     if (!DA.GetData(12, ref tWall)) return;
99     if (!DA.GetData(13, ref baseshear)) return;
100    if (!DA.GetData(14, ref Cd)) return;
101    if (!DA.GetData(15, ref ASCEkEq)) return;
102    if (!DA.GetData(16, ref EMassTyp)) return;
103    if (!DA.GetData(17, ref EMassRoof)) return;
104    if (!DA.GetData(18, ref LSeg)) return;
105
106    ///-----
107
108    ///-----
109    ///INITIALIZE VARIABLES
110
111    ///-----
112
113    List<Mesh> floorMesh = new List<Mesh>(); //OUTPUT LIST WITH
114    FLOOR MESHES
115    Mesh roofMesh = new Mesh(); //OUTPUT LIST WITH ROOF MESH
116    List<Line> xCoreWalls = new List<Line>();
117    List<Line> yCoreWalls = new List<Line>();
118    List<Line> xFacWalls = new List<Line>();
119    List<Line> yFacWalls = new List<Line>();
120
121    List<double> height = GetHeights(nz, sz0, sz1); //LIST OF FLOOR
122    LOCATIONS
123
124    double XDim = (nx - 1) * sx; //BUILDING X-DIMENSION (FT)
125    double YDim = (ny - 1) * sy; //BUILDING Y-DIMENSION (FT)
126
127    double kappa = 0.833; //TIMOSHENKO SHEAR COEFFICIENT

```

```

125     double G = 50000; //SHEAR MODULUS (psi)
126     double E = 1800000; //MODULUS OF ELASTICITY (psi)
127     double Cm = 1; //WET SERVICE FACTOR
128     double Ct = 1; //TEMPERATURE FACTOR
129     double phiComp = 0.9;
130     double KComp = 2.4; //FORMAT CONVERSION FACTOR
131     double phiShear = 0.75;
132     double KShear = 2.88; //FORMAT CONVERSION FACTOR
133     double lambda = 1; //TIME EFFECT FACTOR
134     double Ie = 1; //EARTHQUAKE IMPORTANCE FACTOR
135     double DriftLimit = 0.02; //2% Max Story Drift
136     double Fc, Vr;
137
138     double LWallX, LWallY;
139     double XCOG, YCOG;
140     double XOff, YOff;
141     double Stiff1X, Stiff1Y;
142     double StiffTypX, StiffTypY;
143     double kXoffX = 0;
144     double kYoffY = 0;
145     double Inertia;
146     string WallDir;
147
148     double XCOR, YCOR; //CENTER OF ROTATION
149     double AccTorX, AccTorY; //Accidental Torsion
150
151     List<double> ListWallShear = new List<double>();
152     List<double> ListWallMoment = new List<double>();
153     List<Line> AllWalls = new List<Line>();
154
155     //WALL PROPERTIES BY THICKNESS FROM STRUCTURLAM
156     if (tWall == 3.9)
157     {
158         Fc = 1150; //psi
159         Vr = 2906; //lb/ft
160     }
161     else if (tWall == 6.66)
162     {
163         Fc = 1150; //psi
164         Vr = 5812; //lb/ft
165     }
166     else if (tWall == 9.42)
167     {
168         Fc = 1150; //psi
169         Vr = 8718; //lb/ft
170     }
171     else
172     {
173         Fc = 1150; //psi
174         Vr = 11624; //lb/ft
175     }
176
177     //TEMPORARY VARIABLES
178     Mesh tempMesh;
179     Point3d tempPt1, tempPt2, tempPt3, tempPt4;
180     double temp1, temp2;
181     List<double> temp3 = new List<double>();
182     double Fail = 0;

```

```

183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222

```

```

//-----
//-----
//BUILD FLOOR SLABS
//-----
//-----
//-----
for (int i = 0; i < nz; i++)
{
    tempMesh = BuildMesh(new Point3d(0, 0, height[i + 1]), new
    Point3d(0, YDim, height[i + 1]), new Point3d(XDim, YDim,
    height[i + 1]), new Point3d(XDim, 0, height[i + 1]));
    if (i != (nz - 1))
    {
        floorMesh.Add(tempMesh);
    }
    else
    {
        roofMesh = tempMesh;
    }
}
AreaMassProperties amp = AreaMassProperties.Compute(roofMesh);
Point3d Centroid = amp.Centroid;

//-----
//-----
//DESIGN SHEAR WALLS
//-----
//-----
//DRAW PERIMETER CORE WALLS
tempPt1 = new Point3d(cox, coy, 0);
tempPt2 = new Point3d(cox + cx, coy, 0);
tempPt3 = new Point3d(cox, coy + cy, 0);
tempPt4 = new Point3d(cox + cx, coy + cy, 0);

xCoreWalls.Add(new Line(tempPt1, tempPt2));
xCoreWalls.Add(new Line(tempPt3, tempPt4));
yCoreWalls.Add(new Line(tempPt1, tempPt3));
yCoreWalls.Add(new Line(tempPt2, tempPt4));

//DRAW INTERIOR CORE WALLS
double gap = cy / (nIntCore + 1);

```

```

223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279

```

```

for (int i = 0; i < nIntCore; i++)
{
    tempPt1 = new Point3d(cox, coy + (1 + i) * gap, 0);
    tempPt2 = new Point3d(cox + cx, coy + (1 + i) * gap, 0);
    xCoreWalls.Add(new Line(tempPt1, tempPt2));
}

//CALCULATE STORY SHEARS TO FIND MOMENT ARM FOR SEISMIC FORCE
SeismicForces EqForce = new SeismicForces();
EqForce.CalculateSeismicForces(height, EMassTyp, EMassRoof,
amp.Area, baseshear, ASCEkEq);

//CHECK WALLS AND ADD PERIMETER WALLS UNTIL WALLS PASS
double nXfacWalls = 0;
double nYfacWalls = 0;
int loopCount = 0;

while (Fail < 3)
{
    AllWalls = new List<Line>();
    ListWallShear = new List<double>();
    ListWallMoment = new List<double>();

    List<double> ListLWallX = new List<double>();
    List<double> ListLWallY = new List<double>();
    List<double> ListXCOG = new List<double>();
    List<double> ListYCOG = new List<double>();
    List<double> ListXOff = new List<double>();
    List<double> ListYOff = new List<double>();
    List<double> ListStiff1X = new List<double>();
    List<double> ListStiff1Y = new List<double>();
    List<double> ListStiffTypX = new List<double>();
    List<double> ListStiffTypY = new List<double>();
    List<string> ListWallDir = new List<string>();

    foreach (Line v in xCoreWalls)
    {
        AllWalls.Add(v);
    }
    foreach (Line v in yCoreWalls)
    {
        AllWalls.Add(v);
    }
    foreach (Line v in xFacWalls)
    {
        AllWalls.Add(v);
    }
    foreach (Line v in yFacWalls)
    {
        AllWalls.Add(v);
    }

    for (int i = 0; i < AllWalls.Count; i++)
    {
        //FIND WALL LENGTH FROM ENDPOINTS
        LWallX = Math.Abs(AllWalls[i].ToX - AllWalls[i].FromX);
    }
}

```

```

280 //ft
281 ListLWallX.Add(LWallX); //ft
282 LWallY = Math.Abs(AllWalls[i].ToY - AllWalls[i].FromY);
283 //ft
284 ListLWallY.Add(LWallY); //ft
285 //FIND WALL CENTER OF GRAVITY
286 XCOG = LWallX / 2 + AllWalls[i].FromX; //ft
287 ListXCOG.Add(XCOG); //ft
288 YCOG = LWallY / 2 + AllWalls[i].FromY; //ft
289 ListYCOG.Add(YCOG); //ft
290 //FIND WALL OFFSET FROM BUILDING CENTER OF MASS
291 XOff = XCOG - Centroid.X; //ft
292 ListXOff.Add(XOff); //ft
293 YOff = YCOG - Centroid.Y; //ft
294 ListYOff.Add(YOff); //ft
295 //FIND DIRECTION OF WALL
296 if (LWallX < 0.1)
297 {
298     WallDir = "y";
299 }
300 else
301 {
302     WallDir = "x";
303 }
304 ListWallDir.Add(WallDir);
305 //CALCULATE WALL STIFFNESS
306 if (WallDir == "x")
307 {
308     Stiff1Y = 0;
309     StiffTypY = 0;
310     Inertia = tWall * Math.Pow(LWallX * 12, 3) / 12;
311     //in4
312     Stiff1X = 1 / ((Math.Pow(sz0 * 12, 3) / (3 * E *
313     Inertia)) + (sz0 * 12 / (kappa * tWall * LWallX *
314     12 * G))); //lb/in
315     StiffTypX = 1 / ((Math.Pow(sz1 * 12, 3) / (3 * E *
316     Inertia)) + (sz1 * 12 / (kappa * tWall * LWallX *
317     12 * G))); //lb/in
318 }
319 else
320 {
321     Stiff1X = 0;
322     StiffTypX = 0;
323     Inertia = tWall * Math.Pow(LWallY * 12, 3) / 12;
324     //in4
325     temp1 = Math.Pow(sz0 * 12, 3) / (3 * E * Inertia);
326     temp2 = sz0 * 12 / (kappa * tWall * LWallY * 12 * G);
327     Stiff1Y = 1 / (temp1 + temp2); //lb/in
328 }
329 ListStiff1Y.Add(Stiff1Y);

```

```

330 ListStiffTypY.Add(StiffTypY);
331 ListStiff1X.Add(Stiff1X);
332 ListStiffTypX.Add(StiffTypX);
333
334 kXoffX = kXoffX + Stiff1X * XOff * 12; //lb
335 kYoffY = kYoffY + Stiff1Y * YOff * 12; //lb
336 } //END LOOP OVER EACH WALL
337
338 XCOR = kXoffX / (12 * ListStiff1X.Sum()); //ft
339 YCOR = kYoffY / (12 * ListStiff1Y.Sum()); //ft
340 AccTorX = Math.Max(0.05, Math.Abs(XCOR) / XDim);
341 AccTorY = Math.Max(0.05, Math.Abs(YCOR) / YDim);
342
343 //CHECK WALLS FOR FORCES
344 double WallShear, WallMoment;
345 double ShearPerLengthX, ShearPerLengthY, BendingStressX,
346 BendingStressY;
347 double SWall; //Section Modulus
348 List<double> ShearDCX = new List<double>();
349 List<double> ShearDCY = new List<double>();
350 List<double> BendDCX = new List<double>();
351 List<double> BendDCY = new List<double>();
352 List<int> WallPassX = new List<int>();
353 List<int> WallPassY = new List<int>();
354
355 for (int i = 0; i < AllWalls.Count; i++)
356 {
357     if (ListWallDir[i] == "x")
358     {
359         WallShear = baseshear * 1000 * ListStiff1X[i] * (1
360         + AccTorX * 2) / ListStiff1X.Sum(); //lb
361         ListWallShear.Add(WallShear / 1000);
362         WallMoment = WallShear * EqForce.MomentArm * 12;
363         //lb-in
364         ListWallMoment.Add(WallMoment / (1000 * 12));
365         ShearPerLengthX = WallShear / ListLWallX[i]; //lb/ft
366         SWall = tWall * Math.Pow((ListLWallX[i] * 12), 2) /
367         6; //in^3
368         BendingStressX = WallMoment / SWall; //psi
369
370         ShearDCX.Add(ShearPerLengthX / (phiShear * KShear *
371         Vr * Cm * Ct * lambda));
372         BendDCX.Add(BendingStressX / (phiComp * KComp * Fc
373         * Cm * Ct * lambda));
374         if (ShearDCX.Max() > 1)
375         {
376             WallPassX.Add(0); //WALL FAILS
377         }
378         else
379         {
380             if (BendDCX.Max() > 1)
381             {
382                 WallPassX.Add(0); //WALL FAILS
383             }
384             else
385             {
386                 WallPassX.Add(1); //WALL PASSES
387             }
388         }
389     }
390 }

```

```

382     }
383 }
384 else
385 {
386     WallShear = baseshear * 1000 * ListStiff1Y[i] * (1
+ AccTorY * 2) / ListStiff1Y.Sum(); //lb
387     ListWallShear.Add(WallShear / 1000);
388     WallMoment = WallShear * EqForce.MomentArm * 12;
//lb-in
389     ListWallMoment.Add(WallMoment / (1000 * 12));
390     ShearPerLengthY = WallShear / ListLWallyY[i]; //lb/ft
391     SWall = tWall * Math.Pow(ListLWallyY[i] * 12, 2) /
6; //in^3
392     BendingStressY = WallMoment / SWall; //psi
393
394     ShearDCY.Add(ShearPerLengthY / (phiShear * KShear *
Vr * Cm * Ct * lambda));
395     BendDCY.Add(BendingStressY / (phiComp * KComp * Fc
* Cm * Ct * lambda));
396     temp3.Add(ShearDCX.Max());
397     temp3.Add(ShearDCY.Max());
398     temp3.Add(BendDCX.Max());
399     temp3.Add(BendDCY.Max());
400     if (ShearDCY.Max() > 1)
401     {
402         WallPassY.Add(0); //WALL FAILS
403     }
404     else
405     {
406         if (BendDCY.Max() > 1)
407         {
408             WallPassY.Add(0); //WALL FAILS
409         }
410         else
411         {
412             WallPassY.Add(1); //WALL PASSES
413         }
414     }
415 }
416 } //END LOOP OVER EACH WALL
417
418 //LOOP OVER FLOORS TO CHECK DRIFTS
419 List<double> StoryDispX = new List<double>();
420 List<double> StoryDispY = new List<double>();
421 List<double> StoryDriftX = new List<double>();
422 List<double> StoryDriftY = new List<double>();
423 List<double> DispX = new List<double>();
424 List<double> DispY = new List<double>();
425 List<double> DriftX = new List<double>();
426 List<double> DriftY = new List<double>();
427
428 for (int i = 0; i < nz; i++)
429 {
430     if (i == 0)
431     {
432         //FIRST FLOOR
433         StoryDispX.Add(EqForce.StoryShear[i] * 1000 * (1 +
2 * AccTorX) * Cd / (ListStiff1X.Sum() * Ie));

```

```

434     StoryDispY.Add(EqForce.StoryShear[i] * 1000 * (1 +
2 * AccTorY) * Cd / ListStiff1Y.Sum() * Ie);
435     StoryDriftX.Add(StoryDispX[i] / (sz0 * 12));
436     StoryDriftY.Add(StoryDispY[i] / (sz0 * 12));
437     DispX.Add(StoryDispX[i]);
438     DispY.Add(StoryDispY[i]);
439     DriftX.Add(StoryDriftX[i]);
440     DriftY.Add(StoryDriftY[i]);
441 }
442 else
443 {
444     StoryDispX.Add(EqForce.StoryShear[i] * 1000 * (1 +
2 * AccTorX) * Cd / ListStiffTypX.Sum() * Ie);
445     StoryDispY.Add(EqForce.StoryShear[i] * 1000 * (1 +
2 * AccTorY) * Cd / ListStiffTypY.Sum() * Ie);
446     StoryDriftX.Add(StoryDispX[i] / (sz1 * 12));
447     StoryDriftY.Add(StoryDispY[i] / (sz1 * 12));
448     DispX.Add(StoryDispX[i] + DispX[i - 1]);
449     DispY.Add(StoryDispY[i] + DispY[i - 1]);
450     DriftX.Add(DispX[i] / (height[i + 1] * 12));
451     DriftY.Add(DispY[i] / (height[i + 1] * 12));
452 }
453 }
454
455 double p, q, r, s;
456 p = DriftX.Max();
457 q = DriftY.Max();
458 r = StoryDriftX.Max();
459 s = StoryDriftY.Max();
460 List<double> DriftMaxX = new List<double>();
461 List<double> DriftMaxY = new List<double>();
462 DriftMaxX.Add(p);
463 DriftMaxY.Add(q);
464 DriftMaxX.Add(r);
465 DriftMaxY.Add(s);
466
467 double DriftMax = Math.Max(DriftMaxX.Max(), DriftMaxY.Max());
468 double WallPass = Math.Min(WallPassX.Min(), WallPassY.Min());
469
470 if (DriftMax > DriftLimit)
471 {
472     Fail = 0;
473 }
474 else
475 {
476     if (WallPass < 1)
477     {
478         Fail = 0;
479     }
480     else
481     {
482         Fail = 3;
483     }
484 }
485
486 loopCount = loopCount + 1;
487 if (loopCount > 15)
488 {

```

```

489         break;
490     }
491
492     //ADD ONE X-WALL IF X WALLS FAIL
493     temp3.Add(DriftMaxX.Max());
494     temp3.Add(DriftMaxY.Max());
495     if (DriftMaxX.Max() > DriftLimit)
496     {
497         nXfacWalls = nXfacWalls + 1;
498     }
499     else
500     {
501         if (WallPassX.Min() < 1)
502         {
503             nXfacWalls = nXfacWalls + 1;
504         }
505         else { }
506     }
507
508     //ADD ONE Y-WALL IF Y WALLS FAIL
509     if (DriftMaxY.Max() > DriftLimit)
510     {
511         nYfacWalls = nYfacWalls + 1;
512     }
513     else
514     {
515         if (WallPassY.Min() < 1)
516         {
517             nYfacWalls = nYfacWalls + 1;
518         }
519         else { }
520     }
521
522     //BUILD FACADE WALL (LINE)
523     double Xwall = nXfacWalls * LSeg;
524     double Ywall = nYfacWalls * LSeg;
525
526     double NXsp = nXfacWalls - 1;
527     double NYsp = nYfacWalls - 1;
528
529     double LXsp = (XDim - nXfacWalls * LSeg) / NXsp;
530     double LYsp = (YDim - nYfacWalls * LSeg) / NYsp;
531
532     //X-Facade Walls
533     if (Xwall > (XDim))
534     {
535         xFacWalls = new List<Line>();
536         //Full wall is Solid
537         tempPt1 = new Point3d(0, 0, 0);
538         tempPt2 = new Point3d(XDim, 0, 0);
539         tempPt3 = new Point3d(0, YDim, 0);
540         tempPt4 = new Point3d(XDim, YDim, 0);
541         xFacWalls.Add(new Line(tempPt1, tempPt2));
542         xFacWalls.Add(new Line(tempPt3, tempPt4));
543     }
544     else if (Xwall != 0)
545     {
546         //Distribute wall segments of length Seg along facade

```

```

547     xFacWalls = new List<Line>();
548     for (int i = 0; i < nXfacWalls; i++)
549     {
550         if (nXfacWalls != 1)
551         {
552             tempPt1 = new Point3d(i * LSeg + i * LXsp, 0, 0);
553             tempPt2 = new Point3d((i + 1) * LSeg + i *
554                 LXsp, 0, 0);
555             tempPt3 = new Point3d(i * LSeg + i * LXsp,
556                 YDim, 0);
557             tempPt4 = new Point3d((i + 1) * LSeg + i *
558                 LXsp, YDim, 0);
559             xFacWalls.Add(new Line(tempPt1, tempPt2));
560             xFacWalls.Add(new Line(tempPt3, tempPt4));
561         }
562         else
563         {
564             tempPt1 = new Point3d(0.5 * XDim - LSeg / 2, 0,
565                 0);
566             tempPt2 = new Point3d(0.5 * XDim + LSeg / 2, 0,
567                 0);
568             tempPt3 = new Point3d(0.5 * XDim - LSeg / 2,
569                 YDim, 0);
570             tempPt4 = new Point3d(0.5 * XDim + LSeg / 2,
571                 YDim, 0);
572             xFacWalls.Add(new Line(tempPt1, tempPt2));
573             xFacWalls.Add(new Line(tempPt3, tempPt4));
574         }
575     }
576
577     //Y-Facade Walls
578     if (Ywall > (YDim))
579     {
580         yFacWalls = new List<Line>();
581         //Full wall is Solid
582         tempPt1 = new Point3d(0, 0, 0);
583         tempPt2 = new Point3d(0, YDim, 0);
584         tempPt3 = new Point3d(XDim, 0, 0);
585         tempPt4 = new Point3d(XDim, YDim, 0);
586         yFacWalls.Add(new Line(tempPt1, tempPt2));
587         yFacWalls.Add(new Line(tempPt3, tempPt4));
588     }
589     else if (Ywall != 0)
590     {
591         yFacWalls = new List<Line>();
592         //Distribute wall segments of length Seg along facade
593         for (int i = 0; i < nYfacWalls; i++)
594         {
595             if (nYfacWalls != 1)
596             {
597                 tempPt1 = new Point3d(0, i * LSeg + i * LYsp, 0);
598                 tempPt2 = new Point3d(0, (i + 1) * LSeg + i *
599                     LYsp, 0);
600                 tempPt3 = new Point3d(XDim, i * LSeg + i *
601                     LYsp, 0);
602                 tempPt4 = new Point3d(XDim, (i + 1) * LSeg + i
603                     * LYsp, 0);

```

```

595         yFacWalls.Add(new Line(tempPt1, tempPt2));
596         yFacWalls.Add(new Line(tempPt3, tempPt4));
597     }
598     else
599     {
600         tempPt1 = new Point3d(0, 0.5 * YDim - LSeg / 2,
601                               0);
602         tempPt2 = new Point3d(0, 0.5 * YDim + LSeg / 2,
603                               0);
604         tempPt3 = new Point3d(XDim, 0.5 * YDim - LSeg /
605                               2, 0);
606         tempPt4 = new Point3d(XDim, 0.5 * YDim + LSeg /
607                               2, 0);
608         yFacWalls.Add(new Line(tempPt1, tempPt2));
609         yFacWalls.Add(new Line(tempPt3, tempPt4));
610     }
611 }
612 }
613 }//END WHILE LOOP TO DESIGN SHEAR WALLS
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640

```

```

641 {
642     //Loop Y Coordinates
643     for (double j = 1.0; j <= ny; j += 1.0)
644     {
645         //Loop Z Coordinates
646         for (int k = 0; k <= nz; k++)
647         {
648             //Node Coordinate X & Y
649             xx = (i - 1.0) * sx;
650             yy = (j - 1.0) * sy;
651
652             //Adjacent Coordinate X & Y
653             xx1 = i * sx;
654             yy1 = j * sy;
655
656             //Check if coordinates are in or out of core
657             ptin = false;
658             ptlxin = false;
659             ptlyin = false;
660             bmXin = false;
661             bmYin = false;
662
663             //Point
664             if (xx <= cxmax && xx >= cxmin && yy <= cymax && yy
665                 >= cymin)
666             {
667                 ptin = true;
668             }
669             //Next Point X
670             if (xx1 <= cxmax && xx1 >= cxmin && yy <= cymax &&
671                 yy >= cymin)
672             {
673                 ptlxin = true;
674             }
675             //Next Point Y
676             if (yy1 <= cymax && yy1 >= cymin && xx <= cxmax &&
677                 xx >= cxmin)
678             {
679                 ptlyin = true;
680             }
681             //Beam Crosses Core
682             if (xx < cxmin && xx1 > cxmax && yy <= cymax && yy
683                 >= cymin)
684             {
685                 bmXin = true;
686             }
687             if (yy < cymin && yy1 > cymax && xx <= cxmax && xx
688                 >= cxmin)
689             {
690                 bmYin = true;
691             }
692
693             //CHECK IF POINT IS ON WALL
694             tempPt1 = new Point3d(xx, yy, 0);
695             tempPt2 = new Point3d(xx1, yy, 0);
696             tempPt3 = new Point3d(xx, yy1, 0);
697             bool PtOnLine = false;
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901

```

```

694 List<Line> OnWall = new List<Line>();
695 List<Line> X1OnWall = new List<Line>();
696 List<Line> Y1OnWall = new List<Line>();
697
698 foreach (Line v in AllWalls)
699 {
700     if (v.MinimumDistanceTo(tempPt1) < 0.1)
701     {
702         PtOnLine = true;
703         OnWall.Add(v);
704     }
705     if (v.MinimumDistanceTo(tempPt2) < 0.1)
706     {
707         X1OnWall.Add(v);
708     }
709     if (v.MinimumDistanceTo(tempPt3) < 0.1)
710     {
711         Y1OnWall.Add(v);
712     }
713 }
714
715 ///Draw X Beams
716 if (i == nx) { }
717 else if (j == 1)
718 {
719     if (k != 0)
720     {
721         //FIRST PERIMETER
722         if (nXfacWalls == 0)
723         {
724             node = new Point3d(xx, yy, height[k]);
725             node1x = new Point3d(xx1, yy, height[k]);
726             beamsX.Add(new LineCurve(node, node1x));
727         }
728         else
729         {
730             xx0 = xx;
731             foreach (Line v in OnWall)
732             {
733                 if (v.FromX > xx0)
734                 {
735                     xx0 = v.FromX;
736                 }
737                 if (v.ToX > xx0)
738                 {
739                     xx0 = v.ToX;
740                 }
741             }
742             if (xx0 >= xx1)
743             {
744                 //WALL ALONG ENTIRE BAY, NO BEAM
745             }
746             else
747             {
748                 xx2 = xx1;
749                 foreach (Line v in X1OnWall)
750                 {
751                     if (v.FromX < xx2)

```

```

752     {
753         xx2 = v.FromX;
754     }
755     if (v.ToX < xx2)
756     {
757         xx2 = v.ToX;
758     }
759 }
760 bool wallBtwn = false;
761 xx3 = 0;
762 xx4 = 0;
763 foreach (Line v in AllWalls)
764 {
765     if (v.FromY == v.ToY && v.FromY == yy)
766     {
767         if (v.FromX > xx && v.ToX < xx1)
768         {
769             xx3 = v.FromX;
770             xx4 = v.ToX;
771             wallBtwn = true;
772         }
773     }
774 }
775 if (wallBtwn == true)
776 {
777     //WALL BETWEEN POINTS
778     node = new Point3d(xx0, yy, height[k]);
779     node1x = new Point3d(xx3, yy, height[k]);
780     beamsX.Add(new LineCurve(node, node1x));
781     node = new Point3d(xx4, yy, height[k]);
782     node1x = new Point3d(xx2, yy, height[k]);
783     beamsX.Add(new LineCurve(node, node1x));
784 }
785 else
786 {
787     node = new Point3d(xx0, yy, height[k]);
788     node1x = new Point3d(xx2, yy, height[k]);
789     beamsX.Add(new LineCurve(node, node1x));
790 }
791 }
792 }
793 }
794 }
795 }
796 else if (j == ny)
797 {
798     if (k != 0)

```

```

799 //SECOND PERIMETER
800 if (nXfacWalls == 0)
801 {
802     node = new Point3d(xx, yy, height[k]);
803     node1x = new Point3d(xx1, yy, height[k]);
804     beamsX.Add(new LineCurve(node, node1x));
805 }
806 else
807 {
808     xx0 = xx;
809     foreach (Line v in OnWall)
810     {
811         if (v.FromX > xx0)
812         {
813             xx0 = v.FromX;
814         }
815         if (v.ToX > xx0)
816         {
817             xx0 = v.ToX;
818         }
819     }
820     if (xx0 >= xx1)
821     {
822         //WALL ALONG ENTIRE BAY, NO BEAM
823     }
824     else
825     {
826         xx2 = xx1;
827         foreach (Line v in X1OnWall)
828         {
829             if (v.FromX < xx2)
830             {
831                 xx2 = v.FromX;
832             }
833             if (v.ToX < xx2)
834             {
835                 xx2 = v.ToX;
836             }
837         }
838         bool wallBtwn = false;
839         xx3 = 0;
840         xx4 = 0;
841         foreach (Line v in AllWalls)
842         {
843             if (v.FromY == v.ToY && v.FromY == yy)
844             {
845                 if (v.FromX > xx && v.ToX < xx1)
846                 {
847                     xx3 = v.FromX;
848                     xx4 = v.ToX;
849                     wallBtwn = true;
850                 }
851             }
852         }
853         if (wallBtwn == true)
854         {

```

```

855 //WALL BETWEEN POINTS
856 node = new Point3d(xx0, yy, height[k]);
857 node1x = new Point3d(xx3, yy, height[k]);
858 beamsX.Add(new LineCurve(node, node1x));
859 node = new Point3d(xx4, yy, height[k]);
860 node1x = new Point3d(xx2, yy, height[k]);
861 beamsX.Add(new LineCurve(node, node1x));
862 }
863 else
864 {
865     node = new Point3d(xx0, yy, height[k]);
866     node1x = new Point3d(xx2, yy, height[k]);
867     beamsX.Add(new LineCurve(node, node1x));
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }

```

```

900         height[k + 1]);
          WallColumn.Add(new
901             LineCurve(node0z, node1z));
          nodes.Add(node0z);
902     }
903 }
904 else if (ptin == false && ptlxin == true)
905 {
906     xx1 = cxmin;
907     node = new Point3d(xx, yy, height[k]);
908     node1x = new Point3d(xx1, yy, height[k]);
909     beamsX.Add(new LineCurve(node, node1x));
910
911     if (k != nz)
912     {
913         if (k == 1)
914         {
915             node0z = new Point3d(xx1, yy,
916                 height[k - 1]);
917             node1z = new Point3d(xx1, yy,
918                 height[k]);
919             WallColumn.Add(new
920                 LineCurve(node0z, node1z));
921             nodes.Add(node0z);
922
923             node0z = new Point3d(xx1, yy,
924                 height[k]);
925             node1z = new Point3d(xx1, yy,
926                 height[k + 1]);
927             WallColumn.Add(new
928                 LineCurve(node0z, node1z));
929             nodes.Add(node0z);
930
931             node = new Point3d(xx, yy,
932                 height[k]);
933             node1x = new Point3d(cxmin, yy,
934                 height[k]);
935             beamsX.Add(new LineCurve(node,
936                 node1x));
937             nodes.Add(new Point3d(cxmax, yy, 0));
938
939             node = new Point3d(cxmax, yy,
940                 height[k]);
941             node1x = new Point3d(xx1, yy,
942

```

```

943         height[k]);
          beamsX.Add(new LineCurve(node,
944             node1x));
          }
945     }
946 }
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998

```

```

999         if (v.FromX == v.ToX && v.FromY
1000 == xx)
1001         {
1002             if (v.FromY > yy && v.ToY <
1003 yy1)
1004             {
1005                 yy3 = v.FromY;
1006                 yy4 = v.ToY;
1007                 wallBtwn = true;
1008             }
1009         }
1010         if (wallBtwn == true)
1011         {
1012             //WALL BETWEEN POINTS
1013             node = new Point3d(xx, yy0,
1014 height[k]);
1015             nodely = new Point3d(xx, yy3,
1016 height[k]);
1017             beamsY.Add(new LineCurve(node,
1018 nodely));
1019             node = new Point3d(xx, yy4,
1020 height[k]);
1021             nodely = new Point3d(xx, yy2,
1022 height[k]);
1023             beamsY.Add(new LineCurve(node,
1024 nodely));
1025         }
1026     }
1027 }
1028 }
1029 else if (i == nx)
1030 {
1031     if (k != 0)
1032     {
1033         //SECOND PERIMETER
1034         if (nYfacWalls == 0)
1035         {
1036             node = new Point3d(xx, yy, height[k]);
1037             nodely = new Point3d(xx, yy1, height[k]);
1038             beamsY.Add(new LineCurve(node, nodely));
1039         }
1040         else
1041         {
1042             yy0 = yy;
1043             foreach (Line v in OnWall)
1044             {
1045                 if (v.FromY > yy0)

```

```

1046         {
1047             yy0 = v.FromY;
1048         }
1049         if (v.ToY > yy0)
1050         {
1051             yy0 = v.ToY;
1052         }
1053     }
1054     if (yy0 >= yy1)
1055     {
1056         //WALL ALONG ENTIRE BAY, NO BEAM
1057     }
1058     else
1059     {
1060         yy2 = yy1;
1061         foreach (Line v in Y1OnWall)
1062         {
1063             if (v.FromY < yy2)
1064             {
1065                 yy2 = v.FromY;
1066             }
1067             if (v.ToY < yy2)
1068             {
1069                 yy2 = v.ToY;
1070             }
1071         }
1072         bool wallBtwn = false;
1073         yy3 = 0;
1074         yy4 = 0;
1075         foreach (Line v in AllWalls)
1076         {
1077             if (v.FromX == v.ToX && v.FromY
1078 == xx)
1079             {
1080                 if (v.FromY > yy && v.ToY <
1081 yy1)
1082                 {
1083                     yy3 = v.FromY;
1084                     yy4 = v.ToY;
1085                     wallBtwn = true;
1086                 }
1087             }
1088         }
1089     }
1090     if (wallBtwn == true)
1091     {
1092         //WALL BETWEEN POINTS
1093         node = new Point3d(xx, yy0,
1094 height[k]);
1095         nodely = new Point3d(xx, yy3,
1096 height[k]);
1097         beamsY.Add(new LineCurve(node,
1098 nodely));
1099         node = new Point3d(xx, yy4,
1100 height[k]);
1101         nodely = new Point3d(xx, yy2,
1102 height[k]);
1103         beamsY.Add(new LineCurve(node,
1104 nodely));

```

```

1096         }
1097         else
1098         {
1099             node = new Point3d(xx, yy0,
1100                 height[k]);
1101             nodely = new Point3d(xx, yy2,
1102                 height[k]);
1103             beamsY.Add(new LineCurve(node,
1104                 nodely));
1105         }
1106     }
1107 else
1108 {
1109     if (k != 0)
1110     {
1111         if (ptin == true && ptlyin == true)
1112         {
1113             ///No Y Beam
1114         }
1115         else if (ptin == true && ptlyin == false)
1116         {
1117             yy0 = cymax;
1118             node = new Point3d(xx, yy0, height[k]);
1119             nodely = new Point3d(xx, yy1, height[k]);
1120             beamsY.Add(new LineCurve(node, nodely));
1121
1122             if (k != nz)
1123             {
1124                 if (k == 1)
1125                 {
1126                     node0z = new Point3d(xx, yy0,
1127                         height[k - 1]);
1128                     node1z = new Point3d(xx, yy0,
1129                         height[k]);
1130                     WallColumn.Add(new
1131                         LineCurve(node0z, node1z));
1132                     nodes.Add(node0z);
1133                 }
1134                 node0z = new Point3d(xx, yy0,
1135                     height[k]);
1136                 node1z = new Point3d(xx, yy0,
1137                     height[k + 1]);
1138                 WallColumn.Add(new
1139                     LineCurve(node0z, node1z));
1140                 nodes.Add(node0z);
1141             }
1142         }
1143         else if (ptin == false && ptlyin == true)
1144         {
1145             yy1 = cymin;
1146             node = new Point3d(xx, yy, height[k]);
1147             nodely = new Point3d(xx, yy1, height[k]);
1148             beamsY.Add(new LineCurve(node, nodely));
1149
1150             if (k != nz)

```

```

1145         {
1146             if (k == 1)
1147             {
1148                 node0z = new Point3d(xx, yy1,
1149                     height[k - 1]);
1150                 node1z = new Point3d(xx, yy1,
1151                     height[k]);
1152                 WallColumn.Add(new
1153                     LineCurve(node0z, node1z));
1154                 nodes.Add(node0z);
1155             }
1156             node0z = new Point3d(xx, yy1,
1157                 height[k]);
1158             node1z = new Point3d(xx, yy1,
1159                 height[k + 1]);
1160             WallColumn.Add(new
1161                 LineCurve(node0z, node1z));
1162             nodes.Add(node0z);
1163         }
1164     }
1165     else if (ptin == false && ptlyin == false)
1166     {
1167         if (bmYin == true)
1168         {
1169             node = new Point3d(xx, yy,
1170                 height[k]);
1171             nodely = new Point3d(xx, cymin,
1172                 height[k]);
1173             beamsY.Add(new LineCurve(node,
1174                 nodely));
1175             nodes.Add(new Point3d(xx, cymin, 0));
1176             node = new Point3d(xx, cymax,
1177                 height[k]);
1178             nodely = new Point3d(xx, yy1,
1179                 height[k]);
1180             beamsY.Add(new LineCurve(node,
1181                 nodely));
1182             nodes.Add(new Point3d(xx, yy1, 0));
1183         }
1184         else
1185         {
1186             node = new Point3d(xx, yy,
1187                 height[k]);
1188             nodely = new Point3d(xx, yy1,
1189                 height[k]);
1190             beamsY.Add(new LineCurve(node,
1191                 nodely));
1192         }
1193     }
1194 }
1195
1196 ///Draw Columns
1197 if (k == nz) { }
1198 else
1199 {
1200     if (ptin == true) { }
1201     else

```

```

1188         {
1189             node = new Point3d(xx, yy, height[k]);
1190             nodelz = new Point3d(xx, yy, height[k + 1]);
1191             if (PtOnLine == true)
1192             {
1193                 WallColumn.Add(new LineCurve(node,
1194                                         nodelz));
1195             }
1196             else
1197             {
1198                 columnsZ.Add(new LineCurve(node,
1199                                         nodelz));
1200             }
1201             nodes.Add(node);
1202         }
1203     }
1204 }
1205
1206
1207
1208
1209
1210
1211 List<Mesh> WallMeshCore = new List<Mesh>();
1212 List<Mesh> WallMeshPerim = new List<Mesh>();
1213 List<Point3d> supports = new List<Point3d>();
1214
1215 //COLUMN/WALL BASE POINTS
1216 foreach (Point3d v in nodes)
1217 {
1218     if (v.Z < 0.1)
1219     {
1220         supports.Add(v);
1221     }
1222 }
1223 foreach (Line v in AllWalls)
1224 {
1225     supports.Add(v.From);
1226     supports.Add(v.To);
1227 }
1228
1229 foreach (Line u in AllWalls)
1230 {
1231     for (int j = 0; j < nz; j++)
1232     {
1233         //FIND POINTS ON LINE
1234         List<Point3d> WallPoints = new List<Point3d>();
1235         foreach (Point3d v in supports)

```

```

1236     {
1237         if (u.MinimumDistanceTo(v) < 0.1)
1238         {
1239             WallPoints.Add(v);
1240         }
1241     }
1242 }
1243 //BUILD WALL MESHES
1244 if (u.FromX == u.ToX)
1245 {
1246     //Y-Direction Wall
1247     WallPoints = WallPoints.OrderByDescending(p =>
1248         p.Y).ToList();
1249     for (int i = 1; i < (WallPoints.Count); i++)
1250     {
1251         Mesh WallCurrent = BuildMesh(new
1252             Point3d(u.FromX, WallPoints[i - 1].Y,
1253                 height[j]), new Point3d(u.FromX, WallPoints[i -
1254             1].Y, height[j + 1]), new Point3d(u.ToX,
1255             WallPoints[i].Y, height[j + 1]), new
1256             Point3d(u.ToX, WallPoints[i].Y, height[j]));
1257         if (u.FromX == 0 || u.FromX == XDim)
1258         {
1259             WallMeshPerim.Add(WallCurrent);
1260         }
1261         else
1262         {
1263             WallMeshCore.Add(WallCurrent);
1264         }
1265     }
1266 }
1267 else
1268 {
1269     //X-Direction Wall
1270     WallPoints = WallPoints.OrderByDescending(p =>
1271         p.X).ToList();
1272     for (int i = 1; i < WallPoints.Count; i++)
1273     {
1274         Mesh WallCurrent = BuildMesh(new
1275             Point3d(WallPoints[i - 1].X, WallPoints[i -
1276             1].Y, height[j]), new Point3d(WallPoints[i -
1277             1].X, WallPoints[i - 1].Y, height[j + 1]), new
1278             Point3d(WallPoints[i].X, WallPoints[i].Y,
1279             height[j + 1]), new Point3d(WallPoints[i].X,
1280             WallPoints[i].Y, height[j]));
1281         if (u.FromY == 0 || u.FromY == YDim)
1282         {
1283             WallMeshPerim.Add(WallCurrent);
1284         }
1285         else
1286         {
1287             WallMeshCore.Add(WallCurrent);
1288         }
1289     }
1290 }

```

```

1281     }
1282
1283     ///-----
1284     ///-----
1285     ///OUTPUTS
1286     ///-----
1287     ///-----
1288     DA.SetDataList(0, beamsX);
1289     DA.SetDataList(1, beamsY);
1290     DA.SetDataList(2, columnsZ);
1291     DA.SetDataList(3, floorMesh);
1292     DA.SetData(4, roofMesh);
1293     DA.SetDataList(5, WallMeshCore);
1294     DA.SetDataList(6, WallMeshPerim);
1295     DA.SetDataList(7, AllWalls);
1296     DA.SetDataList(8, WallColumn);
1297     DA.SetDataList(9, ListWallShear);
1298     DA.SetDataList(10, ListWallMoment);
1299 }
1300
1301 private List<double> GetHeights(double nStories, double Height1,
1302 double HeightTyp)
1303 {
1304     List<double> height = new List<double>();
1305     height.Add(0);
1306     for (int i = 0; i < nStories; i++)
1307     {
1308         height.Add(Height1 + i * HeightTyp);
1309     }
1310     return height;
1311 }
1312 private Mesh BuildMesh(Point3d A, Point3d B, Point3d C, Point3d D)
1313 {
1314     Mesh NewMesh = new Mesh();
1315
1316     NewMesh.Vertices.Add(A);
1317     NewMesh.Vertices.Add(B);
1318     NewMesh.Vertices.Add(C);
1319     NewMesh.Vertices.Add(D);
1320     NewMesh.Faces.AddFace(0, 1, 2, 3);
1321
1322     return NewMesh;
1323 }
1324 public class SeismicForces
1325 {
1326     public List<double> wxhvk;
1327     public List<double> StoryForce;
1328     public List<double> StoryShear;

```

```

1330     public List<double> StoryMoment;
1331     public double MomentArm;
1332
1333     public SeismicForces()
1334     {
1335         wxhvk = new List<double>();
1336         StoryForce = new List<double>();
1337         StoryShear = new List<double>();
1338         StoryMoment = new List<double>();
1339         MomentArm = 0;
1340     }
1341
1342     public SeismicForces(List<double> wxhvkIn, List<double>
1343 StoryForceIn, List<double> StoryShearIn, List<double>
1344 StoryMomentIn, double MomentArmIn)
1345     {
1346         wxhvk = wxhvkIn;
1347         StoryForce = StoryForceIn;
1348         StoryShear = StoryShearIn;
1349         StoryMoment = StoryMomentIn;
1350         MomentArm = MomentArmIn;
1351     }
1352
1353     public void CalculateSeismicForces(List<double> Heights, double
1354 EqMassTyp, double EqMassRoof, double FloorArea, double Veq,
1355 double kEq)
1356     {
1357         List<double> wxhvkCalc = new List<double>();
1358         for (int i = 1; i <= (Heights.Count() - 1); i++)
1359         {
1360             if (i < (Heights.Count() - 1))
1361             {
1362                 //FLOOR
1363                 wxhvkCalc.Add(EqMassTyp * FloorArea *
1364                     Math.Pow(Heights[i], kEq)); //lb and ft
1365             }
1366             else
1367             {
1368                 //ROOF
1369                 wxhvkCalc.Add(EqMassRoof * FloorArea *
1370                     Math.Pow(Heights[i], kEq)); //lb and ft
1371             }
1372         }
1373         wxhvk = wxhvkCalc;
1374
1375         List<double> storyforce = new List<double>();
1376         List<double> storyshear = new List<double>();
1377         List<double> storymoment = new List<double>();
1378         for (int i = 1; i <= Heights.Count() - 1; i++)
1379         {
1380             if (i == 1)
1381             {
1382                 storyforce.Add(wxhvk[i - 1] * Veq / wxhvk.Sum()); //K
1383                 storyshear.Add(Veq); //K
1384                 storymoment.Add(storyforce[i - 1] * Heights[i]);
1385                 //K-ft
1386             }
1387             else

```

```

1381         {
1382             storyforce.Add(wxhxk[i - 1] * Veq / wxhxk.Sum()); //K
1383             storyshear.Add(storyshear[i - 2] - storyforce[i -
1384             2]); //K
1385             storymoment.Add(storyforce[i - 1] * Heights[i]);
1386             //K-ft
1387         }
1388     }
1389     double momentarm = storymoment.Sum() / Veq; //ft
1390
1391     StoryForce = storyforce;
1392     StoryShear = storyshear;
1393     StoryMoment = storymoment;
1394     MomentArm = momentarm;
1395 }
1396 }
1397
1398 public override GH_Exposure Exposure
1399 {
1400     get { return GH_Exposure.primary; }
1401 }
1402
1403 protected override System.Drawing.Bitmap Icon
1404 {
1405     get
1406     {
1407         return null;
1408     }
1409 }
1410
1411 public override Guid ComponentGuid
1412 {
1413     get { return new
1414     Guid("{138e8ac5-1b07-4010-b2fa-ae6fa531e513}"); }
1415 }
1416 }
1417
1418 //-----
1419 //-----
1420 //-----
1421
1422 using Rhino;
1423 using Rhino.Geometry;
1424 using Rhino.DocObjects;
1425 using Rhino.Collections;
1426
1427 using GH_IO;
1428 using GH_IO.Serialization;
1429 using Grasshopper;
1430 using Grasshopper.Kernel;
1431 using Grasshopper.Kernel.Data;

```

```

1432 using Grasshopper.Kernel.Types;
1433
1434 using System;
1435 using System.IO;
1436 using System.Xml;
1437 using System.Xml.Linq;
1438 using System.Linq;
1439 using System.Data;
1440 using System.Drawing;
1441 using System.Reflection;
1442 using System.Collections;
1443 using System.Windows.Forms;
1444 using System.Collections.Generic;
1445 using System.Runtime.InteropServices;
1446
1447 public class Script_Instance : GH_ScriptInstance
1448 {
1449     #region Utility functions
1450     private void Print(string text) { /* Implementation hidden. */ }
1451     private void Print(string format, params object[] args) { /*
1452     Implementation hidden. */ }
1453     private void Reflect(object obj) { /* Implementation hidden. */ }
1454     private void Reflect(object obj, string method_name) { /*
1455     Implementation hidden. */ }
1456     #endregion
1457
1458     #region Members
1459     private readonly RhinoDoc RhinoDocument;
1460     private readonly GH_Document GrasshopperDocument;
1461     private readonly IGH_Component Component;
1462     private readonly int Iteration;
1463     #endregion
1464
1465     private void RunScript(double TypSIL, double TypSDL, double TypLL,
1466     double SpanX, double SpanY, double DLFact, double lambda, double tConc,
1467     bool cSlab, bool fire, ref object tSlab, ref object SlabSW, ref object
1468     SlabDir, ref object pass)
1469     {
1470         //-----
1471         //-----
1472         //-----
1473         //-----
1474         //-----
1475         //-----
1476         //-----
1477         //-----
1478         //-----
1479         //-----
1480         //-----
1481         //-----
1482         //-----
1483         //-----
1484         //-----
1485         //-----
1486         //-----
1487         //-----
1488         //-----
1489         //-----
1490         //-----
1491         //-----
1492         //-----
1493         //-----
1494         //-----
1495         //-----
1496         //-----
1497         //-----
1498         //-----
1499         //-----
1500         //-----
1501         //-----
1502         //-----
1503         //-----
1504         //-----
1505         //-----
1506         //-----
1507         //-----
1508         //-----
1509         //-----
1510         //-----
1511         //-----
1512         //-----
1513         //-----
1514         //-----
1515         //-----
1516         //-----
1517         //-----
1518         //-----
1519         //-----
1520         //-----
1521         //-----
1522         //-----
1523         //-----
1524         //-----
1525         //-----
1526         //-----
1527         //-----
1528         //-----
1529         //-----
1530         //-----
1531         //-----
1532         //-----
1533         //-----
1534         //-----
1535         //-----
1536         //-----
1537         //-----
1538         //-----
1539         //-----
1540         //-----
1541         //-----
1542         //-----
1543         //-----
1544         //-----
1545         //-----
1546         //-----
1547         //-----
1548         //-----
1549         //-----
1550         //-----
1551         //-----
1552         //-----
1553         //-----
1554         //-----
1555         //-----
1556         //-----
1557         //-----
1558         //-----
1559         //-----
1560         //-----
1561         //-----
1562         //-----
1563         //-----
1564         //-----
1565         //-----
1566         //-----
1567         //-----
1568         //-----
1569         //-----
1570         //-----
1571         //-----
1572         //-----
1573         //-----
1574         //-----
1575         //-----
1576         //-----
1577         //-----
1578         //-----
1579         //-----
1580         //-----
1581         //-----
1582         //-----
1583         //-----
1584         //-----
1585         //-----
1586         //-----
1587         //-----
1588         //-----
1589         //-----
1590         //-----
1591         //-----
1592         //-----
1593         //-----
1594         //-----
1595         //-----
1596         //-----
1597         //-----
1598         //-----
1599         //-----
1600         //-----
1601         //-----
1602         //-----
1603         //-----
1604         //-----
1605         //-----
1606         //-----
1607         //-----
1608         //-----
1609         //-----
1610         //-----
1611         //-----
1612         //-----
1613         //-----
1614         //-----
1615         //-----
1616         //-----
1617         //-----
1618         //-----
1619         //-----
1620         //-----
1621         //-----
1622         //-----
1623         //-----
1624         //-----
1625         //-----
1626         //-----
1627         //-----
1628         //-----
1629         //-----
1630         //-----
1631         //-----
1632         //-----
1633         //-----
1634         //-----
1635         //-----
1636         //-----
1637         //-----
1638         //-----
1639         //-----
1640         //-----
1641         //-----
1642         //-----
1643         //-----
1644         //-----
1645         //-----
1646         //-----
1647         //-----
1648         //-----
1649         //-----
1650         //-----
1651         //-----
1652         //-----
1653         //-----
1654         //-----
1655         //-----
1656         //-----
1657         //-----
1658         //-----
1659         //-----
1660         //-----
1661         //-----
1662         //-----
1663         //-----
1664         //-----
1665         //-----
1666         //-----
1667         //-----
1668         //-----
1669         //-----
1670         //-----
1671         //-----
1672         //-----
1673         //-----
1674         //-----
1675         //-----
1676         //-----
1677         //-----
1678         //-----
1679         //-----
1680         //-----
1681         //-----
1682         //-----
1683         //-----
1684         //-----
1685         //-----
1686         //-----
1687         //-----
1688         //-----
1689         //-----
1690         //-----
1691         //-----
1692         //-----
1693         //-----
1694         //-----
1695         //-----
1696         //-----
1697         //-----
1698         //-----
1699         //-----
1700         //-----
1701         //-----
1702         //-----
1703         //-----
1704         //-----
1705         //-----
1706         //-----
1707         //-----
1708         //-----
1709         //-----
1710         //-----
1711         //-----
1712         //-----
1713         //-----
1714         //-----
1715         //-----
1716         //-----
1717         //-----
1718         //-----
1719         //-----
1720         //-----
1721         //-----
1722         //-----
1723         //-----
1724         //-----
1725         //-----
1726         //-----
1727         //-----
1728         //-----
1729         //-----
1730         //-----
1731         //-----
1732         //-----
1733         //-----
1734         //-----
1735         //-----
1736         //-----
1737         //-----
1738         //-----
1739         //-----
1740         //-----
1741         //-----
1742         //-----
1743         //-----
1744         //-----
1745         //-----
1746         //-----
1747         //-----
1748         //-----
1749         //-----
1750         //-----
1751         //-----
1752         //-----
1753         //-----
1754         //-----
1755         //-----
1756         //-----
1757         //-----
1758         //-----
1759         //-----
1760         //-----
1761         //-----
1762         //-----
1763         //-----
1764         //-----
1765         //-----
1766         //-----
1767         //-----
1768         //-----
1769         //-----
1770         //-----
1771         //-----
1772         //-----
1773         //-----
1774         //-----
1775         //-----
1776         //-----
1777         //-----
1778         //-----
1779         //-----
1780         //-----
1781         //-----
1782         //-----
1783         //-----
1784         //-----
1785         //-----
1786         //-----
1787         //-----
1788         //-----
1789         //-----
1790         //-----
1791         //-----
1792         //-----
1793         //-----
1794         //-----
1795         //-----
1796         //-----
1797         //-----
1798         //-----
1799         //-----
1800         //-----
1801         //-----
1802         //-----
1803         //-----
1804         //-----
1805         //-----
1806         //-----
1807         //-----
1808         //-----
1809         //-----
1810         //-----
1811         //-----
1812         //-----
1813         //-----
1814         //-----
1815         //-----
1816         //-----
1817         //-----
1818         //-----
1819         //-----
1820         //-----
1821         //-----
1822         //-----
1823         //-----
1824         //-----
1825         //-----
1826         //-----
1827         //-----
1828         //-----
1829         //-----
1830         //-----
1831         //-----
1832         //-----
1833         //-----
1834         //-----
1835         //-----
1836         //-----
1837         //-----
1838         //-----
1839         //-----
1840         //-----
1841         //-----
1842         //-----
1843         //-----
1844         //-----
1845         //-----
1846         //-----
1847         //-----
1848         //-----
1849         //-----
1850         //-----
1851         //-----
1852         //-----
1853         //-----
1854         //-----
1855         //-----
1856         //-----
1857         //-----
1858         //-----
1859         //-----
1860         //-----
1861         //-----
1862         //-----
1863         //-----
1864         //-----
1865         //-----
1866         //-----
1867         //-----
1868         //-----
1869         //-----
1870         //-----
1871         //-----
1872         //-----
1873         //-----
1874         //-----
1875         //-----
1876         //-----
1877         //-----
1878         //-----
1879         //-----
1880         //-----
1881         //-----
1882         //-----
1883         //-----
1884         //-----
1885         //-----
1886         //-----
1887         //-----
1888         //-----
1889         //-----
1890         //-----
1891         //-----
1892         //-----
1893         //-----
1894         //-----
1895         //-----
1896         //-----
1897         //-----
1898         //-----
1899         //-----
1900         //-----
1901         //-----
1902         //-----
1903         //-----
1904         //-----
1905         //-----
1906         //-----
1907         //-----
1908         //-----
1909         //-----
1910         //-----
1911         //-----
1912         //-----
1913         //-----
1914         //-----
1915         //-----
1916         //-----
1917         //-----
1918         //-----
1919         //-----
1920         //-----
1921         //-----
1922         //-----
1923         //-----
1924         //-----
1925         //-----
1926         //-----
1927         //-----
1928         //-----
1929         //-----
1930         //-----
1931         //-----
1932         //-----
1933         //-----
1934         //-----
1935         //-----
1936         //-----
1937         //-----
1938         //-----
1939         //-----
1940         //-----
1941         //-----
1942         //-----
1943         //-----
1944         //-----
1945         //-----
1946         //-----
1947         //-----
1948         //-----
1949         //-----
1950         //-----
1951         //-----
1952         //-----
1953         //-----
1954         //-----
1955         //-----
1956         //-----
1957         //-----
1958         //-----
1959         //-----
1960         //-----
1961         //-----
1962         //-----
1963         //-----
1964         //-----
1965         //-----
1966         //-----
1967         //-----
1968         //-----
1969         //-----
1970         //-----
1971         //-----
1972         //-----
1973         //-----
1974         //-----
1975         //-----
1976         //-----
1977         //-----
1978         //-----
1979         //-----
1980         //-----
1981         //-----
1982         //-----
1983         //-----
1984         //-----
1985         //-----
1986         //-----
1987         //-----
1988         //-----
1989         //-----
1990         //-----
1991         //-----
1992         //-----
1993         //-----
1994         //-----
1995         //-----
1996         //-----
1997         //-----
1998         //-----
1999         //-----
2000         //-----

```

```

1477     int A;
1478     double chardepth = 1.2 * (1.5 * Math.Pow(1, 0.813));
1479     double depth;
1480     double Mmax, Vmax;
1481     double yCchar, yCLTchar, yNAchar, Itrchar;
1482
1483     //NDS ADJUSTMENT FACTORS
1484     double Cm = 1;
1485     double Ct = 1;
1486     double Cl = 1;
1487     double KBend = 2.54;
1488     double PhiBend = 0.85;
1489     double KShear = 2.88;
1490     double PhiShear = 0.75;
1491
1492     //MATERIAL PROPERTIES
1493     double ECLT = 1400000;
1494     double rho = 1.0625 * 0.5; //SPECIFIC GRAVITY
1495     double rhoh20 = 62.43; //DENSITY WATER PCF
1496     double rhoconc = 150; //PCF
1497     double rhoCLT = rho * rhoh20;
1498     double sg;
1499
1500     //CLT PANEL PROPERTIES FROM MANUFACTURERS
1501     List<double> ListPanelLength = new List<double>(); //FT
1502     ListPanelLength.Add(64);
1503     ListPanelLength.Add(40);
1504     ListPanelLength.Add(64);
1505     ListPanelLength.Add(64);
1506     ListPanelLength.Add(40);
1507     ListPanelLength.Add(64);
1508     ListPanelLength.Add(64);
1509     ListPanelLength.Add(40);
1510     ListPanelLength.Add(64);
1511     ListPanelLength.Add(40);
1512     ListPanelLength.Add(64);
1513
1514     List<double> ListDepth = new List<double>(); //IN
1515     ListDepth.Add(3.125);
1516     ListDepth.Add(3.9);
1517     ListDepth.Add(4.125);
1518     ListDepth.Add(5.125);
1519     ListDepth.Add(6.66);
1520     ListDepth.Add(6.875);
1521     ListDepth.Add(8.625);
1522     ListDepth.Add(9.42);
1523     ListDepth.Add(9.625);
1524     ListDepth.Add(12.18);
1525     ListDepth.Add(12.375);
1526
1527     List<double> ListSW = new List<double>(); //PSF
1528     ListSW.Add(8.33);
1529     ListSW.Add(10.5);
1530     ListSW.Add(11.0);
1531     ListSW.Add(13.7);
1532     ListSW.Add(17.0);
1533     ListSW.Add(18.3);
1534     ListSW.Add(23.0);

```

```

1535     ListSW.Add(25.0);
1536     ListSW.Add(25.7);
1537     ListSW.Add(32.0);
1538     ListSW.Add(33.0);
1539
1540     List<double> ListEI = new List<double>(); //LB-IN2
1541     ListEI.Add(48000000);
1542     ListEI.Add(79000000);
1543     ListEI.Add(115000000);
1544     ListEI.Add(184000000);
1545     ListEI.Add(321000000);
1546     ListEI.Add(440000000);
1547     ListEI.Add(853000000);
1548     ListEI.Add(818000000);
1549     ListEI.Add(1404000000);
1550     ListEI.Add(1662000000);
1551     ListEI.Add(2794000000);
1552
1553     List<double> ListGA = new List<double>(); //LB
1554     ListGA.Add(340000);
1555     ListGA.Add(490000);
1556     ListGA.Add(460000);
1557     ListGA.Add(690000);
1558     ListGA.Add(1000000);
1559     ListGA.Add(920000);
1560     ListGA.Add(1400000);
1561     ListGA.Add(1500000);
1562     ListGA.Add(2000000);
1563     ListGA.Add(2100000);
1564     ListGA.Add(2400000);
1565
1566     List<double> ListVmax = new List<double>(); //LB
1567     ListVmax.Add(1070);
1568     ListVmax.Add(1340);
1569     ListVmax.Add(1430);
1570     ListVmax.Add(1470);
1571     ListVmax.Add(1860);
1572     ListVmax.Add(1970);
1573     ListVmax.Add(2400);
1574     ListVmax.Add(2370);
1575     ListVmax.Add(3200);
1576     ListVmax.Add(2875);
1577     ListVmax.Add(3875);
1578
1579     List<double> ListMmax = new List<double>(); //LB-IN
1580     ListMmax.Add(2525);
1581     ListMmax.Add(1800);
1582     ListMmax.Add(4525);
1583     ListMmax.Add(5800);
1584     ListMmax.Add(4275);
1585     ListMmax.Add(10400);
1586     ListMmax.Add(15975);
1587     ListMmax.Add(7700);
1588     ListMmax.Add(23700);
1589     ListMmax.Add(12075);
1590     ListMmax.Add(36700);
1591
1592     List<double> ListFb0 = new List<double>();

```

```

1593 ListFb0.Add(1950);
1594 ListFb0.Add(875);
1595 ListFb0.Add(1950);
1596 ListFb0.Add(1950);
1597 ListFb0.Add(875);
1598 ListFb0.Add(1950);
1599 ListFb0.Add(1950);
1600 ListFb0.Add(875);
1601 ListFb0.Add(1950);
1602 ListFb0.Add(875);
1603 ListFb0.Add(1950);
1604
1605 List<double> ListFv0 = new List<double>();
1606 ListFv0.Add(135);
1607 ListFv0.Add(135);
1608 ListFv0.Add(135);
1609 ListFv0.Add(135);
1610 ListFv0.Add(135);
1611 ListFv0.Add(135);
1612 ListFv0.Add(135);
1613 ListFv0.Add(135);
1614 ListFv0.Add(135);
1615 ListFv0.Add(135);
1616 ListFv0.Add(135);
1617
1618 //CONCRETE PROPERTIES
1619 double fc = 3000; //psi
1620 double Ec = 57000 * Math.Sqrt(fc / 1000); //psi
1621 double Ic = 1 / 12 * 12 * Math.Pow(tConc, 3);
1622
1623 //GEOMETRY PARAMETERS
1624 double nSpan;
1625 double span = Math.Min(SpanX, SpanY);
1626
1627 //SLAB SPAN DIRECTION
1628 int dir;
1629 if (SpanX <= SpanY)
1630 {
1631     dir = 2;
1632 }
1633 else
1634 {
1635     dir = 1;
1636 }
1637 SlabDir = dir;
1638
1639 //LOOP THROUGH POSSIBLE CLT SLAB DEPTHS INCREASING DEPTH EACH LOOP
1640 for (int e = 0; e < 11; e++)
1641 {
1642     if (fire == false)
1643     {
1644         depth = ListDepth[e];
1645         Mmax = ListMmax[e];
1646         Vmax = ListVmax[e];
1647     }
1648     else
1649     {
1650         depth = ListDepth[e] - chardepth;

```

```

1651 Mmax = ListFb0[e] * 12 * Math.Pow(depth, 2) / 6;
1652 Vmax = ListFv0[e] * 2 * 12 * depth / 3;
1653 }
1654
1655 if (cSlab == false)
1656 {
1657     //NON-COMPOSITE SLABS
1658     A = 0;
1659     Vcap = Vmax * Cm * Ct * KShear * PhiShear * lambda;
1660     Vc = 0.75 * 2 * 1 * Math.Sqrt(fc) * 12 * tConc;
1661     Vtot = Vcap + Vc;
1662     Mcap = Mmax * Cm * Ct * Cl * KBend * PhiBend * lambda;
1663     nSpan = Math.Floor(ListPanelLength[e] / span);
1664
1665     TypFactLoad = TypSIL + DLFact * ListSW[e];
1666     TypDL = TypSDL + ListSW[e];
1667
1668     //SINGLE SPAN
1669     if (nSpan == 1)
1670     {
1671         Mdem = TypFactLoad * Math.Pow(span, 2) / 8;
1672         Vdem = TypFactLoad * span / 2;
1673         Ddl = 5 * (TypDL + TypLL) * Math.Pow(span, 4) * 12 /
1674             (384 * ListEI[e]);
1675         Dl = 5 * (TypLL) * Math.Pow(span, 4) * 12 / (384 *
1676             ListEI[e]);
1677     }
1678     //DOUBLE SPAN
1679     else if (nSpan == 2)
1680     {
1681         Mdem = 0.125 * TypFactLoad * Math.Pow(span, 2);
1682         Vdem = 0.625 * TypFactLoad * span;
1683         Ddl = (TypDL + TypLL) * Math.Pow(span, 4) * 12 / (185 *
1684             ListEI[e]);
1685         Dl = TypLL * Math.Pow(span, 4) * 12 / (185 * ListEI[e]);
1686     }
1687     //TRIPLE SPAN
1688     else
1689     {
1690         Mdem = 0.1 * TypFactLoad * Math.Pow(span, 2);
1691         Vdem = 0.6 * TypFactLoad * span;
1692         Ddl = 0.0069 * (TypDL + TypLL) * Math.Pow(span, 4) * 12
1693             / ListEI[e];
1694         Dl = 0.0069 * TypLL * Math.Pow(span, 4) * 12 / ListEI[e];
1695     }
1696
1697     //CHECK VIBRATIONS
1698     EIapp = 1 / ((1 / ListEI[e]) + (11.52 / (ListGA[e] *
1699         Math.Pow((12 * span), 2))));
1700     vibSpan = Math.Pow(EIapp, 0.293) / (12.05 * Math.Pow((rho *
1701         ListDepth[e] * 12), 0.122)); //CLT HANDBOOK CH7-3.4
1702
1703     if (Mdem < Mcap & Vdem < Vcap & Ddl < (span / 240) & Dl <
1704         (span / 360) & span < vibSpan)
1705     {
1706         A = 1;
1707     }
1708     else

```

```

1702     {
1703         A = 0;
1704     }
1705
1706     //SLAB PASSES WRITE OUTPUT
1707     if (A == 1)
1708     {
1709         tSlab = ListDepth[e];
1710         SlabSW = ListSW[e];
1711         pass = true;
1712         break;
1713     }
1714     else
1715     {
1716         if (e == 10)
1717         {
1718             tSlab = ListDepth[e];
1719             SlabSW = ListSW[e];
1720             pass = false;
1721         }
1722     }
1723     //SLAB FAILS LOOP
1724 }
1725 else
1726 {
1727     //DESIGN COMPOSITE SLABS
1728     A = 0;
1729     Vcap = ListFv0[e] * Cm * Ct * KShear * PhiShear * lambda;
1730     //ADJUSTED SHEAR STRESS
1731     Mcap = ListFb0[e] * Cm * Ct * Cl * KBend * PhiBend *
1732     lambda; //ADJUSTED BENDING STRESS
1733     nSpan = Math.Floor(ListPanelLength[e] / span);
1734
1735     TypFactLoad = TypSIL + DLFact * ListSW[e];
1736     TypDL = TypSDL + ListSW[e];
1737
1738     //NO CHARRING
1739     yC = tConc / 2 + ListDepth[e];
1740     yCLT = ListDepth[e] / 2;
1741     yNA = (tConc * 12 * Ec * yC + ListDepth[e] * 12 * ECLT) /
1742     (tConc * 12 * Ec + ListDepth[e] * 12 * ECLT);
1743
1744     n = Ec / ECLT;
1745     Itr = 1 * 12 * n * Math.Pow(tConc, 3) / (12) + tConc * 12 *
1746     Math.Pow((yC - yNA), 2) * n + 1 * 12 *
1747     Math.Pow(ListDepth[e], 3) / 12 + ListDepth[e] * 12 *
1748     Math.Pow((yCLT - yNA), 2);
1749
1750     //CHARRING
1751     yCchar = tConc / 2 + depth;
1752     yCLTchar = depth / 2;
1753     yNAchar = (tConc * 12 * Ec * yC + depth * 12 * ECLT) /
1754     (tConc * 12 * Ec + depth * 12 * ECLT);
1755
1756     Itrchar = 1 * 12 * Math.Pow(tConc, 3) * n / (12) + tConc *
1757     12 * Math.Pow((yCchar - yNAchar), 2) * n + 1 * 12 *
1758     Math.Pow(depth, 3) / 12 + depth * 12 * Math.Pow((yCLTchar -
1759     yNAchar), 2);

```

```

1750
1751     if (fire == false)
1752     {
1753         yCchar = yC;
1754         yCLTchar = yCLT;
1755         yNAchar = yNA;
1756         Itrchar = Itr;
1757     }
1758
1759     if (nSpan == 1)
1760     {
1761         Mdem = TypFactLoad * Math.Pow(span, 2) / 8;
1762         Vdem = 3 * TypFactLoad * span / (2 * 2 * (depth +
1763         tConc) * 12); //fv = 3wL/4bd
1764         Ddl = 5 * (TypDL + TypLL) * Math.Pow(span, 4) * 12 /
1765         (384 * ECLT * Itr);
1766         Dl = 5 * (TypLL) * Math.Pow(span, 4) * 12 / (384 * ECLT
1767         * Itr);
1768     }
1769     else if (nSpan == 2)
1770     {
1771         Mdem = 0.125 * TypFactLoad * Math.Pow(span, 2);
1772         Vdem = 3 * 0.625 * TypFactLoad * span / (2 * (depth +
1773         tConc) * 12);
1774         Ddl = (TypDL + TypLL) * Math.Pow(span, 4) * 12 / (185 *
1775         ECLT * Itr);
1776         Dl = TypLL * Math.Pow(span, 4) * 12 / (185 * ECLT * Itr);
1777     }
1778     else
1779     {
1780         Mdem = 0.1 * TypFactLoad * Math.Pow(span, 2);
1781         Vdem = 3 * 0.6 * TypFactLoad * span / (2 * (depth +
1782         tConc) * 12);
1783         Ddl = 0.0069 * (TypDL + TypLL) * Math.Pow(span, 4) * 12
1784         / (ECLT * Itr);
1785         Dl = 0.0069 * TypLL * Math.Pow(span, 4) * 12 / (ECLT *
1786         Itr);
1787     }
1788
1789     sigmaConc = -Mdem * (yCchar - yNAchar) * Ec / (Ec * Ic +
1790     (ECLT * 12 * Math.Pow(depth, 3) / 12));
1791     sigmaCLT = -Mdem * (yCLTchar - yNAchar) * ECLT / (Ec * Ic +
1792     (ECLT * 12 * Math.Pow(depth, 3) / 12));
1793
1794     //CHECK VIBRATIONS
1795     sg = ((rhoconc * tConc * 144 + rhoCLT * ListDepth[e] * 144)
1796     / (tConc * 144 + ListDepth[e] * 144)) / rhoh20;
1797     EIapp = 1 / (((1 / (ECLT * Itr)) + (11.52 / (ListGA[e] *
1798     Math.Pow((12 * span), 2)))));
1799     vibSpan = Math.Pow(EIapp, 0.293) / (12.05 * Math.Pow((sg *
1800     (tConc + ListDepth[e] * 12), 0.122))); //CLT HANDBOOK CH7-3.4
1801
1802     if (sigmaCLT < Mcap & sigmaConc < fc & Vdem < Vcap & Ddl <
1803     (span / 240) & Dl < (span / 360) & span < vibSpan)
1804     {
1805         A = 1;
1806     }
1807     else

```

```

1794     {
1795         A = 0;
1796     }
1797
1798     if (A == 1)
1799     {
1800         tSlab = ListDepth[e];
1801         SlabSW = ListSW[e];
1802         pass = true;
1803         break;
1804     }
1805     else
1806     {
1807         if (e == 10)
1808         {
1809             tSlab = ListDepth[e];
1810             SlabSW = ListSW[e];
1811             pass = false;
1812         }
1813     }
1814 }
1815 } //END LOOP THROUGH SLAB DEPTHS
1816 }
1817 }
1818
1819 //-----
1820 //-----
1821 // CHECK WALLS FOR AXIAL LOADS & ADD COLUMNS AS REQUIRED
1822 //-----
1823 //-----
1824
1825     using Rhino;
1826     using Rhino.Geometry;
1827     using Rhino.DocObjects;
1828     using Rhino.Collections;
1829
1830     using GH_IO;
1831     using GH_IO.Serialization;
1832     using Grasshopper;
1833     using Grasshopper.Kernel;
1834     using Grasshopper.Kernel.Data;
1835     using Grasshopper.Kernel.Types;
1836
1837     using System;
1838     using System.IO;
1839     using System.Xml;
1840     using System.Xml.Linq;
1841     using System.Linq;
1842     using System.Data;
1843     using System.Drawing;
1844     using System.Reflection;
1845     using System.Collections;
1846     using System.Windows.Forms;
1847     using System.Collections.Generic;

```

```

1848     using System.Runtime.InteropServices;
1849
1850     public class Script_Instance : GH_ScriptInstance
1851     {
1852         #region Utility functions
1853         private void Print(string text) { /* Implementation hidden. */ }
1854         private void Print(string format, params object[] args) { /*
1855             Implementation hidden. */ }
1856         private void Reflect(object obj) { /* Implementation hidden. */ }
1857         private void Reflect(object obj, string method_name) { /*
1858             Implementation hidden. */ }
1859         #endregion
1860
1861         #region Members
1862         private readonly RhinoDoc RhinoDocument;
1863         private readonly GH_Document GrasshopperDocument;
1864         private readonly IGH_Component Component;
1865         private readonly int Iteration;
1866         #endregion
1867
1868         private void RunScript(List<Line> WallLoc, List<Vector3d> RF,
1869             List<Point3d> Pos, List<double> VWall, List<double> MWall, double
1870             tWall, double Ht1, double HtTyp, double nFloors, double lambda, ref
1871             object WallColOut, ref object WallDC)
1872         {
1873             // INITIALIZE VARIABLES
1874             double LWall, PWall, PCap, PEq;
1875             double z0, z1;
1876             double phiComp = 0.9;
1877             double KComp = 2.4;
1878             List<double> ListPWall = new List<double>();
1879             List<double> DC = new List<double>();
1880             List<Line> WallColInt = new List<Line>();
1881
1882             // PANEL ALLOWABLE AXIAL LOAD FROM STRUCTURLAM
1883             double Pr = 0; // LB/FT
1884             if (Ht1 <= 16)
1885             {
1886                 if (tWall == 6.66)
1887                 {
1888                     Pr = 34344;
1889                 }
1890                 else if (tWall == 9.42)
1891                 {
1892                     Pr = 61864;
1893                 }
1894                 else if (tWall == 12.18)
1895                 {
1896                     Pr = 84689;
1897                 }
1898             }
1899             else if (Ht1 <= 18)
1900             {
1901                 if (tWall == 6.66)
1902                 {
1903                     Pr = 29124;
1904                 }
1905                 else if (tWall == 9.42)

```

```

1901     {
1902         Pr = 57440;
1903     }
1904     else if (tWall == 12.18)
1905     {
1906         Pr = 82206;
1907     }
1908 }
1909 else if (Ht1 <= 20)
1910 {
1911     if (tWall == 6.66)
1912     {
1913         Pr = 24667;
1914     }
1915     else if (tWall == 9.42)
1916     {
1917         Pr = 52187;
1918     }
1919     else if (tWall == 12.18)
1920     {
1921         Pr = 78947;
1922     }
1923 }
1924
1925 // LOOP OVER EACH WALL, CHECKING FOR COMBINED AXIAL LOADING SEISMIC
LOADING
1926 for (int i = 0; i < WallLoc.Count(); i++)
1927 {
1928     LWall = WallLoc[i].Length; // FT
1929     PWall = 0;
1930     PEq = MWall[i] * 1000 * 12 * 0.5 * LWall * 12 / (tWall *
Math.Pow(LWall * 12, 3) / 12); // PSI
1931
1932     for (int j = 0; j < Pos.Count(); j++)
1933     {
1934         if (WallLoc[i].DistanceTo(Pos[j], true) < 0.1)
1935         {
1936             PWall = PWall + RF[j].Z; // K
1937         }
1938     }
1939
1940     ListPWall.Add(PWall);
1941     PWall = PWall * 1000 / (LWall * 12 * tWall) + PEq; // PSI
1942     PCap = phiComp * KComp * Pr / (12 * tWall); // PSI
1943     DC.Add(PWall / PCap);
1944
1945     if (DC[i] > 1)
1946     {
1947         // ADD COLUMNS TO TAKE GRAVITY LOADS
1948         foreach (Point3d v in Pos)
1949         {
1950             if (WallLoc[i].DistanceTo(v, true) < 0.1)
1951             {
1952                 for (double k = 0; k < nFloors; k += 1)
1953                 {
1954                     if (k == 0)
1955                     {
1956                         z0 = 0;

```

```

1957         z1 = Ht1;
1958     }
1959     else
1960     {
1961         z0 = Ht1 + (k - 1) * HtTyp;
1962         z1 = Ht1 + k * HtTyp;
1963     }
1964     WallColInt.Add(new Line(new Point3d(v.X, v.Y,
z0), new Point3d(v.X, v.Y, z1)));
1965 }
1966 }
1967 }
1968 }
1969 }
1970
1971     WallColOut = WallColInt;
1972     WallDC = DC;
1973 }
1974 }
1975
1976 //-----
1977 //-----
1978 // SIZE BEAMS AND COLUMNS
1979 //-----
1980 //-----
1981
1982     using Rhino;
1983     using Rhino.Geometry;
1984     using Rhino.DocObjects;
1985     using Rhino.Collections;
1986
1987     using GH_IO;
1988     using GH_IO.Serialization;
1989     using Grasshopper;
1990     using Grasshopper.Kernel;
1991     using Grasshopper.Kernel.Data;
1992     using Grasshopper.Kernel.Types;
1993
1994     using System;
1995     using System.IO;
1996     using System.Xml;
1997     using System.Xml.Linq;
1998     using System.Linq;
1999     using System.Data;
2000     using System.Drawing;
2001     using System.Reflection;
2002     using System.Collections;
2003     using System.Windows.Forms;
2004     using System.Collections.Generic;
2005     using System.Runtime.InteropServices;
2006
2007     public class Script_Instance : GH_ScriptInstance
2008     {
2009         #region Utility functions

```

```

2010 private void Print(string text) { /* Implementation hidden. */ }
2011 private void Print(string format, params object[] args) { /*
Implementation hidden. */ }
2012 private void Reflect(object obj) { /* Implementation hidden. */ }
2013 private void Reflect(object obj, string method_name) { /*
Implementation hidden. */ }
2014 #endregion
2015
2016 #region Members
2017 private readonly RhinoDoc RhinoDocument;
2018 private readonly GH_Document GrasshopperDocument;
2019 private readonly IGH_Component Component;
2020 private readonly int Iteration;
2021 #endregion
2022
2023 private void RunScript(List<double> Nmax, List<double> Nmin,
List<double> Vzmax, List<double> Vzmin, List<double> Vymax,
List<double> Vymin, List<double> Mtmax, List<double> Mtmin,
List<double> Mymax, List<double> Mymin, List<double> Mzmax,
List<double> Mzmin, double FracDLFloor, double FracLLFloor, double
lambda, List<double> Length, List<Point3d> StartPt, List<Point3d>
EndPt, DataTree<double> GLBeams, DataTree<double> GLColumns, double
nElem, bool fire, bool cBeam, double tCLTslab, double tCLTroof, double
htActual, ref object width, ref object depth, ref object MaxBmD, ref
object ifBeam, ref object Error)
{
2024
2025
2026     double Cm = 1;
2027     double Ct = 1;
2028     double Cp = 1;
2029     double CComp = 0.9;
2030     double KComp = 2.4;
2031     double PhiComp = 0.9;
2032     double Cl = 1;
2033     double Cv = 1;
2034     double Cfu = 1;
2035     double Cc = 1;
2036     double Ci = 1;
2037     double KBend = 2.54;
2038     double PhiBend = 0.85;
2039     double Cvr = 1;
2040     double KShear = 2.88;
2041     double PhiShear = 0.75;
2042
2043     bool Flag = false;
2044     double DC, DCNComp, DCNTens, DCMyPos, DCMyNeg, DCMz, DCVy, DCVz;
2045     List<double> DCList;
2046     int Fail, A, B, C, Section;
2047     double NCompDem, NTensDem, MyPosDem, MyNegDem, MyDem, MzDem, VzDem,
VyDem, MtDem;
2048     double NCompAdj, NTensAdj, MyPosAdj, MyNegAdj, MzAdj, VzAdj, VyAdj;
2049     double NCompAll, NTensAll, MyPosAll, MyNegAll, MzAll, VzAll, VyAll;
2050     List<double> wOut = new List<double>();
2051     List<double> dOut = new List<double>();
2052     double wIt, dIt;
2053     double DeflFact;
2054     double FcE, FcStar, KELD, Z, time, fc, fb1, fb2, FcPrime, Fb1Prime,
Emin1Prime, Le1, FcE1, Fb2Prime, Emin2Prime, Le2, FcE2, RB1, RB2,

```

```

2055     FbE, EQ393, EQ394, DeflL, DeflDL;
2056     bool Beam, roof;
2057     Point3d pts, pte;
2058     List<bool> BeamOut = new List<bool>();
2059
2060     double fbGLB = 2400; //psi
2061     double ECLT = 1400000; //psi
2062     double EGLB = 1800000; //psi
2063     double yNA, yCLT, yGLB, ICLT, IGLB, Itr, n, bEff;
2064     double tCurr;
2065
2066     List<double> debug = new List<double>();
2067
2068     //ITERATE THROUGH ELEMENTS
2069     for (int i = 0; i < nElem; i++)
2070     {
2071         if (Nmin[i] < 0)
2072         {
2073             NCompDem = Math.Abs(Nmin[i]);
2074         }
2075         else
2076         {
2077             NCompDem = 0;
2078         }
2079         if (Nmax[i] > 0)
2080         {
2081             NTensDem = Math.Abs(Nmax[i]);
2082         }
2083         else
2084         {
2085             NTensDem = 0;
2086         }
2087         if (Mymin[i] < 0)
2088         {
2089             MyPosDem = Math.Abs(Mymin[i]);
2090         }
2091         else
2092         {
2093             MyPosDem = 0;
2094         }
2095         if (Mymax[i] > 0)
2096         {
2097             MyNegDem = Math.Abs(Mymax[i]);
2098         }
2099         else
2100         {
2101             MyNegDem = 0;
2102         }
2103         MyDem = Math.Max(MyPosDem, MyNegDem);
2104         MzDem = Math.Max(Math.Abs(Mzmax[i]), Math.Abs(Mzmin[i]));
2105         VzDem = Math.Max(Math.Abs(Vzmax[i]), Math.Abs(Vzmin[i]));
2106         VyDem = Math.Max(Math.Abs(Vymax[i]), Math.Abs(Vymin[i]));
2107         MtDem = Math.Max(Math.Abs(Mtmax[i]), Math.Abs(Mtmin[i]));
2108
2109         pts = StartPt[i];
2110         pte = EndPt[i];
2111         if (pts.Z == pte.Z)

```

```

2112     Beam = true;
2113 }
2114 else
2115 {
2116     Beam = false;
2117 }
2118 BeamOut.Add(Beam);
2119
2120 if (Beam == true)
2121 {
2122     Fail = 0;
2123     A = 0;
2124     B = 0;
2125     C = 0;
2126     Section = 0;
2127
2128     if (cBeam == false)
2129     {
2130         while (Fail < 3)
2131         {
2132             //BEAM DESIGN
2133
2134             foreach (double v in GLBeams.Branch(5))
2135             {
2136                 if (MyDem <= v)
2137                 {
2138                     break;
2139                 }
2140                 Section = Section + 1;
2141             }
2142
2143             //SECTION PROPERTIES
2144             wIt = GLBeams.Branch(0)[Section];
2145             dIt = GLBeams.Branch(1)[Section];
2146             NCompAll = GLBeams.Branch(2)[Section];
2147             NTensAll = GLBeams.Branch(3)[Section];
2148             MyPosAll = GLBeams.Branch(4)[Section];
2149             MyNegAll = GLBeams.Branch(5)[Section];
2150             MzAll = GLBeams.Branch(6)[Section];
2151             VzAll = GLBeams.Branch(7)[Section];
2152             VyAll = GLBeams.Branch(8)[Section];
2153
2154             //GENERATE PROPERTY SPECIFIC FACTORS
2155             FcE = 0.822 * 0.85 * Math.Pow(10, 6) /
Math.Pow((Length[i] * 12 / dIt), 2);
2156             FcStar = 1650 * Cm * Ct * KComp * PhiComp * lambda;
2157             if (NCompDem == 0)
2158             {
2159                 Cp = 1;
2160             }
2161             else
2162             {
2163                 Cp = ((1 + (FcE / FcStar)) / (2 * CComp)) -
Math.Sqrt(Math.Pow(((1 + (FcE / FcStar)) / (2 *
CComp)), 2) - (FcE / FcStar) / CComp);
2164             }
2165             Cv = Math.Min(1, Math.Pow((21 / Length[i]), 0.1) *
Math.Pow((12 / dIt), 0.1) * Math.Pow((5.125 /

```

```

2166     Math.Min(wIt, 10.75)), 0.1));
2167     if (MzDem > 1 && wIt < 12)
2168     {
2169         Cfu = Math.Pow((12 / wIt), (1 / 9));
2170     }
2171     else
2172     {
2173         Cfu = 1;
2174     }
2175
2176     //ADJUST DEMANDS
2177     NCompAdj = NCompDem / (Cm * Ct * Cp * KComp *
PhiComp * lambda);
2178     NTensAdj = NTensDem / (Cm * Ct * KComp * PhiComp *
lambda);
2179     MyPosAdj = MyPosDem / (Cm * Ct * Cl * Cv * Cfu * Cc
* Ci * KBend * PhiBend * lambda);
2180     MyNegAdj = MyNegDem / (Cm * Ct * Cl * Cv * Cfu * Cc
* Ci * KBend * PhiBend * lambda);
2181     MzAdj = MzDem / (Cm * Ct * Cl * Cv * Cfu * Cc * Ci
* KBend * PhiBend * lambda);
2182     VzAdj = VzDem / (Cm * Ct * Cvr * KShear * PhiShear
* lambda);
2183     VyAdj = VyDem / (Cm * Ct * Cvr * KShear * PhiShear
* lambda);
2184
2185     //CHECK D/C RATIOS
2186     DCList = new List<double>();
2187     DCNComp = NCompAdj / NCompAll;
2188     DCList.Add(DCNComp);
2189     DCNTens = NTensAdj / NTensAll;
2190     DCList.Add(DCNTens);
2191     DCMYPos = MyPosAdj / MyPosAll;
2192     DCList.Add(DCMYPos);
2193     DCMYNeg = MyNegAdj / MyNegAll;
2194     DCList.Add(DCMYNeg);
2195     DCMz = MzAdj / MzAll;
2196     DCList.Add(DCMz);
2197     DCVy = VyAdj / VyAll;
2198     DCList.Add(DCVy);
2199     DCVz = VzAdj / VzAll;
2200     DCList.Add(DCVz);
2201     DC = DCList.Max();
2202
2203     if (DC < 1)
2204     {
2205         A = 1;
2206     }
2207     else
2208     {
2209         A = 0;
2210     }
2211     //FIRE DESIGN
2212     KELD = 1 * Length[i] * 12 / dIt;
2213     if (DC <= 0.5)
2214     {
2215         if (KELD <= 11)
2216         {

```

```

2216         Z = 1.5;
2217     }
2218     else
2219     {
2220         Z = 1.3;
2221     }
2222 }
2223 else
2224 {
2225     if (KELD <= 11)
2226     {
2227         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC +
2228         2.1333;
2229     }
2230     else
2231     {
2232         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC +
2233         1.9333;
2234     }
2235 time = 2.54 * Z * wIt * (4 - wIt / dIt);
2236 if (fire == false)
2237 {
2238     B = 1;
2239 }
2240 else if (time < 60)
2241 {
2242     B = 0;
2243 }
2244 else
2245 {
2246     B = 1;
2247 }
2248 //NO COMBINED BENDING FOR BEAMS
2249 //CHECK BEAM DEFLECTIONS
2250 DeflFact = MyPosDem * 12 * 40 * Math.Pow((Length[i]
2251 * 12), 2) / (384 * 1800 * wIt * Math.Pow(dIt, 3) /
2252 12);
2253 DeflDL = (FracDLFloor + FracLLFloor) * DeflFact;
2254 DeflL = FracLLFloor * DeflFact;
2255 if (DeflDL < ((Length[i] * 12) / 240) && DeflL <
2256 ((Length[i] * 12) / 360))
2257 {
2258     C = 1;
2259 }
2260 else
2261 {
2262     C = 0;
2263 }
2264 if (A + B + C == 3)
2265 {
2266     wOut.Add(wIt);
2267     dOut.Add(dIt);
2268 }
2269 Fail = A + B + C;
2270 Section = Section + 1;
2271 if (Section > 56)

```

```

2269     {
2270         Flag = true;
2271         wOut.Add(0);
2272         dOut.Add(0);
2273         break;
2274     }
2275     else { }
2276 } //END BEAM SECTION WHILE LOOP
2277 }
2278 else
2279 {
2280     //COMPOSITE BEAM DESIGN
2281     while (Fail < 3)
2282     {
2283         //BEAM DESIGN
2284
2285         //SECTION PROPERTIES
2286         wIt = GLBeams.Branch(0)[Section];
2287         dIt = GLBeams.Branch(1)[Section];
2288         NCompAll = GLBeams.Branch(2)[Section];
2289         NTensAll = GLBeams.Branch(3)[Section];
2290         MyNegAll = GLBeams.Branch(5)[Section];
2291         MzAll = GLBeams.Branch(6)[Section];
2292         VyAll = GLBeams.Branch(8)[Section];
2293
2294         yGLB = dIt / 2;
2295         IGLB = 1 * wIt * Math.Pow(dIt, 3) / 12;
2296         n = ECLT / EGLB;
2297         if (pts.Z == htActual)
2298         {
2299             roof = true;
2300         }
2301         else
2302         {
2303             roof = false;
2304         }
2305
2306         if (roof == true)
2307         {
2308             tCurr = tCLTroof;
2309         }
2310         else
2311         {
2312             tCurr = tCLTslab;
2313         }
2314
2315         yCLT = dIt + tCurr / 2;
2316         bEff = Math.Min(0.25 * Length[i], 2 * 8 * tCurr +
2317 wIt);
2318         ICLT = 1 * bEff * Math.Pow(tCurr, 3) / 12;
2319         yNA = (tCurr * bEff * ECLT * yCLT + dIt * wIt *
2320 EGLB * yGLB) / (tCurr * bEff * ECLT + wIt * dIt *
2321 EGLB);
2322         Itr = 1 * wIt * Math.Pow(dIt, 3) / (12 * n) + wIt *
2323 dIt * Math.Pow((yGLB - yNA), 2) / n + 1 * bEff *
2324 Math.Pow(tCurr, 3) / 12 + bEff * tCurr *
2325 Math.Pow((yGLB - yNA), 2);

```

```

2321 MyPosAll = fbGLB * (ECLT * ICLT + EGLB * IGLB) /
      ((yCLT - yNA) * EGLB); //lb-in
2322 MyPosAll = MyPosAll / (1000 * 12); //K-ft
2323 VzAll = GLBeams.Branch(7)[Section];
2324
2325 //GENERATE PROPERTY SPECIFIC FACTORS
2326 FcE = 0.822 * 0.85 * Math.Pow(10, 6) /
      Math.Pow((Length[i] * 12 / dIt), 2);
2327 FcStar = 1650 * Cm * Ct * KComp * PhiComp * lambda;
2328 if (NCompDem == 0)
2329 {
2330     Cp = 1;
2331 }
2332 else
2333 {
2334     Cp = ((1 + (FcE / FcStar)) / (2 * CComp)) -
           Math.Sqrt(Math.Pow(((1 + (FcE / FcStar)) / (2 *
           CComp)), 2) - (FcE / FcStar) / CComp);
2335 }
2336 Cv = Math.Min(1, Math.Pow((21 / Length[i]), 0.1) *
      Math.Pow((12 / dIt), 0.1) * Math.Pow((5.125 /
      Math.Min(wIt, 10.75)), 0.1));
2337 if (MzDem > 1 && wIt < 12)
2338 {
2339     Cfu = Math.Pow((12 / wIt), (1 / 9));
2340 }
2341 else
2342 {
2343     Cfu = 1;
2344 }
2345
2346 //ADJUST DEMANDS
2347 NCompAdj = NCompDem / (Cm * Ct * Cp * KComp *
      PhiComp * lambda);
2348 NTensAdj = NTensDem / (Cm * Ct * KComp * PhiComp *
      lambda);
2349 MyPosAdj = MyPosDem / (Cm * Ct * Cl * Cv * Cfu * Cc
      * Ci * KBend * PhiBend * lambda);
2350 MyNegAdj = MyNegDem / (Cm * Ct * Cl * Cv * Cfu * Cc
      * Ci * KBend * PhiBend * lambda);
2351 MzAdj = MzDem / (Cm * Ct * Cl * Cv * Cfu * Cc * Ci
      * KBend * PhiBend * lambda);
2352 VzAdj = VzDem / (Cm * Ct * Cvr * KShear * PhiShear
      * lambda);
2353 VyAdj = VyDem / (Cm * Ct * Cvr * KShear * PhiShear
      * lambda);
2354
2355 //CHECK D/C RATIOS
2356 DCList = new List<double>();
2357 DCNComp = NCompAdj / NCompAll;
2358 DCList.Add(DCNComp);
2359 DCNTens = NTensAdj / NTensAll;
2360 DCList.Add(DCNTens);
2361 DCMYPos = MyPosAdj / MyPosAll;
2362 DCList.Add(DCMYPos);
2363 DCMYNeg = MyNegAdj / MyNegAll;
2364 DCList.Add(DCMYNeg);
2365 DCMz = MzAdj / MzAll;
2366
2367 DCList.Add(DCMz);
2368 DCVy = VyAdj / VyAll;
2369 DCList.Add(DCVy);
2370 DCVz = VzAdj / VzAll;
2371 DCList.Add(DCVz);
2372 DC = DCList.Max();
2373 if (DC < 1)
2374 {
2375     A = 1;
2376 }
2377 else
2378 {
2379     A = 0;
2380 }
2381 //FIRE DESIGN
2382 KELD = 1 * Length[i] * 12 / dIt;
2383 if (DC <= 0.5)
2384 {
2385     if (KELD <= 11)
2386     {
2387         Z = 1.5;
2388     }
2389     else
2390     {
2391         Z = 1.3;
2392     }
2393 }
2394 else
2395 {
2396     if (KELD <= 11)
2397     {
2398         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC +
2399         2.1333;
2400     }
2401     else
2402     {
2403         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC +
2404         1.9333;
2405     }
2406 }
2407 time = 2.54 * Z * wIt * (4 - wIt / dIt);
2408 if (fire == false)
2409 {
2410     B = 1;
2411 }
2412 else if (time < 60)
2413 {
2414     B = 0;
2415 }
2416 else
2417 {
2418     B = 1;
2419 }
2420 //NO COMBINED BENDING FOR BEAMS
2421
2422 //CHECK BEAM DEFLECTIONS
2423 DeflFact = MyPosDem * 12 * 40 * Math.Pow((Length[i]
      * 12), 2) / (384 * ECLT * Itr);

```

```

2421     DeflDL = (FracDLFloor + FracLLFloor) * DeflFact;
2422     DeflL = FracLLFloor * DeflFact;
2423     if (DeflDL < ((Length[i] * 12) / 240) && DeflL <
        ((Length[i] * 12) / 360))
2424     {
2425         C = 1;
2426     }
2427     else
2428     {
2429         C = 0;
2430     }
2431     if (A + B + C == 3)
2432     {
2433         wOut.Add(wIt);
2434         dOut.Add(dIt);
2435     }
2436     Fail = A + B + C;
2437     Section = Section + 1;
2438     if (Section == 56)
2439     {
2440         Flag = true;
2441         wOut.Add(0);
2442         dOut.Add(0);
2443         break;
2444     }
2445     else { }
2446 } //END COMPOSITE BEAM SECTION WHILE LOOP
2447 }
2448 } //End Beam If
2449 else
2450 {
2451     Fail = 0;
2452     A = 0;
2453     B = 0;
2454     C = 0;
2455     Section = 0;
2456     foreach (double v in GLColumns.Branch(2))
2457     {
2458         if (NCompDem <= v)
2459         {
2460             break;
2461         }
2462         Section = Section + 1;
2463     }
2464 }
2465 while (Fail < 3)
2466 {
2467     //SECTION PROPERTIES
2468     wIt = GLColumns.Branch(0)[Section];
2469     dIt = GLColumns.Branch(1)[Section];
2470     NCompAll = GLColumns.Branch(2)[Section];
2471     NTensAll = GLColumns.Branch(3)[Section];
2472     MyPosAll = GLColumns.Branch(4)[Section];
2473     MyNegAll = GLColumns.Branch(5)[Section];
2474     MzAll = GLColumns.Branch(6)[Section];
2475     VzAll = GLColumns.Branch(7)[Section];
2476     VyAll = GLColumns.Branch(8)[Section];
2477

```

```

2478 //GENERATE PROPERTY SPECIFIC FACTORS
2479 FcE = 0.822 * 0.85 * Math.Pow(10, 6) /
        Math.Pow((Length[i] * 12 / dIt), 2);
2480 FcStar = 1650 * Cm * Ct * KComp * PhiComp * lambda;
2481 if (NCompDem == 0)
2482 {
2483     Cp = 1;
2484 }
2485 else
2486 {
2487     Cp = ((1 + (FcE / FcStar)) / (2 * CComp)) -
        Math.Sqrt(Math.Pow(((1 + (FcE / FcStar)) / (2 *
        CComp)), 2) - (FcE / FcStar) / CComp);
2488 }
2489 Cv = Math.Min(1, Math.Pow((21 / Length[i]), 0.1) *
        Math.Pow((12 / dIt), 0.1) * Math.Pow((5.125 /
        Math.Min(wIt, 10.75)), 0.1));
2490 if (MzDem > 1 && wIt < 12)
2491 {
2492     Cfu = Math.Pow((12 / wIt), (1 / 9));
2493 }
2494 else
2495 {
2496     Cfu = 1;
2497 }
2498 }
2499 //ADJUST DEMANDS
2500 NCompAdj = NCompDem / (Cm * Ct * Cp * KComp * PhiComp *
        lambda);
2501 NTensAdj = NTensDem / (Cm * Ct * KComp * PhiComp *
        lambda);
2502 MyPosAdj = MyPosDem / (Cm * Ct * Cl * Cv * Cfu * Cc *
        Ci * KBend * PhiBend * lambda);
2503 MyNegAdj = MyNegDem / (Cm * Ct * Cl * Cv * Cfu * Cc *
        Ci * KBend * PhiBend * lambda);
2504 MzAdj = MzDem / (Cm * Ct * Cl * Cv * Cfu * Cc * Ci *
        KBend * PhiBend * lambda);
2505 VzAdj = VzDem / (Cm * Ct * Cvr * KShear * PhiShear *
        lambda);
2506 VyAdj = VyDem / (Cm * Ct * Cvr * KShear * PhiShear *
        lambda);
2507 }
2508 //CHECK D/C RATIOS
2509 DCList = new List<double>();
2510 DCNComp = NCompAdj / NCompAll;
2511 DCList.Add(DCNComp);
2512 DCNTens = NTensAdj / NTensAll;
2513 DCList.Add(DCNTens);
2514 DCMYPos = MyPosAdj / MyPosAll;
2515 DCList.Add(DCMYPos);
2516 DCMYNeg = MyNegAdj / MyNegAll;
2517 DCList.Add(DCMYNeg);
2518 DCMz = MzAdj / MzAll;
2519 DCList.Add(DCMz);
2520 DCVy = VyAdj / VyAll;
2521 DCList.Add(DCVy);
2522 DCVz = VzAdj / VzAll;
2523 DCList.Add(DCVz);

```

```

2524 DC = DCList.Max();
2525 if (DC < 1)
2526 {
2527     A = 1;
2528 }
2529 else
2530 {
2531     A = 0;
2532 }
2533 //FIRE DESIGN
2534 KELD = 1 * Length[i] * 12 / dIt;
2535 if (DC <= 0.5)
2536 {
2537     if (KELD <= 11)
2538     {
2539         Z = 1.5;
2540     }
2541     else
2542     {
2543         Z = 1.3;
2544     }
2545 }
2546 else
2547 {
2548     if (KELD <= 11)
2549     {
2550         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC + 2.1333;
2551     }
2552     else
2553     {
2554         Z = 2 * Math.Pow(DC, 2) / 3 - 1.6 * DC + 1.9333;
2555     }
2556 }
2557 time = 2.54 * Z * wIt * (3 - wIt / dIt);
2558 if (fire == false)
2559 {
2560     B = 1;
2561 }
2562 else if (time < 60)
2563 {
2564     B = 0;
2565 }
2566 else
2567 {
2568     B = 1;
2569 }
2570 //COMBINED BENDING
2571 if ((MyPosAdj / MyPosAll) < 0.01 && (MyNegAdj /
MyNegAll) < 0.01 && (MzAdj / MzAll) < 0.01)
2572 {
2573     //NO COMBINED BENDING
2574     C = 1;
2575 }
2576 else
2577 {
2578     fc = NCompDem * 1000 / (wIt * dIt);
2579     fb1 = Math.Max(MyPosDem, MyNegDem) * 1000 * 12 /
(wIt * Math.Pow(dIt, 2) / 6);

```

```

2580 fb2 = MzDem * 1000 * 12 / (wIt * Math.Pow(dIt, 2) /
6);
2581 FcPrime = NCompAll * 1000 * (Cm * Ct * Cp * KComp *
PhiComp * lambda) / (wIt * dIt);
2582 Fb1Prime = MyPosAll * 1000 * (Cm * Ct * Cl * Cv *
Cfu * Cc * Ci * KBend * PhiBend * lambda) / (wIt *
Math.Pow(dIt, 2) / 6);
2583 Emin1Prime = 0.95 * Cm * Ct * 1.76 * 0.85 *
Math.Pow(10, 6);
2584 if ((Length[i] * 12 / dIt) < 7)
2585 {
2586     Le1 = 1.87 * Length[i] * 12;
2587 }
2588 else
2589 {
2590     Le1 = 1.44 * Length[i] * 12 + 3 * dIt;
2591 }
2592 FcE1 = 0.822 * Emin1Prime / Math.Pow((Le1 / dIt), 2);
2593 Fb2Prime = MzDem * 1000 * 12 * (Cm * Ct * Cl * Cv *
Cfu * Cc * Ci * KBend * PhiBend * lambda) / (dIt *
Math.Pow(wIt, 2) / 6);
2594 Emin2Prime = 0.85 * Cm * Ct * 1.76 * 0.85 *
Math.Pow(10, 6);
2595 if ((Length[i] * 12 / wIt) < 7)
2596 {
2597     Le2 = 1.87 * Length[i] * 12;
2598 }
2599 else
2600 {
2601     Le2 = 1.44 * Length[i] * 12 + 3 * wIt;
2602 }
2603 FcE2 = 0.822 * Emin2Prime / Math.Pow((Le2 / wIt), 2);
2604 RB1 = Math.Sqrt(Le1 * dIt / Math.Pow(wIt, 2));
2605 RB2 = Math.Sqrt(Le2 * wIt / Math.Pow(dIt, 2));
2606 FbE = Math.Min(1.2 * Emin1Prime / Math.Pow(RB1, 2),
1.2 * Emin2Prime / Math.Pow(RB2, 2));
2607 EQ393 = Math.Pow(fc / FcPrime, 2) + fb1 / (Fb1Prime
* (1 - (fc / FcE1))) + fb2 / (Fb2Prime * (1 - (fc /
FcE2) - Math.Pow((fb1 / FbE), 2)));
2608 EQ394 = fc / FcE2 + Math.Pow((fb1 / FbE), 2);
2609
2610 if (EQ393 < 1 && EQ394 < 1)
2611 {
2612     C = 1;
2613 }
2614 else
2615 {
2616     C = 0;
2617 }
2618 }
2619 if (A + B + C == 3)
2620 {
2621     wOut.Add(wIt);
2622     dOut.Add(dIt);
2623 }
2624 Fail = A + B + C;
2625 Section = Section + 1;
2626 if (Section == 289)

```

```

2627         {
2628             Flag = true;
2629             wOut.Add(0);
2630             dOut.Add(0);
2631             break;
2632         }
2633         else { }
2634     } //END COLUMN SECTION LOOP
2635 } //End Column If
2636 } //End Element Loop
2637
2638 double MBD = 0;
2639 for (int i = 0; i < BeamOut.Count(); i++)
2640 {
2641     if (BeamOut[i] == true)
2642     {
2643         if (MBD < dOut[i])
2644         {
2645             MBD = dOut[i];
2646         }
2647     }
2648 }
2649
2650 width = wOut;
2651 depth = dOut;
2652 MaxBmD = MBD;
2653 ifBeam = BeamOut;
2654 Error = Flag;
2655 }
2656 }
2657
2658 //-----
2659 //-----
2660 // CALCULATE LCA IMPACTS
2661 //-----
2662 //-----
2663 using Rhino;
2664 using Rhino.Geometry;
2665 using Rhino.DocObjects;
2666 using Rhino.Collections;
2667
2668 using GH_IO;
2669 using GH_IO.Serialization;
2670 using Grasshopper;
2671 using Grasshopper.Kernel;
2672 using Grasshopper.Kernel.Data;
2673 using Grasshopper.Kernel.Types;
2674
2675 using System;
2676 using System.IO;
2677 using System.Xml;
2678 using System.Xml.Linq;
2679 using System.Linq;
2680 using System.Data;

```

```

2681 using System.Drawing;
2682 using System.Reflection;
2683 using System.Collections;
2684 using System.Windows.Forms;
2685 using System.Collections.Generic;
2686 using System.Runtime.InteropServices;
2687
2688 public class Script_Instance : GH_ScriptInstance
2689 {
2690     #region Utility functions
2691     private void Print(string text) { /* Implementation hidden. */ }
2692     private void Print(string format, params object[] args) { /*
2693         Implementation hidden. */ }
2694     private void Reflect(object obj) { /* Implementation hidden. */ }
2695     private void Reflect(object obj, string method_name) { /*
2696         Implementation hidden. */ }
2697     #endregion
2698
2699     #region Members
2700     private readonly RhinoDoc RhinoDocument;
2701     private readonly GH_Document GrasshopperDocument;
2702     private readonly IGH_Component Component;
2703     private readonly int Iteration;
2704     #endregion
2705
2706     private void RunScript(double AreaCLTCoreWall, double AreaCLTFacWall,
2707         double tWall, double AreaCLTFloor, double tCLTFloor, double
2708         tConcreteFloor, double AreaCLTRoof, double tCLTRoof, double
2709         tConcreteRoof, List<double> LElem, List<double> tElem, List<double>
2710         dElem, List<bool> ifBeam, double XDimAct, double YDimAct, double HtAct,
2711         List<double> ACin, List<double> ADin, List<double> waste, ref object
2712         TotAtoC, ref object TotAtoD, ref object AtoCperSF, ref object
2713         AtoDperSF, ref object OutputTot, ref object OutputSF, ref object CVol,
2714         ref object BVol, ref object AFac)
2715     {
2716         //INITIALIZE VARIABLES
2717         double beamvol = 0;
2718         double colvol = 0;
2719         double slabCLTvol = 0;
2720         double roofCLTvol = 0;
2721         double slabConcVol = 0;
2722         double roofConcVol = 0;
2723         double wallCLTvol = 0;
2724         double AreaGlass = 0;
2725         double steelVol = 0;
2726         double ac = 0;
2727         double ad = 0;
2728         int matNo = 0;
2729         List<double> acIn = ACin;
2730         List<double> adIn = ADin;
2731         List<double> acOut = new List<double>();
2732         List<double> adOut = new List<double>();
2733         List<string> names = new List<string>();
2734
2735         //CALCULATE VOLUME OF GLULAM BEAMS AND COLUMNS
2736         for (int i = 0; i < LElem.Count; i++)
2737         {
2738             if (ifBeam[i] == true)

```

```

2729     {
2730         //ELEMENT IS A BEAM
2731         beamvol = beamvol + tElem[i] * dElem[i] * LElem[i] / 144;
                //FT3
2732     }
2733     else
2734     {
2735         //ELEMENT IS A COLUMN
2736         colvol = colvol + tElem[i] * dElem[i] * LElem[i] / 144; //FT3
2737     }
2738 }
2739
2740 //CALCULATE VOLUME OF CLT FLOOR SLABS
2741 slabCLTvol = AreaCLTFloor * tCLTFloor / 12; //FT3
2742 slabConcVol = AreaCLTFloor * tConcreteFloor / (12 * 27); //CY
2743 roofCLTvol = AreaCLTRoof * tCLTRoof / 12; //FT3
2744 roofConcVol = AreaCLTRoof * tConcreteRoof / (12 * 27); //CY
2745
2746 //CALCULATE VOLUME OF CLT SHEAR WALLS
2747 wallCLTvol = (AreaCLTCoreWall + AreaCLTFacWall) * tWall / 12;
2748 AreaGlass = 2 * (XDimAct + YDimAct) * HtAct - AreaCLTFacWall;
2749
2750 //IMPACTS BY MATERIAL/ASSEMBLY
2751 //GLULAM BEAMS
2752 matNo = 102; //GluLam Sections
2753 ac = ac + acIn[matNo] * beamvol * waste[matNo] / 100;
2754 ad = ad + adIn[matNo] * beamvol * waste[matNo] / 100;
2755 acOut.Add(acIn[matNo] * beamvol * waste[matNo] / 100);
2756 adOut.Add(adIn[matNo] * beamvol * waste[matNo] / 100);
2757 names.Add("GluLam Beams");
2758
2759 //GLULAM COLUMNS
2760 matNo = 102; //GluLam Sections
2761 ac = ac + acIn[matNo] * colvol * waste[matNo] / 100;
2762 ad = ad + adIn[matNo] * colvol * waste[matNo] / 100;
2763 acOut.Add(acIn[matNo] * colvol * waste[matNo] / 100);
2764 adOut.Add(adIn[matNo] * colvol * waste[matNo] / 100);
2765 names.Add("GluLam Columns");
2766
2767 //CLT FLOOR SLABS
2768 matNo = 54; //Cross Laminated Timber
2769 ac = ac + acIn[matNo] * slabCLTvol * waste[matNo] / 100;
2770 ad = ad + adIn[matNo] * slabCLTvol * waste[matNo] / 100;
2771 acOut.Add(acIn[matNo] * slabCLTvol * waste[matNo] / 100);
2772 adOut.Add(adIn[matNo] * slabCLTvol * waste[matNo] / 100);
2773 names.Add("CLT Floors");
2774
2775 //FLOOR SLAB UNDERLAYMENT
2776 matNo = 145; //Polyiso Foam Board
2777 ac = ac + acIn[matNo] * AreaCLTFloor * waste[matNo] / 100;
2778 ad = ad + adIn[matNo] * AreaCLTFloor * waste[matNo] / 100;
2779 acOut.Add(acIn[matNo] * AreaCLTFloor * waste[matNo] / 100);
2780 adOut.Add(adIn[matNo] * AreaCLTFloor * waste[matNo] / 100);
2781 names.Add("Floor Underlayment");
2782
2783 //FLOOR SLAB CONCRETE TOPPING
2784 matNo = 47; //Concrete Benchmark 3000 psi
2785 ac = ac + acIn[matNo] * slabConcVol * waste[matNo] / 100;

```

```

2786 ad = ad + adIn[matNo] * slabConcVol * waste[matNo] / 100;
2787 acOut.Add(acIn[matNo] * slabConcVol * waste[matNo] / 100);
2788 adOut.Add(adIn[matNo] * slabConcVol * waste[matNo] / 100);
2789 names.Add("Floor Concrete Topping");
2790
2791 //CLT ROOF SLAB
2792 matNo = 54; //Cross Laminated Timber
2793 ac = ac + acIn[matNo] * roofCLTvol * waste[matNo] / 100;
2794 ad = ad + adIn[matNo] * roofCLTvol * waste[matNo] / 100;
2795 acOut.Add(acIn[matNo] * roofCLTvol * waste[matNo] / 100);
2796 adOut.Add(adIn[matNo] * roofCLTvol * waste[matNo] / 100);
2797 names.Add("CLT Roof");
2798
2799 //ROOF SLAB CONCRETE TOPPING
2800 matNo = 47; //Concrete Benchmark 3000 psi
2801 ac = ac + acIn[matNo] * roofConcVol * waste[matNo] / 100;
2802 ad = ad + adIn[matNo] * roofConcVol * waste[matNo] / 100;
2803 acOut.Add(acIn[matNo] * roofConcVol * waste[matNo] / 100);
2804 adOut.Add(adIn[matNo] * roofConcVol * waste[matNo] / 100);
2805 names.Add("Roof Concrete Topping");
2806
2807 //ROOF SLAB MEMBRANE
2808 matNo = 9; //3 mil Polyethylene
2809 ac = ac + acIn[matNo] * AreaCLTRoof * waste[matNo] / 100;
2810 ad = ad + adIn[matNo] * AreaCLTRoof * waste[matNo] / 100;
2811 acOut.Add(acIn[matNo] * AreaCLTRoof * waste[matNo] / 100);
2812 adOut.Add(adIn[matNo] * AreaCLTRoof * waste[matNo] / 100);
2813 names.Add("Roof Membrane");
2814
2815 //ROOF SLAB INSULATION
2816 matNo = 69; //Extruded Polystyrene
2817 ac = ac + acIn[matNo] * AreaCLTRoof * 6 * waste[matNo] / 100;
2818 ad = ad + adIn[matNo] * AreaCLTRoof * 6 * waste[matNo] / 100;
2819 acOut.Add(acIn[matNo] * AreaCLTRoof * 6 * waste[matNo] / 100);
2820 adOut.Add(adIn[matNo] * AreaCLTRoof * 6 * waste[matNo] / 100);
2821 names.Add("Roof Insulation");
2822
2823 //CLT WALL
2824 matNo = 54; //Cross Laminated Timber
2825 ac = ac + acIn[matNo] * wallCLTvol * waste[matNo] / 100;
2826 ad = ad + adIn[matNo] * wallCLTvol * waste[matNo] / 100;
2827 acOut.Add(acIn[matNo] * wallCLTvol * waste[matNo] / 100);
2828 adOut.Add(adIn[matNo] * wallCLTvol * waste[matNo] / 100);
2829 names.Add("CLT Wall");
2830
2831 //CORE WALL GWB
2832 matNo = 5; //1/2" Gypsum Board
2833 ac = ac + acIn[matNo] * 2 * AreaCLTCoreWall * waste[matNo] / 100;
2834 ad = ad + adIn[matNo] * 2 * AreaCLTCoreWall * waste[matNo] / 100;
2835 acOut.Add(acIn[matNo] * 2 * AreaCLTCoreWall * waste[matNo] / 100);
2836 adOut.Add(adIn[matNo] * 2 * AreaCLTCoreWall * waste[matNo] / 100);
2837 names.Add("Core Wall GWB");
2838
2839 //FACADE WALL WATERPROOFING
2840 matNo = 9; //3 mil Polyethylene
2841 ac = ac + acIn[matNo] * AreaCLTFacWall * waste[matNo] / 100;
2842 ad = ad + adIn[matNo] * AreaCLTFacWall * waste[matNo] / 100;
2843 acOut.Add(acIn[matNo] * AreaCLTFacWall * waste[matNo] / 100);

```

```

2844 adOut.Add(adIn[matNo] * AreaCLTFacWall * waste[matNo] / 100);
2845 names.Add("Facade Waterproofing");
2846
2847 //FACADE WALL INSULATION
2848 matNo = 69; //Extruded Polystyrene
2849 ac = ac + acIn[matNo] * 6 * AreaCLTFacWall * waste[matNo] / 100;
2850 ad = ad + adIn[matNo] * 6 * AreaCLTFacWall * waste[matNo] / 100;
2851 acOut.Add(acIn[matNo] * 6 * AreaCLTFacWall * waste[matNo] / 100);
2852 adOut.Add(adIn[matNo] * 6 * AreaCLTFacWall * waste[matNo] / 100);
2853 names.Add("Facade Insulation");
2854
2855 //FACADE WALL CLADDING
2856 matNo = 142; //Pine T&G Siding
2857 ac = ac + acIn[matNo] * AreaCLTFacWall * waste[matNo] / 100;
2858 ad = ad + adIn[matNo] * AreaCLTFacWall * waste[matNo] / 100;
2859 acOut.Add(acIn[matNo] * AreaCLTFacWall * waste[matNo] / 100);
2860 adOut.Add(adIn[matNo] * AreaCLTFacWall * waste[matNo] / 100);
2861 names.Add("Facade Cladding");
2862
2863 //CURTAIN WALL
2864 matNo = 56; //Double Glazed Windows
2865 ac = ac + acIn[matNo] * AreaGlass * waste[matNo] / 100;
2866 ad = ad + adIn[matNo] * AreaGlass * waste[matNo] / 100;
2867 acOut.Add(acIn[matNo] * AreaGlass * waste[matNo] / 100);
2868 adOut.Add(adIn[matNo] * AreaGlass * waste[matNo] / 100);
2869 names.Add("Curtain Wall");
2870
2871 //CURTAIN WALL ALUMINUM
2872 matNo = 24; //Aluminum Window Frame
2873 //From Athena 0.1262 Tons/100 SF of Standard Curtain
//Wall (includes waste)
2874 ac = ac + acIn[matNo] * 0.1262 * AreaGlass / (100 * 100);
2875 ad = ad + adIn[matNo] * 0.1262 * AreaGlass / (100 * 100);
2876 acOut.Add(acIn[matNo] * 0.1262 * AreaGlass / (100 * 100));
2877 adOut.Add(adIn[matNo] * 0.1262 * AreaGlass / (100 * 100));
2878 names.Add("Curtain Wall Mullions");
2879
2880 //TIMBER STEEL CONNECTIONS
2881 matNo = 171; //Steel Plate
2882 steelVol = 0.0025 * (beamvol + colvol + slabCLTvol + roofCLTvol +
wallCLTvol);
2883 //steel density 490 lb/ft^3
2884 ac = ac + acIn[matNo] * steelVol * 0.245 * waste[matNo] / 100;
2885 ad = ad + adIn[matNo] * steelVol * 0.245 * waste[matNo] / 100;
2886 acOut.Add(acIn[matNo] * steelVol * 0.245 * waste[matNo] / 100);
2887 adOut.Add(adIn[matNo] * steelVol * 0.245 * waste[matNo] / 100);
2888 names.Add("Steel Connections");
2889
2890 //Itemized Output
2891 DataTree<System.Object> LCAItems = new DataTree<System.Object>();
2892 DataTree<System.Object> LCAItemsSF = new DataTree<System.Object>();
2893 GH_Path A = new GH_Path(0);
2894 GH_Path B = new GH_Path(1);
2895 GH_Path C = new GH_Path(2);
2896
2897 for (int i = 0; i < names.Count; i++)
2898 {
2899     LCAItems.Add(names[i], A);

```

```

2900     LCAItemsSF.Add(names[i], A);
2901     LCAItems.Add(acOut[i], B);
2902     LCAItemsSF.Add(acOut[i] / (AreaCLTFloor + AreaCLTRoof), B);
2903     LCAItems.Add(adOut[i], C);
2904     LCAItemsSF.Add(adOut[i] / (AreaCLTFloor + AreaCLTRoof), C);
2905 }
2906
2907 //Output per SF
2908 double acSFOut = ac / (AreaCLTFloor + AreaCLTRoof);
2909 double adSFOut = ad / (AreaCLTFloor + AreaCLTRoof);
2910
2911 //OUTPUTS
2912 TotAtoC = ac;
2913 TotAtoD = ad;
2914 AtoCperSF = acSFOut;
2915 AtoDperSF = adSFOut;
2916 OutputTot = LCAItems;
2917 OutputSF = LCAItemsSF;
2918 CVol = colvol;
2919 BVol = beamvol;
2920 AFac = AreaGlass;
2921 }
2922 }
2923
2924 //-----
2925 //-----
2926 // CALCULATE COST IMPACTS
2927 //-----
2928 //-----
2929
2930 using Rhino;
2931 using Rhino.Geometry;
2932 using Rhino.DocObjects;
2933 using Rhino.Collections;
2934
2935 using GH_IO;
2936 using GH_IO.Serialization;
2937 using Grasshopper;
2938 using Grasshopper.Kernel;
2939 using Grasshopper.Kernel.Data;
2940 using Grasshopper.Kernel.Types;
2941
2942 using System;
2943 using System.IO;
2944 using System.Xml;
2945 using System.Xml.Linq;
2946 using System.Linq;
2947 using System.Data;
2948 using System.Drawing;
2949 using System.Reflection;
2950 using System.Collections;
2951 using System.Windows.Forms;
2952 using System.Collections.Generic;
2953 using System.Runtime.InteropServices;

```

```

2954
2955 public class Script_Instance : GH_ScriptInstance
2956 {
2957     #region Utility functions
2958     private void Print(string text) { /* Implementation hidden. */ }
2959     private void Print(string format, params object[] args) { /*
2960     Implementation hidden. */ }
2961     private void Reflect(object obj) { /* Implementation hidden. */ }
2962     private void Reflect(object obj, string method_name) { /*
2963     Implementation hidden. */ }
2964     #endregion
2965
2966     #region Members
2967     private readonly RhinoDoc RhinoDocument;
2968     private readonly GH_Document GrasshopperDocument;
2969     private readonly IGH_Component Component;
2970     private readonly int Iteration;
2971     #endregion
2972
2973     private void RunScript(double AreaCLTCoreWall, double AreaCLTFacWall,
2974     double tWall, double AreaCLTFloor, double tCLTFloor, double
2975     tConcreteFloor, double AreaCLTRoof, double tCLTRoof, double
2976     tConcreteRoof, List<double> LElem, List<double> tElem, List<double>
2977     dElem, List<bool> ifBeam, double XDimAct, double YDimAct, double HtAct,
2978     List<double> CostIn, ref object TotCost, ref object CostperSF, ref
2979     object OutputTot, ref object OutputSF)
2980     {
2981         //INITIALIZE VARIABLES
2982         double beamvol = 0;
2983         double colvol = 0;
2984         double slabCLTv1 = 0;
2985         double roofCLTv1 = 0;
2986         double slabConcVol = 0;
2987         double roofConcVol = 0;
2988         double wallCLTv1 = 0;
2989         double AreaGlass = 0;
2990         int matNo = 0;
2991         double tCost = 0;
2992         List<double> CostOut = new List<double>();
2993         List<string> names = new List<string>();
2994
2995         //CALCULATE VOLUME OF GLULAM BEAMS AND COLUMNS
2996         for (int i = 0; i < LElem.Count; i++)
2997         {
2998             if (ifBeam[i] == true)
2999             {
3000                 //ELEMENT IS A BEAM
3001                 beamvol = beamvol + tElem[i] * dElem[i] * LElem[i] / 144;
3002                 //FT3
3003             }
3004             else
3005             {
3006                 //ELEMENT IS A COLUMN
3007                 colvol = colvol + tElem[i] * dElem[i] * LElem[i] / 144; //FT3
3008             }
3009         }
3010
3011         //CALCULATE VOLUME OF CLT FLOOR SLABS

```

```

3003     slabCLTv1 = AreaCLTFloor * tCLTFloor / 12; //FT3
3004     slabConcVol = AreaCLTFloor * tConcreteFloor / (12 * 27); //CY
3005     roofCLTv1 = AreaCLTRoof * tCLTRoof / 12; //FT3
3006     roofConcVol = AreaCLTRoof * tConcreteRoof / (12 * 27); //CY
3007
3008     //CALCULATE VOLUME OF CLT SHEAR WALLS
3009     wallCLTv1 = (AreaCLTCoreWall + AreaCLTFacWall) * tWall / 12;
3010     AreaGlass = 2 * (XDimAct + YDimAct) * HtAct - AreaCLTFacWall;
3011
3012     //IMPACTS BY MATERIAL/ASSEMBLY
3013     //GLULAM BEAMS
3014     matNo = 0; //GluLam Sections
3015     tCost = tCost + CostIn[matNo] * beamvol;
3016     CostOut.Add(CostIn[matNo] * beamvol);
3017     names.Add("GluLam Beams");
3018
3019     //GLULAM COLUMNS
3020     matNo = 1; //GluLam Sections
3021     tCost = tCost + CostIn[matNo] * colvol;
3022     CostOut.Add(CostIn[matNo] * colvol);
3023     names.Add("GluLam Columns");
3024
3025     //CLT FLOOR SLABS
3026     matNo = 2; //Cross Laminated Timber
3027     tCost = tCost + CostIn[matNo] * slabCLTv1;
3028     CostOut.Add(CostIn[matNo] * slabCLTv1);
3029     names.Add("CLT Floors");
3030
3031     //FLOOR SLAB UNDERLAYMENT
3032     matNo = 3; //Polyiso Foam Board
3033     tCost = tCost + CostIn[matNo] * AreaCLTFloor;
3034     CostOut.Add(CostIn[matNo] * AreaCLTFloor);
3035     names.Add("Floor Underlayment");
3036
3037     //FLOOR SLAB CONCRETE TOPPING
3038     matNo = 4; //Concrete Benchmark 3000 psi
3039     tCost = tCost + CostIn[matNo] * AreaCLTFloor;
3040     CostOut.Add(CostIn[matNo] * AreaCLTFloor);
3041     names.Add("Floor Concrete Topping");
3042
3043     //CLT ROOF SLAB
3044     matNo = 5; //Cross Laminated Timber
3045     tCost = tCost + CostIn[matNo] * roofCLTv1;
3046     CostOut.Add(CostIn[matNo] * roofCLTv1);
3047     names.Add("CLT Roof");
3048
3049     //ROOF SLAB CONCRETE TOPPING
3050     matNo = 6; //Concrete Benchmark 3000 psi
3051     tCost = tCost + CostIn[matNo] * AreaCLTRoof;
3052     CostOut.Add(CostIn[matNo] * AreaCLTRoof);
3053     names.Add("Roof Concrete Topping");
3054
3055     //ROOF SLAB MEMBRANE
3056     matNo = 7; //3 mil Polyethylene
3057     tCost = tCost + CostIn[matNo] * AreaCLTRoof;
3058     CostOut.Add(CostIn[matNo] * AreaCLTRoof);
3059     names.Add("Roof Membrane");
3060

```

```

3061 //ROOF SLAB INSULATION
3062 matNo = 8; //Extruded Polystyrene
3063 tCost = tCost + CostIn[matNo] * AreaCLTRoof;
3064 CostOut.Add(CostIn[matNo] * AreaCLTRoof);
3065 names.Add("Roof Insulation");
3066
3067 //CLT WALL
3068 matNo = 9; //Cross Laminated Timber
3069 tCost = tCost + CostIn[matNo] * wallCLTvol;
3070 CostOut.Add(CostIn[matNo] * wallCLTvol);
3071 names.Add("CLT Wall");
3072
3073 //CORE WALL GWB
3074 matNo = 10; //1/2" Gypsum Board
3075 tCost = tCost + CostIn[matNo] * 2 * AreaCLTCoreWall;
3076 CostOut.Add(CostIn[matNo] * 2 * AreaCLTCoreWall);
3077 names.Add("Core Wall GWB");
3078
3079 //FACADE WALL WATERPROOFING
3080 matNo = 11; //3 mil Polyethylene
3081 tCost = tCost + CostIn[matNo] * AreaCLTFacWall;
3082 CostOut.Add(CostIn[matNo] * AreaCLTFacWall);
3083 names.Add("Facade Waterproofing");
3084
3085 //FACADE WALL INSULATION
3086 matNo = 12; //Extruded Polystyrene
3087 tCost = tCost + CostIn[matNo] * AreaCLTFacWall;
3088 CostOut.Add(CostIn[matNo] * AreaCLTFacWall);
3089 names.Add("Facade Insulation");
3090
3091 //FACADE WALL CLADDING
3092 matNo = 13; //Pine T&G Siding
3093 tCost = tCost + CostIn[matNo] * AreaCLTFacWall;
3094 CostOut.Add(CostIn[matNo] * AreaCLTFacWall);
3095 names.Add("Facade Cladding");
3096
3097 //CURTAIN WALL
3098 matNo = 14; //Double Glazed Windows & aluminum mullions
3099 tCost = tCost + CostIn[matNo] * AreaGlass;
3100 CostOut.Add(CostIn[matNo] * AreaGlass);
3101 names.Add("Curtain Wall");
3102
3103 //TIMBER STEEL CONNECTIONS
3104 //1% of timber cost
3105 tCost = tCost + 0.01 * (CostOut[0] + CostOut[1] + CostOut[2] +
3106 CostOut[5] + CostOut[9]);
3107 CostOut.Add(0.01 * (CostOut[0] + CostOut[1] + CostOut[2] +
3108 CostOut[5] + CostOut[9]));
3109 names.Add("Steel Connections");
3110
3111 //Itemized Output
3112 DataTree<System.Object> CostItems = new DataTree<System.Object>();
3113 DataTree<System.Object> CostItemsSF = new DataTree<System.Object>();
3114 GH_Path A = new GH_Path(0);
3115 GH_Path B = new GH_Path(1);
3116
3117 for (int i = 0; i < names.Count; i++)
3118 {

```

```

3117 CostItems.Add(names[i], A);
3118 CostItemsSF.Add(names[i], A);
3119 CostItems.Add(CostOut[i], B);
3120 CostItemsSF.Add(CostOut[i] / (AreaCLTFloor + AreaCLTRoof), B);
3121 }
3122
3123 //Output per SF
3124 double CostSFOut = tCost / (AreaCLTFloor + AreaCLTRoof);
3125
3126 //OUTPUTS
3127 TotCost = tCost;
3128 CostperSF = CostSFOut;
3129 OutputTot = CostItems;
3130 OutputSF = CostItemsSF;
3131 }
3132 }

```

DESIGN EXAMPLES

D

The following sections contain design examples showing the structural design of slabs, beams, column, and shear walls in the buildings.

SLAB DESIGN:

INPUTS: TOTAL SUPERIMPOSED LOAD: 146 PSF (FACTORED, $(1.2D+1.6L)$)
 SUPERIMPOSED DEAD LOAD: 35 PSF (UNFACTORED)
 LIVE LOAD: 65 PSF (UNFACTORED)
 SPAN LENGTH: 15'
 CONCRETE THICKNESS: 2"

$\lambda = 0.8$ NDS TIME EFFECT FACTOR, SPECIFIED BY LOAD CASE
 CHECK 5/8" THICK PANEL, MAXIMUM LENGTH 64'
 * ALGORITHM STARTS WITH THINNEST PANEL, CHECKS AND INCREMENTS TO NEXT THICKEST IF IT FAILS.

FROM MANUFACTURER:

Mallowable = 5,800 lb-ft/ft $f_{b,0} = 1950$ psi
 Vallowable = 1,470 lb/ft $f_{v,0} = 135$ psi

CHARRING DEPTH

$$d_{char} = 1.2 * (1.5 * (1)^{0.813}) = 1.8 \text{ inches}$$

DEMANDS:

$$M_u = 0.125 w L^2 = 0.125 \cdot 146 \cdot 15^2 = 4106 \text{ lb-ft/ft (3-SPAN CONTINUOUS)}$$

$$V_u = 0.6 w L = 0.6 \cdot 146 \cdot 15 = 1314 \text{ lb/ft}$$

CAPACITY:

EXCLUDING CHARRING, NON-COMPOSITE:

$$\phi M_n = \phi K_F \lambda C_m C_t C_L f_b \cdot b \cdot d^2 / l_o$$

$$= 0.95 \cdot 2.54 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 \cdot 1950 \cdot 12 \cdot 5.125^2 / 6 = 14,744 \text{ lb-ft/ft}$$

$$\phi V_n = \phi K_F \lambda C_m C_t \cdot \frac{2}{3} b d \cdot f_v$$

$$= 0.75 \cdot 2.88 \cdot 0.8 \cdot 1 \cdot 1 \cdot \frac{2}{3} \cdot 12 \cdot 5.125 \cdot 135 = 9504 \text{ lb/ft}$$

INCLUDING CHARRING, NON-COMPOSITE:

$$\phi M_n = \phi K_F \lambda C_m C_t C_L f_b \cdot b \cdot d^2 / l_o$$

$$= 0.95 \cdot 2.54 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 \cdot 1950 \cdot 12 \cdot (5.125 - 1.8)^2 / 6$$

$$= 6206 \text{ lb-ft/ft}$$

$$\phi V_n = \phi K_F \lambda C_m C_t \cdot \frac{2}{3} b \cdot d \cdot f_v$$

$$= 0.75 \cdot 2.88 \cdot 0.8 \cdot 1 \cdot 1 \cdot \frac{2}{3} \cdot 12 \cdot (5.125 - 1.8) \cdot 135$$

$$= 6205 \text{ lb/ft}$$

EXCLUDING CHARRING, COMPOSITE:

CALCULATE POSITION OF NEUTRAL AXIS:

$$y_{NA} = \frac{t_{conc} \cdot w \cdot E_{conc} \cdot y_{conc} + t_{clt} \cdot w \cdot E_{clt} \cdot y_{clt}}{t_{conc} \cdot w \cdot E_{conc} + t_{clt} \cdot w \cdot E_{clt}}$$

$$E_{conc} = 57000 \sqrt{f'_c} = 3,122,018 \text{ psi}$$

$$f'_c = 3000 \text{ psi}$$

$$y_{NA} = \frac{2 \cdot 12 \cdot 3,122,018 \cdot 6.125 + 5.125 \cdot 12 \cdot 1,400,000 \cdot 2.5625}{2 \cdot 12 \cdot 3,122,018 + 5.125 \cdot 12 \cdot 1,400,000}$$

$$y_{NA} = 4.22 \text{ in}$$

$$E_{clt} = 1,400,000 \text{ psi}$$

CALCULATE TRANSFORMED SECTION:

$$n = E_{conc} / E_{clt} = 3,122,018 / 1,400,000 = 2.23$$

$$I_{tr} = \left(\frac{1}{12} b h^3 \right)_{conc} + (A d^2)_{conc} + \left(\frac{1}{12} b h^3 \right)_{clt} + (A d^2)_{clt}$$

$$= \frac{1}{12} \cdot 12 \cdot 2.23 \cdot 2^3 + 12 \cdot 2.23 \cdot 2 \cdot (6.125 - 4.22)^2$$

$$+ \frac{1}{12} \cdot 12 \cdot 5.125^3 + 12 \cdot 5.125 \cdot (2.5625 - 4.22)^2$$

$$= 516 \text{ in}^4$$

DEMANDS:

$$f_{b,u} = M_u \cdot y / I$$

FOR CLT:

$$f_{b,u} = 4106 \text{ lb-ft/ft} \cdot 4.22 \text{ in} / 516 \text{ in}^4 = 403 \text{ psi}$$

FOR CONCRETE

$$f_{b,u} = 4106 \cdot (7.125 - 4.22) / 516 = 23 \text{ psi}$$

CAPACITY

$$\phi f_{b,n} = \phi K_F \lambda \cdot C_m \cdot C_t \cdot C_i \cdot f_{b,o} \quad (\text{CLT})$$

$$= 0.95 \cdot 2.54 \cdot 0.8 \cdot 1.1 \cdot 1.1 \cdot 1950 = 3368 \text{ psi}$$

$$\phi V_n = \phi K_F \lambda C_m C_t \frac{2}{3} b d f_v + \phi 2 \lambda \sqrt{f'_c} b t_{\text{conc}}$$

$$= 0.75 \cdot 2.88 \cdot 0.8 \cdot 1.1 \cdot \frac{2}{3} \cdot 12 \cdot 5.125 \cdot 135$$

$$+ 0.75 \cdot 2 \cdot 1 \cdot \sqrt{3000} \cdot 12 \cdot 2$$

$$= 11536 \text{ lb/ft}$$

$$\phi f_c = 0.9 \cdot 3000 = 2700 \text{ psi}$$

INCLUDING CHARRING, COMPOSITE:

NEUTRAL AXIS:

$$y_{NA} = \frac{2 \cdot 12 \cdot 3122018 \cdot (1 + 5.125 - 1.8) + (5.125 - 1.8) \cdot 12 \cdot 1400000 \cdot 1.6625}{2 \cdot 12 \cdot 3122018 + (5.125 - 1.8) \cdot 12 \cdot 1400000}$$

$$y_{NA} = 3.19 \text{ in}$$

TRANSFORMED SECTION

$$I_{tr} = \frac{1}{12} \cdot 12 \cdot 2 \cdot 23 \cdot 2^3 + 12 \cdot 2 \cdot 23 \cdot 2 \cdot (4.325 - 3.19)^2$$

$$+ \frac{1}{12} \cdot 12 \cdot 3 \cdot 325^3 + 12 \cdot 3 \cdot 325 \cdot (3.6625 - 3.19)^2$$

$$= 216.6 \text{ in}^4$$

DEMANDS:

FOR CLT:

$$f_{b,u} = 4106 \cdot 3.19 / 216.6 = 726 \text{ psi (CLT)}$$

FOR CONCRETE:

$$f_{b,u} = 4106 \cdot (5.325 - 3.19) / 216.6 = 40.5 \text{ psi (CONCRETE)}$$

CAPACITY:

$$\phi f_{b,n} = 3368 \text{ psi (CLT)}$$

$$\phi f'_c = 2700 \text{ psi (CONCRETE)}$$

$$\phi V_n = \phi K_F \lambda C_m C_t \frac{2}{3} b d f_v + \phi 2 \lambda \sqrt{f'_c} b t_{\text{conc}}$$

$$= 0.75 \cdot 2.88 \cdot 0.8 \cdot 1.1 \cdot \frac{2}{3} \cdot 12 \cdot 3.325 \cdot 135$$

$$+ 0.75 \cdot 2 \cdot 1 \cdot \sqrt{3000} \cdot 12 \cdot 2$$

$$= 8177 \text{ lb/ft}$$

DEMAND CAPACITY RATIOS.

	NON-COMPOSITE NO CHARRING	NON-COMPOSITE CHARRING	COMPOSITE NO CHARRING	COMPOSITE CHARRING
BENDING	0.28	0.66	0.12	0.22
SHEAR	0.14	0.21	0.11	0.16

DEFLECTIONS AND VIBRATIONS.

NON-COMPOSITE

$$\Delta_{D+L} = 0.0069 (w) L^4 / EI$$

$$= 0.0069 (35 + 13.7 + 65) \frac{\text{lb}}{\text{ft}} (15)^4 / 184000000 \text{ lb-in}^2 \cdot \frac{144 \text{ in}^2}{\text{ft}^2}$$

$$= 0.03 \text{ ft}$$

$$\Delta_L = 0.0069 (65) \cdot (15)^4 \cdot 144 / 184,000,000 = 0.02 \text{ ft}$$

$$EI = 184,000,000 \text{ lb-in}^2/\text{ft} \quad (\text{FROM MANUFACTURER})$$

$$EI_{APP} = \frac{1}{\left(\frac{1}{EI}\right) + \left(\frac{11.52}{GA \cdot L^2}\right)}$$

$$= \frac{1}{\left(\frac{1}{184,000,000}\right) + \left(\frac{11.52}{690,000(15.12)^2}\right)} = 94,448,669$$

$GA = 690,000 \text{ lb/ft}$ (FROM MANUFACTURER)

$$\text{VIBRATION SPAN} = EI_{APP}^{0.293} / 12.05 \cdot (\rho A)^{0.122}$$

$$= 94,448,669^{0.293} / 12.05 \cdot (0.5 \cdot 5.125 \cdot 12)^{0.122} = 11.9 \text{ ft}$$

$\rho = 0.5$ (DOUGLAS FIR SPECIFIC GRAVITY)

COMPOSITE SLABS:

$$\Delta_{D+L} = 0.0069 \cdot (35 + 13.7 + 65) \cdot (15)^4 \cdot 144 / 1400000 \cdot 516$$

$$= 0.008 \text{ ft}$$

$$\Delta_L = 0.0069 (65) (15)^4 \cdot 144 / 1400000 \cdot 516$$

$$= 0.005 \text{ ft}$$

$$EI_{APP} = \frac{1}{\left(\frac{1}{1400000 \cdot 516}\right) + \left(\frac{11.52}{690000(15.12)^2}\right)} = 526,434,224$$

$$\rho = \frac{\rho_{conc} t_{conc} \cdot 144 \text{ in}^2 + \rho_{alt} t_{alt} \cdot 144 \text{ in}^2}{(t_{conc} + t_{alt}) \cdot 144 \text{ in}^2} \cdot \frac{1}{\rho_{H2O}}$$

$$= \frac{150 \text{ lb/ft}^3 \cdot 2 \text{ in} \cdot 144 \text{ in}^2 + 33 \text{ lb/ft}^3 \cdot 5.125 \text{ in} \cdot 144 \text{ in}^2}{7.125 \text{ in} \cdot 144 \text{ in}^2} \cdot \frac{1}{62.43 \text{ lb/ft}^3}$$

$$= 1.05$$

$$\text{VIBRATION SPAN} = \frac{526,434,224^{0.293}}{(1.05 \cdot 12 \cdot 7.125)^{0.122} \cdot 12.05} = 17.2 \text{ ft}$$

DEFLECTION AND VIBRATION DEMAND/CAPACITY RATIOS.

	NON-COMPOSITE	COMPOSITE
Δ_{D+L}	0.48	0.13
Δ_L	0.48	0.12
VIBRATION	1.26	0.87

BEAM DESIGN

INPUTS:

$$V_u = 19.9 \text{ K}$$

$$M_u = -74.5 \text{ K-ft}$$

$$\text{Length} = 15 \text{ ft}$$

CHECK 5.5x10.5 BEAM

*ALGORITHM ITERATES INCREASING BEAM AREA UNTIL SECTION PASSES.

NON-COMPOSITE:

CAPACITY:

$$\phi V_n = \phi K_F \lambda \cdot C_m \cdot C_t \cdot C_{VR} \cdot \frac{2}{3} b \cdot d \cdot f_{v,o}$$

$$= 0.75 \cdot 2.88 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 \cdot \frac{2}{3} \cdot 5.5 \cdot 10.5 \cdot 265$$

$$= 27,704 \text{ lb}$$

$$\phi M_n = \phi K_F \lambda \cdot C_m \cdot C_t \cdot C_i \cdot C_v \cdot C_{Fu} \cdot C_i \cdot b d^2 / 6 \cdot f_{c,o}$$

$$= 0.85 \cdot 2.54 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 5.5 \cdot 10.5^2 / 6 \cdot 2400$$

$$= 86,208 \text{ lb-ft}$$

DEMAND CAPACITY RATIOS:

$$\text{SHEAR } 19.9 / 27.7 = 0.72$$

$$\text{BENDING } 74.5 / 86.2 = 0.86$$

FIRE RATING (ONLY IF CHARRING IS INCLUDED)

$$K_e \cdot L / d = 1 \cdot (15 \cdot 12) / 10.5 = 10.9$$

$$Z = \frac{2}{3} (\rho/c)^2 - 1.16 (\rho/c) + 2.1333$$

$$= \frac{2}{3} (0.86)^2 - 1.16 (0.86) + 2.1333 = 1.25$$

$$t = 2.54 \cdot Z \cdot w (4 - w/d)$$

$$= 2.54 \cdot 1.25 \cdot 5.5 (4 - 5.5/10.5)$$

$$= 64 \text{ min} > 60 \text{ min.}?$$

DEFLECTIONS

$$\Delta_{\text{FACTORED}} = \frac{M_u \cdot 12 \cdot 40 \cdot L^2}{384 E I}$$

$$= \frac{(74.5 \cdot 12) \text{ K-in} \cdot 40 \cdot 15^2 \text{ ft}^2 \cdot 144 \text{ in}^2/\text{ft}^2}{384 \cdot 1800 \text{ ksi} \cdot 1/12 \cdot 5.5 \cdot 10.5^3 \text{ in}^4}$$

$$= 0.81 \text{ in}$$

$$\Delta_{D+L} = 0.703 \cdot 0.81 = 0.57 \text{ in} \quad L / 240 = 0.75 \text{ in.}?$$

$$\frac{\text{UNFACTORED } D+L}{\text{FACTORED LOADS}} = 0.703$$

$$\Delta_L = 0.39 \cdot 0.81 = 0.32 \text{ in} \quad L / 360 = 0.5 \text{ in.}?$$

$$\frac{\text{UNFACTORED } L}{\text{FACTORED LOADS}} = 0.39$$

COMPOSITE BEAM-CLT SLAB.

EFFECTIVE FLANGE WIDTH:

$$b_{\text{eff}} = \min \begin{cases} 0.25 \cdot L_{\text{beam}} = 0.25 \cdot 15 \text{ ft} = 45 \text{ in} \\ 16 \cdot t_{\text{slab}} + w_{\text{beam}} = 16 \cdot 5.125 \text{ in} + 5.5 \text{ in} = 87.5 \text{ in} \\ \text{slab span} = 15 \text{ ft} = 180 \text{ in} \end{cases}$$

$$b_{\text{eff}} = 45 \text{ in}$$

CALCULATE LOCATION OF NEUTRAL AXIS:

$$y_{NA} = \frac{t_{CLT} \cdot b_{eff} \cdot E_{CLT} \cdot y_{CLT} + d_{beam} \cdot W_{beam} \cdot E_{beam} \cdot y_{beam}}{t_{CLT} \cdot b_{eff} \cdot E_{CLT} + d_{beam} \cdot W_{beam} \cdot E_{beam}}$$

$$= \frac{5.125 \cdot 45 \cdot 1400 \cdot 19.0625 + 16.5 \cdot 5.5 \cdot 1800 \cdot 8.25}{5.125 \cdot 45 \cdot 1400 + 16.5 \cdot 5.5 \cdot 1800}$$

$$= 15.4 \text{ in}$$

CALCULATE TRANSFORMED SECTION

$$n = E_{CLT} / E_{GLB} = 1400 / 1800 = 0.78$$

$$I_{tr} = \left(\frac{1}{12} b h^3 \right)_{CLT} + (A d^2)_{CLT} + \left(\frac{1}{12} b h^3 \right)_{GLB} + (A d^2)_{GLB}$$

$$= \left(\frac{1}{12} \cdot 45 \cdot 5.125^3 \right) + \left(45 \cdot 5.125 \cdot (19.0625 - 15.4)^2 \right)$$

$$+ \left(\frac{1}{12} \cdot 5.5 \cdot 16.5^3 \right) + \left(5.5 \cdot 16.5 \cdot (8.25 - 15.4)^2 \right)$$

$$= 10296 \text{ in}^4$$

CAPACITY:

$$\phi V_n = \phi K_F \lambda C_m C_t C_{vr} \cdot \frac{2}{3} b d f_r$$

$$= 27,704 \text{ lb}$$

$$\phi M_n = \phi K_F \lambda C_m C_t C_c C_v C_{fu} C_i f_{b,p} \cdot \frac{I_{tr}}{y_{NA}}$$

$$= 0.85 \cdot 2.54 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 2400 \cdot \frac{10296}{15.4}$$

$$= 230951 \text{ lb-ft}$$

DEMAND-CAPACITY RATIOS

$$\text{SHEAR} \quad 19.9 / 27.7 = 0.72$$

$$\text{BENDING} \quad 74.5 / 231 = 0.32$$

FIRE RATING (ONLY IF CHARRING IS INCLUDED)

$$K_e L / d = 1.15 \cdot 12 / 16.5 = 10.9$$

$$Z = \frac{2 \cdot 0.72^2 - 1.4 \cdot 0.72^2 + 2.1333}{3}$$

$$= 1.165$$

$$t = 2.54 \cdot 1.165 \cdot 5.5 \cdot (4 - 5.5 / 16.5)$$

$$= 84.5 \text{ min}$$

DEFLECTIONS

$$\Delta_{\text{factored}} = \frac{(74.5 \cdot 12) \cdot 40 \cdot 15^2 \cdot 144}{384 \cdot 1400 \cdot 10296}$$

$$= 0.21 \text{ in}$$

$$\Delta_{D+L} = 0.15 \text{ in} < \frac{L}{240} = 0.75 \text{ in?}$$

$$\Delta_L = 0.08 \text{ in} < \frac{L}{360} = 0.5 \text{ in?}$$

COLUMN DESIGN

INPUTS: $P_u = 169.2 \text{ K}$

$V_u = M_u \approx 0$

LENGTH = 12 ft

CHECK: 8.75x12 COLUMN

*ALGORITHM ITERATES INCREASING COLUMN AREA UNTIL SECTION PASSES.

ADJUSTMENT FACTORS:

$$F_{ce} = 0.822 \cdot E_{min} \cdot \frac{1}{\left(\frac{L}{d}\right)^2}$$

$$= \frac{0.822 \cdot 850,000}{\left(\frac{12 \cdot 12}{12}\right)^2} = 4852 \text{ PSI}$$

$$F_c^* = \phi K_E \lambda \cdot C_m \cdot C_t \cdot F_c$$

$$= 0.9 \cdot 2.4 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1690$$

$$= 2851 \text{ PSI}$$

$$C_p = \frac{(1 + F_{ce}/F_c^*)}{2C} \cdot \sqrt{\left(\frac{1 + F_{ce}/F_c^*}{2C}\right)^2 - \frac{F_{ce}/F_c^*}{C}}$$

$$= \frac{(1 + 4852/2851)}{2 \cdot 0.9} \cdot \sqrt{\left(\frac{1 + 4852/2851}{2 \cdot 0.9}\right)^2 - \frac{4852/2851}{0.9}}$$

$$= 0.9$$

CAPACITY

$$\phi P_n = \phi K_E \lambda \cdot C_m \cdot C_t \cdot C_p \cdot f_c \cdot A$$

$$= 0.9 \cdot 2.4 \cdot 0.8 \cdot 1 \cdot 1 \cdot 0.9 \cdot 1650 \cdot 8.75 \cdot 12$$

$$= 269438 \text{ lb}$$

DEMAND CAPACITY RATIO:

AXIAL $169.2/269.4 = 0.63$

FIRE RATING (ONLY IF CHARRING IS INCLUDED)

$$K_e \cdot L/d = 1 \cdot 12 \cdot 12/12 = 12$$

$$Z = \frac{2 \cdot 0.63^2 - 1.6 \cdot 0.63 + 1.933}{3}$$

$$= 1.19$$

$$t = 2.54 \cdot 1.19 \cdot 8.75 \cdot (3 - 8.75/12)$$

$$= 60.1 \text{ min}$$

$V_u \approx M_u \approx 0$ NO COMBINED BENDING/AXIAL LOADING

SHEAR WALL DESIGN

INPUTS: BASE SHEAR = 2148 K (SEE SPREADSHEET APPENDIX B)

USE 12.18 in THICK WALL, 20' LONG

$$V_r = 11624 \text{ lb/ft (FROM MANUFACTURER)}$$

BUILDING CENTER OF MASS (INPUT)

$$I = \frac{1}{12} \cdot b \cdot h^3 = \frac{1}{12} \cdot 12.18 \cdot (20 \cdot 12)^3 = 14031360 \text{ in}^4$$

$$K_{\text{wall}} = \frac{1}{\left(\frac{L^3}{3EI} + \frac{L}{KGA} \right)}$$
$$= \frac{1}{\left(\frac{(20 \cdot 12)^3}{3 \cdot 1800 \cdot 14031360} + \frac{(20 \cdot 12)}{0.833 \cdot 50,000 \cdot 12.18 \cdot 20 \cdot 12} \right)}$$
$$= 5422.4 \text{ lb/in}$$

BUILDING CENTER OF RIGIDITY

$$COR = \frac{\sum_{i=1}^{n_{\text{walls}}} K_{\text{wall},i} \cdot X_{\text{wall offset from cm},i}}{\sum_{i=1}^{n_{\text{walls}}} K_{\text{wall},i}}$$

ACCIDENTAL TORSION

$$T = \max \left\{ \begin{array}{l} 0.05 \\ COR/BUDG DIM \end{array} \right\} = 0.05 \%$$

FORCES IN WALL (RIGID DIAPHRAGM ASSUMED)

$$V_{\text{wall}} = V_{\text{base shear}} \cdot K_{\text{wall}} \cdot (1 + 2 \cdot T) / \sum_{i=1}^{n_{\text{wall}}} K_{\text{wall},i}$$
$$= 2148 \cdot 5422.4 (1 + 2 \cdot 0.05) / 21689.6$$
$$= 590.7 \text{ K}$$

$$M_{\text{wall}} = V_{\text{wall}} \cdot h = 590.7 \cdot 40 = 23,628 \text{ K-ft}$$

$$h = \frac{\sum_{i=1}^{n_{\text{floors}}} F_{\text{story},i} \cdot h_i}{\sum F_{\text{story},i}} = 40 \text{ ft}$$

$$S_{\text{wall}} = \frac{t \cdot L^2}{6} = \frac{12.18 (20 \cdot 12)^2}{6} = 116928 \text{ in}^3$$

DEMAND/CAPACITY

$$V_u \leq \phi V_n$$

$$590.7 \text{ K} \leq \phi K_F \lambda C_m C_t \cdot V_r \cdot L_{\text{wall}}$$

$$590.7 \text{ K} \leq 0.75 \cdot 2.88 \cdot 1 \cdot 1 \cdot 1 \cdot 11,624 \cdot 20$$

$$590.7 \text{ K} \leq 502.2 \text{ K} \rightarrow \text{MORE WALL NEEDED.}$$

$$f_{b,u} \leq \phi f_{b,n}$$

$$\frac{M}{S} \leq \phi K_F \lambda \cdot C_m C_t f_c$$

$$\frac{23628 \cdot 12}{116928} \leq 0.9 \cdot 2.4 \cdot 1 \cdot 1 \cdot 1 \cdot 1150$$

$$2424 \text{ psi} \leq 2484 \text{ psi}$$

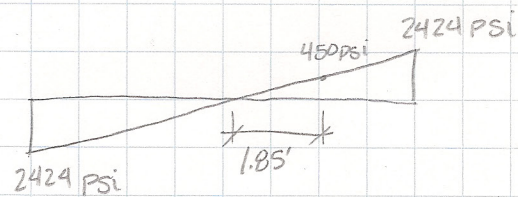
ADD STEEL REINFORCING AT END OF WALL FOR TENSION

$$1" \text{ } \phi \text{ ROD } \quad A = \pi/4 \cdot 1^2 = 0.79 \text{ in}^2$$

$$\text{Strength} = 0.9 \cdot 0.79 \text{ in}^2 \cdot 60 \text{ ksi} = 42 \text{ K}$$

$$f_{t,clt} = 450 \text{ psi}$$

$$T = \text{tensile force} \cdot L_{\text{tension}} \cdot t_{\text{wall}}$$



$$T = (2424 - 450) \cdot \frac{1}{2} \cdot (10 - 1.85) \cdot 12 \cdot 12.18$$

$$= 1175.7 \text{ K}$$

(28) 1" RODS AT 3.5" O.C. ALONG BOTH ENDS OF WALL.

CHECK DRIFTS.

$$\text{STORY DISPLACEMENT} = \frac{F_{\text{story}} \cdot (1 + 2T) \cdot C_d}{k_{\text{story}} \cdot I_e}$$

$$= \frac{200 \cdot (1.1) \cdot 2}{21689.6 \cdot 1}$$

$$= 20.3 \text{ in}$$

$$\% \text{ DRIFT} = \frac{20.3}{12.12} = 14\% \leq 2\% \quad (\text{SINGLE STORY})$$

BUILDING DRIFT

$$\% \text{ DRIFT} = \frac{\sum_{i=1}^{n_{\text{stories}}} \Delta_{\text{story},i}}{n_{\text{story}}} \leq 2\%$$

Wall-Foundation Connection Design (per Linear Foot)		
CLT Panel		
G	0.5	NDS T12.3.3A Douglas Fir-Larch
E	1400000	psi
Panel Thickness	12.18	in
Dowel Type Fasteners		
Fastener Diameter	0.5	in
Hole Diameter	0.5625	in
Fastener Length	10	
Fastener Spacing (Along Wall)	3	in
s_{min}	1.5	in
Number of Rows	11	
Row Spacing	3	in
s_{min}	1.5	in
Edge Distance	3	in
# of Screws per Foot	44	
Steel Plate		
$f_{y,pl}$	36000	psi
$f_{u,pl}$	50000	psi
E	29000000	psi
Plate Thickness	0.5	in
Plate Edge Distance	2	in
Plate Height	35	in
Concrete Reinforcing		
Check Timber		
C_M	1	
C_t	1	
R_{EA}	0.85	
γ	95459	
u_{vert}	1	
u_{horiz}	1	
m_{vert}	0.97	
m_{horiz}	0.95	
n_{vert}	11.00	
n_{horiz}	4.00	
$C_{g,\text{vert}}$	0.99	
$C_{g,\text{horiz}}$	1.00	
C_{Δ}	0.86	
C_{eg}	1	
C_{di}	1	
C_{tn}	1	
K_F	3.32	
ϕ	0.65	
λ	1	

Double Shear T12.3.1A		
l_m	0.5	in
l_s	5.8	in
F_{em}	36000	psi
F_{es}	3150	psi
F_{yb}	45000	psi
R_e	11.43	
k_3	0.50	
K_θ	1.25	
Z_{lm}	*Bolt Hole Bearing	lb
Z_{ls}	3679	lb
Z_{lls}	7865	lb
Z_{lv}	4661	lb
Z	3679	lb
Z'	6737	lb/fastener
Connection Capacity/ft	296.4	Kips/ft
Check Steel Plate		
Bolt Hole Bearing	16.2	K/bolt
Plate Tension	194.4	K/ft
Plate Shear	110	K/ft