Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Elise Munch-Ellingsen
Trine Tveten Berge

IFC-based Element Mapping for Sustainable Construction: Optimising Reuse and Structural Integrity Assessment

A Comprehensive Methodology for Efficient Element Retrieval and Incorporation in New Constructions

Master's thesis in Civil and Environmental Engineering
Supervisor: Sverre Magnus Haakonsen
June 2023

NTNU
Norwegian University of
Science and Technology

Elise Munch-Ellingsen
Trine Tveten Berge

IFC-based Element Mapping for Sustainable Construction: Optimising Reuse and Structural Integrity Assessment

A Comprehensive Methodology for Efficient Element Retrieval and Incorporation in New Constructions

Master's thesis in Civil and Environmental Engineering
Supervisor: Sverre Magnus Haakonsen
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

NTNU
Norwegian University of
Science and Technology

Department of Structural Engineering
Faculty of Engineering
NTNU- Norwegian University of Science and Technology

ACCESSIBILITY

Open

MASTER THESIS 2023

SUBJECT AREA:	DATE:	NO. OF PAGES:
Conceptual Structural Design	10.06.2023	66 + 8 (Appendices)

TITLE:

IFC-Based Element Mapping for Sustainable Construction: Optimising Reuse and Structural Integrity Assessment

IFC-Basert Elementkartlegging for Bærekraftig Konstruksjonspraksis: Optimalisering av Gjenbruk og Strukturell Integritetsvurdering

BY:
Elise Munch-Ellingsen
Trine Tveten Berge

SUMMARY:
The construction industry significantly contributes to global greenhouse gas emissions and environmental degradation through its current practices. The predominant approach of producing, using, and disposing of construction materials leads to high operational energy consumption and CO_2 emissions. The extraction and processing of raw materials further contribute to these emissions. With the projected growth in construction output, there is an urgent need to find solutions that reduce the production of new materials and minimise greenhouse gas emissions.

Reusing components in new constructions reduces the demand for new materials, conserves resources, and minimises the environmental impact associated with material extraction and production. However, the reuse of structural elements in construction projects faces challenges due to the lack of comprehensive inventories of available reusable materials and the inconvenient process of acquiring associated data. To address these challenges, this master's thesis proposes a method that utilises IFC files from existing buildings to extract information for a material database. These materials are then placed in a database and evaluated based on predefined requirements to identify suitable replacements for integration into new structures. This is achieved by the use of Grasshopper. Additionally, a thorough structural analysis is performed directly from the IFC model to ensure the structural integrity of constructions using reused materials. This method integrates the structural analysis seamlessly into the same software used for matching elements through the use of Grasshopper. This facilitates efficient decision-making throughout the design process and encourages a more sustainable construction practice. Reusing components in new constructions reduces the demand for new materials, conserves resources, and minimises the environmental impact associated with material extraction and production.

Two comprehensive case studies have been conducted to assess and evaluate the functionality, strengths, and weaknesses of the proposed method. The findings from the case studies confirmed that the method presented in this master's thesis has significant potential for the efficient reuse of structural elements. A successful material data extraction was performed based on IFC files, laying the foundation for a comprehensive database of reusable elements. An effective mapping process between elements was conducted with the objective to create a predefined structure with the use of reused materials while reducing the cut waste. In Case Study 1 and Case Study 2, the predefined structure successfully fulfilled the demand for approximately 90% and 70% of elements, respectively, given a specific inventory of reusable materials. The case studies proved a reduction of two and seven times the GWP and an increase of cost of 13% and 33%. Through a structural verification conducted using an implemented FEM analysis, it is evident that the utilisation of reused elements in the predefined structures is justified in terms of maintaining structural integrity.

RESPONSIBLE TEACHER: Sverre Magnus Haakonsen

SUPERVISOR(S): Sverre Magnus Haakonsen

CARRIED OUT AT: Department of Structural Engineering, NTNU, Trondheim

Sammendrag

Bygg- og anleggsindustrien bidrar betydelig til globale klimagassutslipp og miljøforringelse gjennom sine nåværende praksiser. Den dominerende bruk-og-kast tilnærmingen til byggematerialer fører til et høyt energiforbruk og høye utslipp. Med en ventet vekst i byggeproduksjon og derav økt press på materialutvinning- og produksjon, er det er stort behov for å finne mer bærekraftige løsninger som kan lette trykket på en stadig økende material etterspørsel.

Gjenbruk av byggematerialer i nye konstruksjoner kan bidra til å redusere presset på utvinning og produksjon av nye materialer. Likevel, blir gjenbruk svært litt gjennomført i praksis i byggebransjen. Dette skyldes blant annet mangelen på tilstrekkelig god nok kartlegging av eksisterende bygg. Denne masteroppgaven foreslår en metode som bruker IFC filer fra eksisterende bygninger for å hente ut informasjon om tilgjengelige gjenbrukbare byggematerialer. Informasjonen om materialene blir plassert i en database som videre brukes for å evaluere en implementering av elementet i et planlagt prosjekt. Videre blir en grundig strukturell analyse utført av konstruksjonen som blir konstruert av gjenbrukbare elementer for å sikre strukturell integritet. Metoden er basert på en kombinasjon av Python og Grasshopper kode, hvor sistenevne muliggjør visuell fremstilling av resultatene, samt strukturell analyse, på samme plattform. Delmetodene innlemmes i samme programvare, og øker dermed effektiviteten gjennom hele designprosessen. Dette kan bidra til å legge til rette for en mer bærekraftig byggepraksis.

To omfattende case-studier er gjennomført for å vurdere funksjonaliteten, styrkene og svakhetene til den foreslåtte metoden. Resultatene fra case-studiene bekrefter at den presenterte metoden har betydelig potensial for å fremme effektiv gjenbruk av bygningsmaterialer. En vellykket uthenting av strukturelle elementer ble utført basert på IFC-filer, og dannet grunnlaget for en omfattende database med gjenbrukbare elementer. En effektiv kartleggingsprosess av elementer ble gjennomført med mål om å finne strukturelt tilfredsstillende kvaliteter for gjenbruk i en forhåndsdefinert struktur. I Case Study 1 og Case Study 2 fikk den forhåndsdefinerte strukturen dekke behovet for omtrent 90% og 70% av elementene, henholdsvis, gitt en spesifikk database av tilgjengelige og gjenbrukbare elementer. Case-studiene viste en reduksjon på to og syv ganger GWP (Global Warming Potential) og kostnadsøkning på 13% og 33%. En strukturell verifisering ble gjennomført i en FEM analyse, som viste at bruken av de gjenbrukte materialene i den forhåndsdefinerte strukturen var berettiget.

Abstract

The construction industry significantly contributes to global greenhouse gas emissions and environmental degradation through its current practices. The predominant approach of producing, using, and disposing of construction materials leads to high operational energy consumption and CO_2 emissions. The extraction and processing of raw materials further contribute to these emissions. With the projected growth in construction output, there is an urgent need to find solutions that reduce the production of new materials and minimise greenhouse gas emissions.

Reusing components in new constructions reduces the demand for new materials, conserves resources, and minimises the environmental impact associated with material extraction and production. However, the reuse of structural elements in construction projects faces challenges due to the lack of comprehensive inventories of available reusable materials and the inconvenient process of acquiring associated data. To address these challenges, this master's thesis proposes a method that utilises IFC files from existing buildings to extract information for a material database. These materials are then placed in a database and evaluated based on predefined requirements to identify suitable replacements for integration into new structures. This is achieved by the use of Grasshopper. Additionally, a thorough structural analysis is performed directly from the IFC model to ensure the structural integrity of constructions using reused materials. This method integrates the structural analysis seamlessly into the same software used for matching elements through the use of Grasshopper. This facilitates efficient decision-making throughout the design process and encourages a more sustainable construction practice. Reusing components in new constructions reduces the demand for new materials, conserves resources, and minimises the environmental impact associated with material extraction and production.

Two comprehensive case studies have been conducted to assess and evaluate the functionality, strengths, and weaknesses of the proposed method. The findings from the case studies confirmed that the method presented in this master's thesis has significant potential for the efficient reuse of structural elements. A successful material data extraction was performed based on IFC files, laying the foundation for a comprehensive database of reusable elements. An effective mapping process between elements was conducted with the objective to create a predefined structure with the use of reused materials while reducing the cut waste. In Case Study 1 and Case Study 2, the predefined structure successfully fulfilled the demand for approximately 90% and 70% of elements, respectively, given a specific inventory of reusable materials. The case studies proved a reduction of two and seven times the GWP and an increase of cost by 13% and 33%. Through a structural verification conducted using an implemented FEM analysis, it is evident that the utilisation of reused elements in the predefined structures is justified in terms of maintaining structural integrity.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Sverre Magnus Haakonsen, for his incredible dedication and invaluable guidance throughout the work on this master's thesis. We have especially appreciated his availability and helpfulness during the final stages of this work. No question asked has been too trivial, and we truly value that. We are grateful for being given significant freedom in choosing and shaping the thesis, as well as the advice provided to help us achieve our desired goals. Additionally, we would like to thank Marcin Luczkowski for always keeping the door open for "two minutes", which consistently resulted in satisfactory solutions and a little more than two minutes.

Additionally, we would like to acknowledge and thank COWI for providing us with the necessary materials to conduct Case Study 2, as presented in Section 6.2, and for the Verification of FEM Analysis in Section 6.3. The IFC files of Tennebekk and Buebygget, along with the calculation report of Tennebekk, proved invaluable in the evaluation of our method. We would also like to express our thanks to Artur Tomczak for laying the groundwork in Grasshopper, which served as the foundation for the Element Mapper presented in Section 5.2. Lastly, we wish to extend our gratitude and recognition to the IfcOpenShell and Grasshopper community for answering our questions in various forums. Their assistance has been truly helpful throughout the development of our method.

Table of Contents

List of Figures

List of Tables

1 Introduction

The "2022 Global Status Report for Buildings and Construction" by the United Nations Environmental Program, highlights that approximately 37% of global operational energy and process-related CO_2 emissions can be attributed to the building and construction industry [38]. This significant environmental impact is partly driven by load-bearing systems, encompassing various stages such as material extraction, production, construction, and demolition [11]. Currently, structural elements in buildings are predominantly designed for single-use purposes, leading to a substantial amount of waste generation at the end of their life cycle. Only a small fraction of materials are recycled, thereby intensifying emissions and aggravating the strain on material demand. Given the projected growth in construction output in the years to come, it has become critical to promote the efficient utilisation of materials and promote a transition towards a circular economy.

It is evident that there is great potential within the construction industry to develop practical methods that can retrieve, test and implement reused materials. There is a need for a comprehensive tool that integrates all aspects of the design process, making it more time- and cost-effective. To optimise the processes involved in reuse projects, the existence of comprehensive material databases is fundamental. Such databases would ideally encompass all available structural elements that may be reused, providing valuable information regarding their location and properties. This would facilitate the process of finding suitable structural elements for incorporation into new structures.

The process of efficiently mapping available supply materials and determining their placement in new construction can be accomplished through the utilisation of digital tools. By employing Computer-Aided Design (CAD) software in combination with an Industry Foundation Class (IFC) file, the entire workflow of mapping supply materials and identifying suitable locations for their installation can be effectively optimised. As an IFC represents a digital twin of a building, the file format includes extensive information regarding the structural elements. Information within such a model can be extracted and lay the groundwork for a comprehensive database representing available supply materials. The utilisation of CAD software, such as Grasshopper, allows for further mapping and analysis of a reuse structure.

The aim of this master's thesis is to introduce a method that encourages the reuse of materials in new structures. The method involves evaluating these materials and identifying suitable matches for integration into new constructions based on their specific requirements. Subsequently, a comprehensive structural analysis is conducted to validate the structural integrity of the new construction. The primary objective of this thesis is to facilitate more efficient retrieval and incorporation of reusable materials into new buildings, driven by the motivation to mitigate environmental impact.

2 Background

2.1 Circular Construction: Reuse of Structural Elements

The term circular construction, deriving from the concept of circular economy, refers to the strategy of reducing the overall climate and environmental impacts by closing the building material loop [20]. This can be obtained by implementing more sustainable practices in the construction sector such as using recycled materials, components or whole building parts, rather than continuing the linear practice of "take-make-consume-dispose" [37]. To achieve circular construction it is essential to consider how to maximise the lifespan of materials and constructions at the very beginning of the design process. An example of such a practice is Design for Disassembly (DfD). The objective of the design concept is to incorporate deconstruction principles into the initial design of buildings, increasing the potential for recovering materials and structural components when the construction reaches its end of life (EoL) [22]. Nevertheless, due to the industry's frequent construction and demolition activities, it becomes imperative to explore the potential for maximising the reuse of building elements. Design for Reuse (DfR), involves integrating reclaimed components into the design of new structures. This approach may include activities such as dismantling, cleaning, testing, storing, and re-fabricating building components to prepare them for reuse [22].

Reuse and recycling are two important strategies in the construction industry for reducing waste, conserving resources, and promoting sustainability. While both approaches contribute to minimising environmental impact, there are distinct differences between reuse and recycling in terms of their processes and outcomes. Reuse, in the given context, is defined as the reuse of an element without transformation [2]. This involves extending the lifespan of building materials, components, or entire structures by giving them a new purpose or function. It focuses on preserving the existing value of materials and preventing them from becoming waste. Reuse can take various forms, such as salvaging and re-purposing materials from demolition sites, utilising reclaimed or refurbished building components, or adapting existing structures for new uses [10]. Recycling, on the other hand, is based on breaking down materials into their raw components and creating new products or materials. This can result in a gradual deterioration of their qualities, making them less predictable and potentially compromising their safety when utilised in structures [3]. In contrast to reuse, recycling generally entails a more resource-intensive process, making it a less sustainable option in many cases.

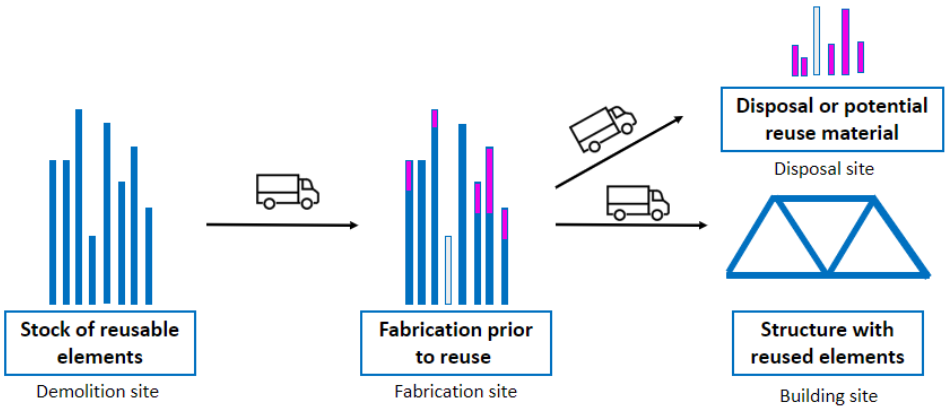


Figure 2.1: Simplified illustration of the reuse cycle of structural elements.

2.1.1 LCA

Life cycle assessment, LCA, is a methodology to quantify environmental impacts associated with all the life stages of a product. An LCA encompasses a thorough evaluation of the energy and materials needed throughout the entire value chain of a product, including its associated processes and services. By quantifying all relevant emissions to the environment, this assessment enables informed and strategic sustainable decision-making during the construction design process [32].

GWP, Global Warming Potential, is a widely used metric in the life cycle of a product, employed to compare the warming potential of different greenhouse gases (GHG) in terms of equivalent CO_2 emissions [36]. Hence, GWP functions as both an environmental impact indicator and a valuable tool for assessing the efficacy of emission reduction measures, as well as guiding policy decisions pertaining to greenhouse gas mitigation.

Two key life cycles that can be implemented in an LCA are the cradle-to-grave and cradle-to-cradle life cycles. A cradle-to-grave assessment consists of mainly five stages: Extraction of raw materials (cradle), transportation, manufacturing, use and disposal (grave) [16]. This correlates to a linear construction. A cradle-to-cradle assessment considers the impact of the life of a product by replacing the disposal stage with a recycling process, therefore representing a circular construction. The ultimate goal is no waste at the end of the life cycle. This results in a reduction in the footprint of the product. An illustration cradle-to-grace and cradle-to-cradle life cycle of a product are shown in Figure 2.2.

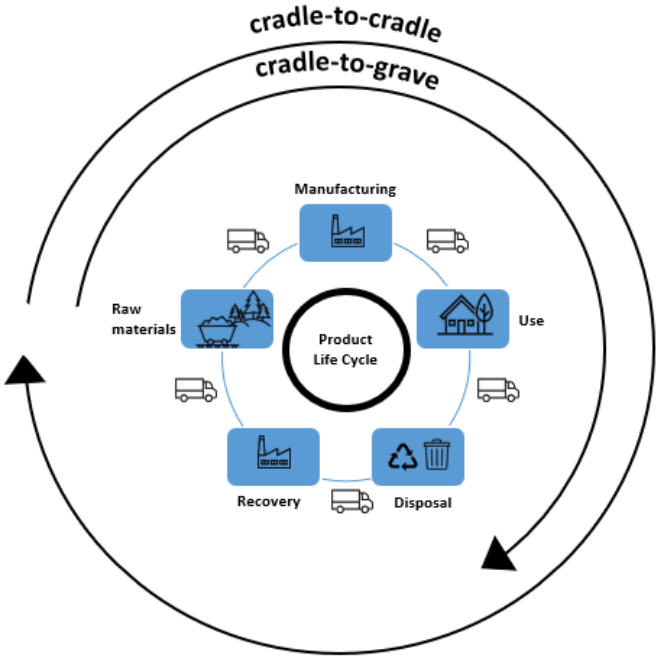


Figure 2.2: Life cycle of a product.

pre-defined

2.2 Reference Project: KA13

Kristian August Street 13 (KA13) is a pilot reuse building project located in the center of Oslo, Norway. The office building was bought by Entra in 2016, who initiated the project in 2018. The goal of the reuse project was to rehabilitate the existing property mass and use as many reused materials as possible for the

extension of the building. The building was completed in 2021 and has a reuse rate of close to 80%, and with this is called "Norway's most ambitions reuse construction project". The collected materials for reuse originated from over 25 different buildings. The data and details in this section are based on the experience rapport, "Erfaringsrapport ombruk: Kristian Augusts gate 13", provided by Entra ASA [15].

57 ton steel elements were collected for the project whereas 45 ton were actually used. The total steel used in the project was 64 ton, where 45 ton were reused elements. The collection of steel elements was sorted based on origin and properties and placed into test groups. Each element was tested with a UCI Hardness Tester to verify homogeneity in the batches. The chemical composition of the elements was also examined by the use of Optical Emission Spectrometry. One element from each test group underwent destructive tensile strength and impact resistance testing.

The cost of the reused steel in the project was approximately 100 NOK/kg. This cost includes the search for the materials, purchase, dismantling, scanning and testing, storage, transport and assembly. The price for new steel at the time of the project was set to 67 NOK/kg, making the reused steel 49 % more expensive. The cost associated with project management and structural consulting was not included. It is estimated that these processes took twice as long as if new elements of optional profiles and dimensions could be used.

It is estimated that the emission saving in the project was around 110 ton CO_2 equivalents, by using reused steel elements rather than new ones. This corresponds to a total emission saving of 97 %. The processes associated with the highest contribution of the emissions are associated with the cutting and sandblasting of elements.

The reuse project posed challenges related to the extraction of structural elements, referred to as a "treasure hunt", encompassing various aspects such as their origin, transportation, storage, testing, and documentation. The design phase was significantly delayed due to these challenges, as the process of gathering these materials proved to be more time-consuming. This, in turn, posed challenges in maintaining financial control, as the costs associated with procurement, logistics, documentation, processing, and assembly became unpredictable.

In comparison to the use of raw elements, the processes related to reused elements lead to a substantially greater investment of time. This also led to challenges in keeping control financially, as the price of procurement, logistics, documentation, processing and assembly was unpredictable.

Studying a project that has already been executed, such as the KA13 project, provides numerous advantages in the context of exploring the reuse of structural elements. Gaining insights from the construction industry, evaluating success, identifying areas for improvement, benchmarking opportunities, and validating research contribute to a more comprehensive understanding of the challenges, outcomes, and best practices associated with incorporating reused materials in construction projects.

The pilot project intensified the initiative to enhance the documentation of existing structural elements for the purpose of reuse in future projects. It is evident that this development will make the reuse of elements more efficient and cost-saving in the years to come. The versatility of building materials will undoubtedly represent a significant milestone in the journey towards a sustainable and circular economy.

3 Theory

3.1 Computer Aided Design (CAD)

Computer-Aided Design (CAD), refers to the utilisation of computer-based software to assist in various design processes. CAD software empowers designers and engineers to create precise and detailed 2D drawings and realistic 3D models of their envisioned products. With its intuitive interface and powerful tools, CAD software enables professionals to efficiently explore, modify, and optimise designs, fostering innovation and enhancing productivity in various industries.

3.2 Building Information Modeling (BIM)

BIM is a digital representation of the physical and functional attributes of a structure. It serves as a collaborative information repository that provides reliable data throughout the entire lifespan of a facility, starting from its initial conception until its eventual demolition.

3.3 Industry Foundation Classes (IFC)

Industry Foundation Classes (IFC) is an open international standard (ISO 16739-1:2018) managed by the nonprofit and neutral organisation buildingSMART [6]. The schema is a standardised, digital description of the built asset used in BIM for the exchange of digital building data between software programs. The IFC schema describes the physical components of a building, mechanical and electrical systems, and manufactured products. In addition, the schema can describe abstract structural analysis models, energy analysis models, cost breakdowns, work schedules and more [6]. With this versatility, IFC provides a comprehensive and efficient solution for managing building information and supporting collaborative workflows across the entire construction process.

3.3.1 IFC formats

The data from an IFC schema can be encoded in various formats, such as STEP, XML and JSON, each having trade-offs of software support, scalability, and readability. STEP Physical Format (IFC-SPF) has the extension ".ifc" and is the most widely used format for IFC. IFC-SPF is based on ISO 10303-21, a standard for clear text representation of EXPRESS language. The standard allows the product data described in EXPRESS to be transferred between computer systems [4].

3.3.2 IFC Specifications

Currently, the most commonly supported versions of Industry Foundation Classes are IFC2X3 (ISO standard since 2005) and IFC4 (ISO standard since 2013). The latest ISO improved standard is IFC4.0.2.1, and was approved in 2018. A full overview of the IFC specifications can be found on buildingSMART's website [5].

Further, in this thesis, the IFC Specification being referenced is the Industry Foundation Classes 4.0.2.1 Version 4.0-Addendum 2- Technical Corrigendum 1 [7].

3.3.3 IFC entity hierarchy

In the IFC schema, the building model information is organised into a collection of entities, a class of information defined by common attributes and constraints within the building and has the prefix "Ifc" in front of it. The entities in the IFC schema are organised into a hierarchy. This structural system defines a set of parent-child relationships, where each child entity inherits properties and attributes from its parent entity. The IFC hierarchy is based on the concept of generalisation and specialisation, where more specific entities are derived from more general ones.

The IFC hierarchy is designed to provide a structured and organized way of representing building information within an IFC schema. By defining relationships between entities in such a way, it becomes possible to create complex building models with a high degree of accuracy and detail. This also enables interoperability between different software platforms and applications, allowing building information to be shared and exchanged across different stakeholders in the construction and building management industries.

IfcBeam entity

An *IfcBeam* is defined as a horizontal, or nearly horizontal structural member able to withstand load primarily by resisting bending. It is not required for such a member to be load-bearing. An illustration of the *IfcBeam* entity inheritance is shown in Figure 3.1.

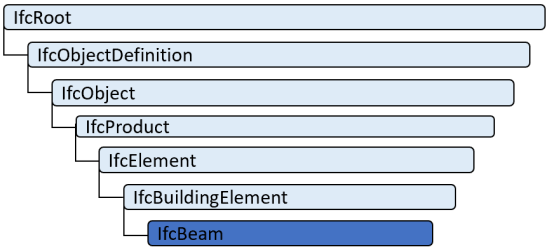


Figure 3.1: *IfcBeam* entity inheritance.

IfcColumn entity

An *IfcColumn* is defined as a vertical, or nearly vertical, structural member often aligned with a structural grid intersection The structural member transmits the weight of the structure above to other structural elements below through compression. It is not required for the IfcColumn to be load-bearing. An illustration of the *IfcColumn* entity inheritance is shown in Figure 3.2.

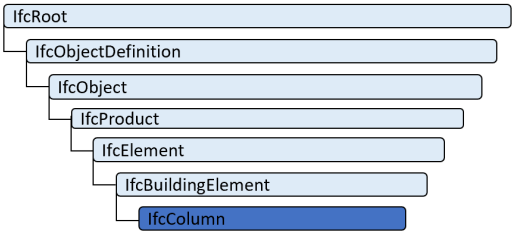


Figure 3.2: *IfcColumn* entity inheritance.

IfcSlab entity

An *IfcSlab* is defined as a structural component in a building that normally encloses a space vertically. The *IfcSlab* may serve as a floor, lower support, or as the ceiling, upper support, in the core of the construction. The flooring and roofing in the upper finish and the ceiling and suspended ceiling in the lower finish are considered to be coverings and not slabs. An illustration of the *IfcSlab* entity inheritance is shown in Figure 3.3.

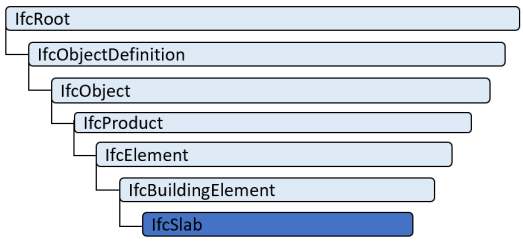


Figure 3.3: *IfcSlab* entity inheritance.

3.3.4 IFC attributes

Each IFC entity has a set of IFC attributes describing the entities characteristics and served as the entities main identifiers. The number of attributes for an entity is specified in the IFC Specification. Amongst others, they may be the GlobalId, which is required, the OwnerHistory, Name and Description. The attributes are consistent throughout the data exchange and cannot be edited by the end user.

The GlobalId, also referred to as a Globally Unique Identifier (GUID), ensures that all instances of an object in an IFC file can be uniquely identified. The data type is an auto-generated 128-bit number [8]. The GUID assigned to each object remains constant throughout the lifespan, ensuring consistency across the entire software world. The GUID attribute can therefore be used for various purposes such as tracking changes, identifying linked objects, and referencing objects.

3.3.5 IFC properties

The *IfcPropertySet* is a container including properties within a property tree. Properties that are related may be grouped into the same property set, Pset for short, and can be assigned to an IFC entity. If the property set is defined in the IFC Specification, it has the prefix "Pset_". Within the property set, there may be several properties that describe the entity further. An example of this is "Pset_BeamCommon" which is a common property set of all *IfcBeam* occurrences. Within this property set there is the property "LoadBearing" indicating if the object is intended to carry loads. This property name is consistent in all IFC files.

3.3.6 IFC quantities

IFC quantity is information that can be quantified, such as length, area, volume, weight or time. Similarly to the IFC properties, the quantities have a name and value and are grouped into quantity sets. If the quantity set is defined in the IFC Specification, it has the prefix "Qtos_", short for Quantity Take-Off. An example of this is the quantity set "Qto_BeamBaseQuantities". This provides the base quantities common for all

the *IfcBeam* occurrences. Quantities within this set are length, cross section area, outer surface area, gross surface area, net surface area, gross volume, net volume, gross weight and net weight.

3.3.7 Relationships

The *IfcRelationship* is an abstract and standardised representation of all objectified relationships within the IFC format. This approach involves storing relationship-specific properties directly within the relationship entity itself. Relationships play a crucial role in capturing the connections, dependencies, and associations between different elements and components within the IFC model. The *IfcRelationship* entity inheritance is illustrated in Figure 3.4.

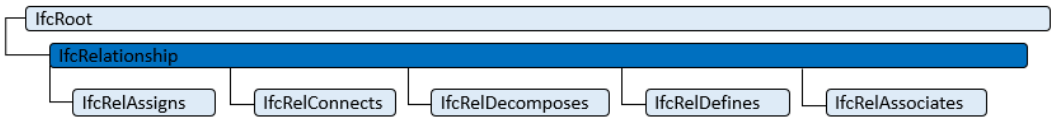


Figure 3.4: *IfcRelationship* entity inheritance.

IfcRelAssociates

The *IfcRelAssociates* relationship represents an association relationship within the IFC schema. It is used to establish connections between objects and external sources of information, such as classifications, libraries, or documents. *IfcRelAssociates* does not imply any dependency between the associated objects. A sub-type of the *IfcRelAssociates* is the *IfcRelAssociatesMaterial*.

3.3.8 IfcSpatialStructureElement

A spatial structure is used in the IFC schema in order to organise a building project. IFC uses a spatial hierarchy in order to organise and position its physical entities within the project. The structure breaks down the project into smaller parts based on location: many parts make up the whole. As defined in the IFC Specification, a spatial structure element, *IfcSpatialStructureElement*, are the elements that make up the spatial structure. *IfcRelContainedInSpatialStructure* is used in the structure in order to assign an element, such as *IfcBeam* or *IfcColumn*, to a certain level of the spatial structure. An element can only be contained within one spatial structure element and exactly which level it should be assigned to might vary depending on the project and region.

There are four spatial structure elements to which an element can be assigned to: to a site as *IfcSite*, to a building as *IfcBuilding*, to a storey as *IfcBuildingStorey* and to a space as *IfcSpace*. The highest level of the spatial structure is *IfcProject*. All spatial structure elements have to be associated with either another spatial structure element or the *IfcProject* by using the *IfcRelAggregates*. *IfcRelAggregates* is a 1-to-many relationship used to create a connection between the different levels of the spatial elements applied to sub-types of physical objects. The aggregation relationship has the attributes "RelatingObject", representing the whole within the whole/part relationship, and "RelatedObject", representing the parts within the whole/parts relationship.

3.3.9 Location

The location of a project site can be found by using the *IfcSite* entity. A site is defined in the IFC Specification as an area of land where the construction is to be/has been completed. The entity may include the definition for a single geographic reference point for the site.

The *IfcSite* entity contains the attributes listed in Listing 1.

```
#IfcSite= [id, type, GlobalId, OwnerHistory, Name, Desciption, ObjectType,
    ObjectPlacement, Representation, LongName, CompositionType, RefLatitude,
    RefLongitude, RefElevation, LandTitleNumber, SiteAddress]
```

Listing 1: The *IfcSite* attributes.

The attributes "RefLatitude" and "RefLongitude" of *IfcSite* return the compound measure *IfcCompoundPlaneAngleMeasure*. This provides the plane angle in degrees (D), minutes (M), seconds (S) and millionth-seconds (MS) of arc. The global positioning can be found by using WGS84 (World Geodetic System 1984) with longitude and latitude coordinates. An illustration of the procedure of finding the coordinates of an *IfcSite* is given in Figure 3.5. These coordinates found in the WGS84 represent the point (0,0,0) for the local placement of the *IfcSite*. The longitude and latitude values can be converted from a degree-minute-second-microsecond format to decimal degrees (DD) by converting the minutes, seconds and microseconds to degrees as shown in Equation (3.1).

$$DD = D + \frac{M}{60} + \frac{S}{3600} + \frac{MS}{3600 \times 1000000} \tag{3.1}$$

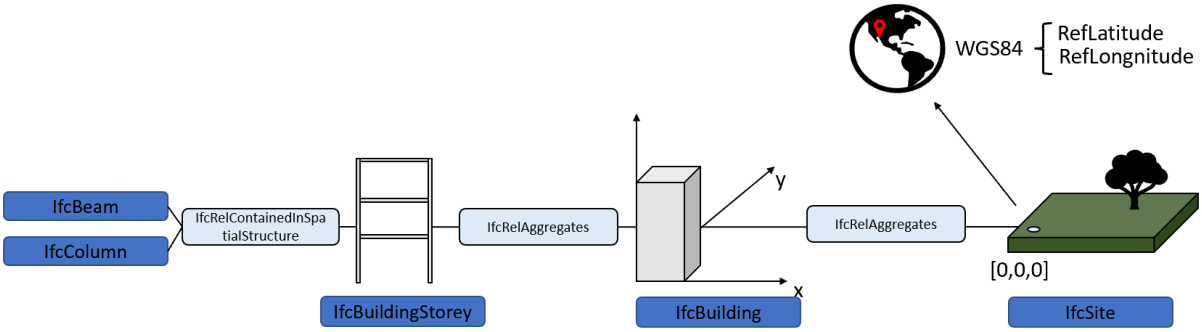


Figure 3.5: Procedure of finding the longitude and latitude coordinates of *IfcSite*.

IfcPostalAddress is defined as an address for delivery of paper-based mail and other postal deliveries" and has the attributes as given in Listing 2.

```
#IfcPostalAddress= [id, type, Purpose, Description, UserDefinedPurpose,
    InternalLocation, AddressLocation, PostalBox, Town, Region, PostalCode, Country
    ]
```

Listing 2: The *IfcPostalAddress* attributes.

3.3.10 Exporting as IFC

There are several IFC-certified software that can be used for the exportation of an IFC file, thereof Autodesk Revit, Tekla Structures and ARCHICAD, Allplan and MagiCAD [23]. Different software applications may structure and organise the exported IFC file in slightly different ways. These differences can impact how the IFC file is interpreted and utilised by other software or systems.

In the IFC model, an IFC entity refers to a uniquely defined object. The allocation of specific default attributes and dependencies within the IFC is determined by the assignment of the entity and type definition. Depending on these factors, the entity is equipped with predefined characteristics that establish its properties and relationships within the broader context of the IFC model [1]. Choosing the correct entities during the creation of an IFC file is therefore crucial when exporting the file. For example, if a beam is not assigned to the *IfcBeam* entity, the attributes required to describe the element clearly are not assigned. This may cause an incorrect interpretation of the element.

When creating an IFC file, numerous decisions must be made by the creator. Amongst others, this includes the assignment of property sets. The creator needs to decide whether to activate the exportation of IFC common property sets to ensure that the default properties defined in the IFC Specification are included in the exported file. Additionally, the exportation of base quantities also requires a decision from the creator. It is important to note that the IFC exporter exclusively transfers valid property and quantity values. In addition, the creator has the opportunity to define the level of detail within the file, which significantly affects both the file size and its interpretation. The level of detail for geometrical elements may therefore vary between different files.

3.4 Structural Optimisation

Optimisation refers to acquiring the best outcome under specific conditions [35]. In the field of civil engineering, structural optimisation has become a central tool for the development of sustainable and efficient designs. When dealing with structural optimisation, generally the aim is to find an optimal layout of structural components under prescribed conditions [29]. What is defined as the optimal layout is dependent on the desired objectives. The process, therefore, involves handling trade-offs between competing objectives, such as strength, weight, cost and safety. In structural optimisation with raw materials, the structural elements may be fabricated according to the optimal design. However, when dealing with structural optimisation using reused materials, a limited number of elements are available, each with predefined dimensional and mechanical properties. The optimal design of such a structure, therefore, includes an additional constraint. An alternative formulation of the optimisation problem in the context of reusing structural elements involves optimising the use of the available stock elements to meet the requirements of a predefined structure. The optimisation objective may entail minimising the amount of cut waste, i.e., unused material from the stock, or maximising the coverage of demand elements by utilising the available stock elements.

3.4.1 Form-fitting strategy

The process of form-fitting can be defined as finding good "fits" between a finite inventory of available structural elements and a desired structural form [9]. The report categorises the process into three main methods: "growth", "attraction" and "fitting". Applying the "growth" method acquires adding elements to one another sequentially to achieve a geometric fit. The "Attraction" method is based on adding elements onto a target geometry and then attracting or pulling them together to achieve a geometric fit. The "Fitting" method is based on replacing elements in a predefined structural geometry with elements from an available inventory meeting the requirements of the predefined structure [9]. The latter is the approach that will be used in this master thesis.

3.4.2 Bin Packing Problem

The Bin Packing Problem (BPP) is a combinatorial optimisation method used in form-fitting. Given a stock of bins with capacity C and n items with size $0 < s_i < C$, where i=1, ..., n, the BPP attempts to pack all items in a minimum amount of bins, so that the total size fit in the bins does not exceed the capacity. In terms of computational complexity, the problem is NP-hard [14].

There has been developed different algorithms for bin packing, whereof First-fit (FF) and Best-fit (BF) are two basic heuristics [9]. In FF the items are considered one by one and placed into the first bin they can fit. In BF the items are placed into the bin which results in the lowest remaining capacity in that bin. The quality, in terms of unused bin capacity, of both these solutions can be improved by pre-sorting the list of items that are considered. This can be done by sorting the elements in descending order by size and is called First-Fit Decreasing (FFD) or Best-Fit Decreasing (BFD). As the BP problems are NP-hard, the heuristic algorithms are a good fit to execute this operation and achieve quality results. [26].

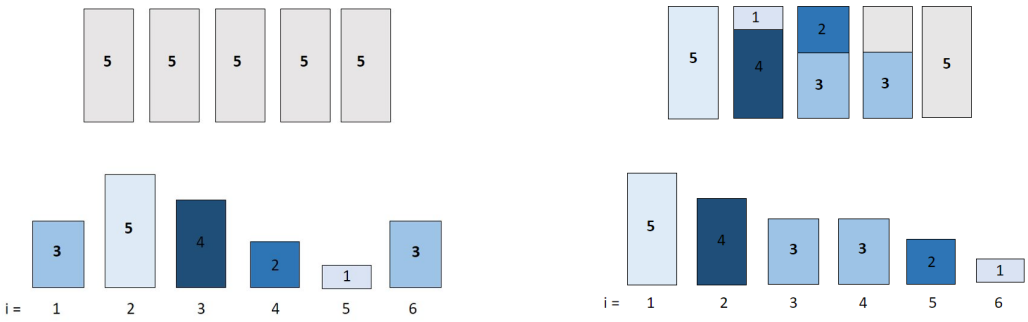


Figure 3.6: The First-Fit Decreasing Bin Packing Problem.

Bin packing problem in structural reuse of elements

A generalisation of the Bin Packing Problem can serve as a useful tool when studying the feasibility of reusing a stock of structural elements in a predefined structure. The stock of available structural elements are the items and the elements in the predefined structure are the bins. The constraints of the bins need to be defined, and may, amongst others, include the length, weight, material or cross-section. If considering the length of elements as the only constraint, in order for the items to fit in a bin, the length of the supply element needs to be longer or equal to the length of the demand element. The supply element is cut so it perfectly matches the demand length. The cut is placed back into the supply stock and may be used further to meet the demand for another element.

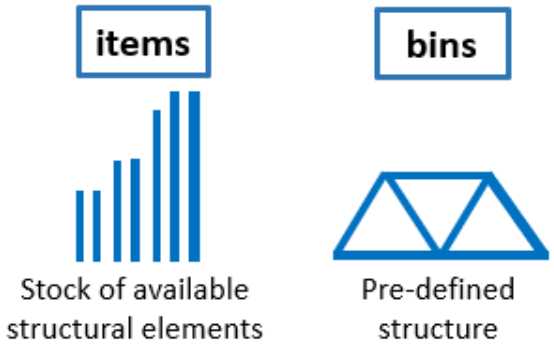


Figure 3.7: Bin packing problem for structural elements.

In the report "Form-Fitting Strategies for Diversity-Tolerant Design" by Bukauskas et al. [9], the performance of First-Fit and Best-Fit heuristics applied to a basic form-fitting problem was studied. For each algorithm, there were recorded results on the number of bins used to fit the items, the number of items which were not successfully fitted, and the total waste length.

It was observed that the performance of First- and Best-Fit heuristic algorithms did not differ significantly. The largest difference was related to the amount of waste material. The Best-Fit with minimising the remaining length, Best-Fit with items pre-sorted by decreasing effect and minimising the remaining length, and First-Fit with pre-sorting of items by decreasing effect and no sorting of bins were the best-performing algorithms based on all three metrics[9]. The running time of these algorithms was not considered in this study. However, this is an important aspect to consider if the algorithms will be used as a design tool. Implementing simpler and faster algorithms can be advantageous in the early stages of design processes, while slower and higher-quality algorithms can be advantageous in design refinement processes.

4 Software

This chapter provides an overview of the software and tools employed in the course of this master's thesis.

4.1 Visual Studio Code

Visual Studio Code is a simplified code editor that offers various development functionalities, including debugging, task execution, and version control. It focuses on providing essential tools to facilitate a fast code-build-debug process, while leaving more intricate workflows to comprehensive integrated development environments (IDEs) like Visual Studio IDE.

4.1.1 IfcOpenShell

IfcOpenShell is an open-source software library for working with IFC files. Amongst other tools, the software includes a C++ and Python API, making it possible to read, write, and manipulate IFC files programmatically. It is widely utilised by software developers, researchers, and professionals in the architecture, engineering, and construction industry. The libraries in IfcOpenShell have support for IFC2X3, IFC4 and IFC4X3. Compatible formats include IFC-SPF, IFC-JSON, IFC-XML and IFC-HDF5. IfcOpenShell is currently available for python version 3.6-3.11 [24].

Utility library

IfcOpenShell contains a utility library, providing a set of functions and tools for handling the information in an IFC file. Amongst others, the tools may be used to collect unit information and to extract entity property sets and quantity sets. The library includes shape processing features, reached though the shape-utility functions, and can be used to retrieve volume, outer surface area, and extrusion in x-, y-, and z direction. A full overview of the available utility functions and may be found in the IfcOpenShell 0.7.0 documentation provided by the IfcOpenShell Contributors [24].

IfcPatch

IfcPatch is a component in the IfcOpenShell library that allows for the predetermined modification on an IFC file. The modification is called an IfcPatch recipe and a full overview of the available recipes can be found in the IfcOpenShell 0.7.0 documentation under IfcPatch ([25]). The "ExtractElements" recipe in IfcPatch enables the selective extraction of specific elements from an existing IFC file, saving them as a new IFC file. An example of how to use the "ExtractElements" recipe is also provided in the IfcOpenShell 0.7.0 documentation.

Introduction to IfcOpenShell in Python

Listing 3 introduces the implementation of IfcOpenShell in Python, showcasing the basic commands required for data retrieval and attribute access.

```
# import the IfcOpenShell library
import ifcopenshell
# load the IFC file and store it as a variable
f = ifcopenshell . open ( file_path )
# Retrieve all instances of IfcBeam
beams = f . by_type ( " IfcBeam " )
print ( beams )
# Retrieve a specific instance of IfcBeam
beam = beams [0]
print ( beam )
# Retreive the attributes and their names in the IfcBeam instance
```

```
beam_info = beams [0]. get_info ()
print ( beam_info )
print ( beam_info . keys () )
# Retreive a specific attribute , e . g . GlobalId , of a specific instance
guid_beam = beam . GlobalId
print ( guid_beam )
```

Listing 3: Basic implementation of IfcOpenShell in Python.

4.2 Rhinoceros3D

Rhinoceros3D, or Rhino, is a 3D modelling and computer-aided design (CAD) software developed by Robert McNeel & Associates. Rhino uses non-uniform rational B-splines (NURBS) to precisely model geometry [34].

4.3 Grasshopper

Grasshopper is a visual programming and parametric modelling tool that runs within Rhino, also developed by Robert McNeel & Associates. The interface is a graphical algorithm editor and enables the creation of complex forms and the ability to generate alternative design rapidly without having any knowledge of programming or scripting. Figure 4.1 illustrates how a line can be created in Rhino using Grasshopper components. This type of visual programming is a "no-code" type of programming where codes are encrypted into the components. The functionality of Grasshopper can be extended through, amongst others, the use of Python or C# scripting to create custom components when desired functions are not available. These components can then be added to Grasshopper as plugins [34].

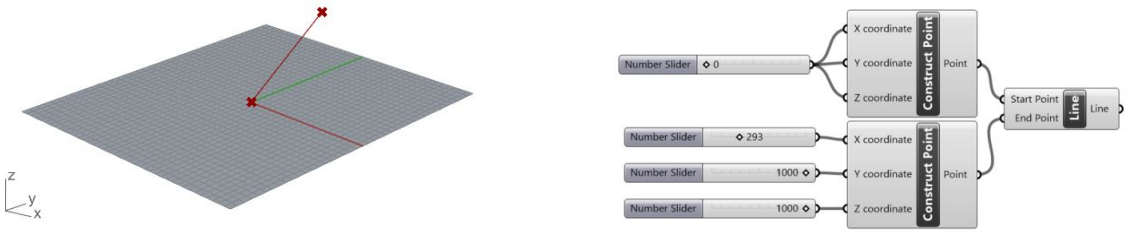


Figure 4.1: Rhino space and Grasshopper components.

Food4Rhino is the official plugin community serviced by McNeel and the users of the website can here find new Rhino plugins and Grasshopper Add-ons. In addition, the user may share their own applications and contact other developers [28].

4.3.1 Data Tree

A Data Tree in Grasshopper is a hierarchical structure for the storage of data in nested lists, enabling the storage of multiple lists within a single parameter. The structure of a Data Tree involves three main components: paths, branches and items.

The paths in a Data Tree specify the location of a branch within the tree. This path is given by numbers separated by brackets, for example {0;0;0}.The amount of numbers between brackets indicate the depth of the tree. Each sub-branch inherits the path number of its parent branch. At each of the branches there may

be stored a list of data items. Each data item is a part of one branch in the tree and has an index that specifies its location within the branch. The structure can be visualised as a tree, hence, the name Data Tree. An illustration of a Data Tree structure is given in Figure 4.2.

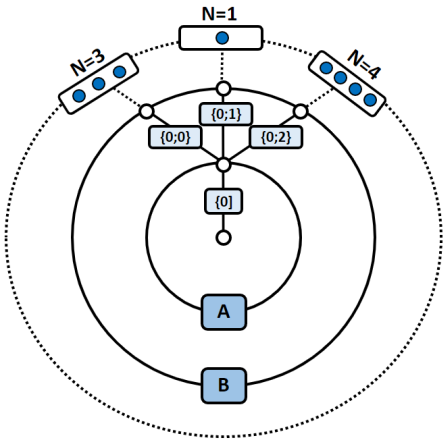


Figure 4.2: Example of a Data Tree structure.

The branch level A has path $\{0\}$, while the rightmost branch at branch level B has path $\{0;2\}$, where "0" is inherited from branch level A. Connected to the path $\{0;2\}$ is N=4 data items.

A practical example of the Data Tree structure can be the break down of a building into its different components and sub-components. A building may consist of several floors, several rooms within each floor, and several pieces of furniture within each room. In order to keep track of all the furniture within a specific room, the data may be stored as a Data Tree.

4.3.2 Geometry Gym

Geometry Gym develops utilities and plugins for various software applications such as Rhino3D, Grasshopper, Revit, Tekla, Navisworks, and more. Geometry Gyms has developed an IFC plugin for Grasshopper which allows for the import, export and modification of IFC files directly in the Rhino and Grasshopper environment. This makes it possible to perform structural analysis on already designed structures [17].

4.3.3 Karamba3D

Karamba3D, is a parametric structural engineering tool embedded in Grasshopper. The tool allows for the combination between parameterised geometric models, finite element calculations and optimisation algorithms performed in frames, spatial trusses and shells [12].

In order for a finite element analysis to be performed on a model in Karamba3D, there are several components that needs to be utilised. The "Assemble Model" component generates a finite element model. To accurately represent a structure in Karamba, the geometry, loads, and supports must be defined in this component. Further, the "Analyze" component computes the mechanical response of the model for a given load case. The component does not account for changes in length along the axial or in-plane direction that may occur as a result of lateral deformations. The "Analyze" component computes the model deformation, maximum nodal displacement, maximum total force of gravity, and internal deformation energy of structure. In order to preview the model response, the "Model View" and "Beam View" component can be used.

The "Model View" and "Beam View" components use the calculated model and allow the user to inspect the current state of the model and its elements [12].

Preconditions embedded in Karamba3D is listed below.

- Materials are assumed to behave in a linear elastic manner.
- Karamba3D expects all force-definitions to be in kilo Newton.
- Calculation of utilisation of steel elements is based on EN 1993-1-1, i.e. Eurocode (EC) 3. It takes into account buckling and lateral torsional buckling.
- Deflections are small as compared to the size of the structure.

5 Method

This chapter aims to systematically explain the implemented method for matching demand elements in a predefined structure to supply elements in a material database. The mapping is coupled with a structural analysis. In order to present the information in a clear and understandable manner, simple 2D, self-drawn, examples of structures are incorporated. Additionally, the workflows of the method are outlined to provide a comprehensive understanding of the process.

The method is divided into four main parts including *IFC Data Extraction* (Section 5.1), *Element Mapper* (Section 5.2), *Visual Mapper* (Section 5.3) and *FEM Analysis* (Section 5.4). Python in Visual Studio Code is mainly used for IFC data extraction and is given in Appendix B, while element- and visual mapping and the FEM analysis are developed in Grasshopper and is given in Appendix A. In addition, a supplementary assessment is included to address the quantification of the environmental and financial impact associated with reused and new elements (Section 5.5). An illustration of the method workflow is shown in Figure 5.1.

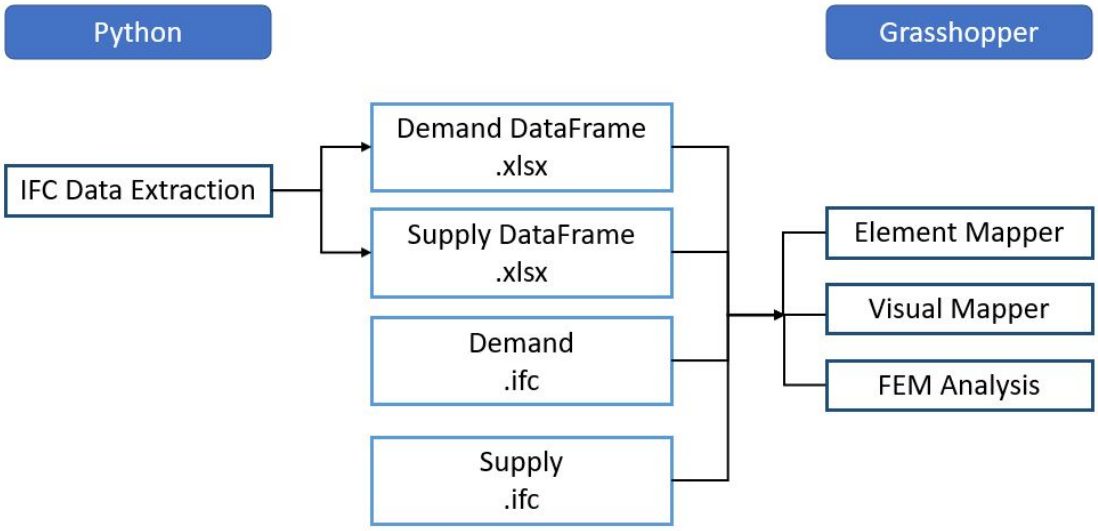


Figure 5.1: Method workflow.

5.1 IFC Data Extraction

In order to analyse the potential for structural reuse of elements given a predefined structure, a Python script was developed in Visual Studio Code. The script extracts necessary data from IFC files for the purpose of creating two DataFrames. One DataFrame with elements from a construction to be dismantled, and one DataFrame with elements from a construction to be assembled. An illustration of the IFC Data Extraction method workflow is shown in Figure 5.2.

The purpose of this script is to optimise the retrieval process of elements for future reuse projects by creating a comprehensive inventory of available materials. By simply inputting an IFC file into the script, all relevant information related to the available elements within the structure is efficiently retrieved. This retrieved information can then be utilised to identify suitable elements for future structures and has the potential to reduce the effort in mapping structural elements significantly.

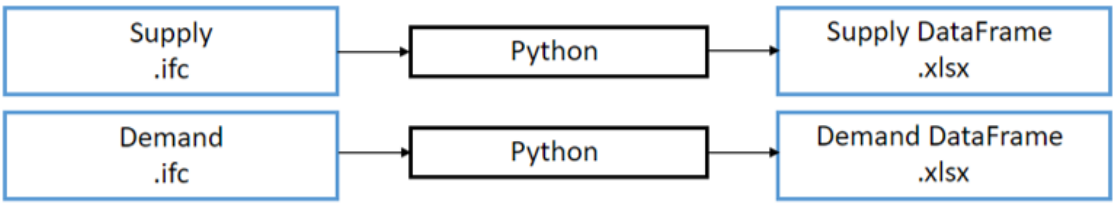


Figure 5.2: IFC Data Extraction workflow.

The method developed for Python-based IFC data extraction has been formulated through rigorous testing of multiple IFC files. The objective of this testing phase was to create a generic code that could efficiently handle diverse scenarios. To achieve this, the method was evaluated using a total of eight different IFC files, consisting of four files sourced from the industry and four files that were self-created. By analysing and assessing these varied files, the method aims to identify an optimal solution applicable to a wide range of IFC data sets.

The script utilises the IfcOpenShell library to extract necessary data from an IFC file. The desired elements to examine are the *IfcBeam*s and *IfcColumn*s. Once the data extraction is complete, the resulting DataFrames are exported to Excel. The necessary data to be extracted using IfcOpenShell for each element is the name, GUID, length, cross section area, cross section name, material and location. Additionally, supplementary information can be incorporated into the DataFrame to enrich the data set. This may include element height, width and volume. An example of a DataFrame created using the IFC Data Extraction method is shown in Figure 5.3.

	Guid	Name	Material	Length [mm]	Cross section area [m^2]	Cross section name	Latitude	Longitude	Material name Karamba3D
0	33as9O3zjEnAsHHJOULXyg	Beam IPE:IPE 270:1393897	Steel	34674,6	0,0046	IPE270	42,41486359	-71,2580719	s355
1	33as9O3zjEnAsHHJOULXyw	Beam IPE:IPE 270:1393913	Steel	11274,6	0,0046	IPE270	42,41486359	-71,2580719	s355
2	33as9O3zjEnAsHHJOULXxA	Beam IPE:IPE 270:1393929	Steel	34674,6	0,0046	IPE270	42,41486359	-71,2580719	s355
3	33as9O3zjEnAsHHJOULXxO	Beam IPE:IPE 270:1393947	Steel	11274,6	0,0046	IPE270	42,41486359	-71,2580719	s355
4	33as9O3zjEnAsHHJOULXwJ	Beam IPE:IPE 400:1394000	Steel	2760	0,0085	IPE400	42,41486359	-71,2580719	s355
5	33as9O3zjEnAsHHJOULXwV	Beam IPE:IPE 400:1394012	Steel	4740	0,0085	IPE400	42,41486359	-71,2580719	s355

Figure 5.3: IFC Data Extraction DataFrame example.

In the case of large IFC files, it may be beneficial to create a new IFC file with only the wanted elements, e.g. *IfcBeam* and *IfcColumn*, by utilising IfcPatch. This new filtered IFC file should then be used as input for when creating the DataFrame, to ensure compliance for later in the method. The code for this is given in Appendix B.

5.1.1 Extracting quantities

If the quantity sets are stored in the IFC file, they can be extracted by using the element utility function "ifcopenshell.util.element.get_psets". However, as the quantity sets are not required to be stored in the model, this function may give the output of all, limited or no quantities. Instead, the extraction of quantities is performed by utilising the shape utility function, "ifcopenshell.util.shape". Through the execution of eight tests on various IFC files it was determined that the data presented in Table 5.1 is consistent across all files. The values extracted was compared with the values obtained by reading the files in Solibri.

Assuming straight elements with a constant cross section along their length, it is a straightforward process to calculate the cross section area given the volume and length of the element.

Table 5.1: Shape utility functions for the retrieval of length, height, width and volume for *IfcBeam* and *IfcColumn*.

ifcopenshell.util.shape				
	get_x	**get_y**	**get_z**	**get_volume**
IfcBeam	length	height	width	gross volume
IfcColumn	height	width	length	gross volume

5.1.2 Extracting cross section name

The retrieval of the cross section name is accomplished by calling on the attribute "ObjectType" for the *IfcBeam* and *IfcColumn* entities. This attribute is used to store user-defined values for the sub-type *IfcObject*. Alternatively, it has been observed that the cross section name is often stored under the property definition "Reference" in the "Pset_BeamCommon" and "Pset_ColumnCommon" property sets.

5.1.3 Extracting material

The associated material of an element in an IFC file can be retrieved by using the element utility function "ifcopenshell.util.element.get_material". This requires for the material data to be populated consistently in the model. In order to create a more generic material retrieval code, the material is instead retrieved by directly accessing the spatial structure and finding the relations of the element. To find the associated materials, the script examines whether the element has any associations by checking the "HasAssociations" attribute. If associations exist, the code proceeds to iterate through each association. For each association, it checks whether it belongs to the *IfcRelAssociatesMaterial* type, which signifies a relationship between an element and a material.

5.1.4 Extracting location

To retrieve the longitude and latitude coordinates of a site from an IFC file, the *IfcSite* entity is utilised. By accessing the attributes "RefLongitude" and "RefLatitude" of the entity, the corresponding values are extracted. To ensure compatibility with the global coordinate system, the extracted coordinates is then converted to the WGS84 standard, by using Equation (3.1). Within the *IfcSite* entity, there is an attribute called "SiteAddress", which corresponds to the *IfcPostalAddress* entity type. It is observed during the creation of IFC files in Solibri that the address assigned to the structure is automatically associated with the *IfcPostalAddress* entity as a default. It is important to note that this behaviour may vary depending on the software used to generate the IFC file.

5.2 Element Mapper

The Element Mapper matches a stock of supply elements to a stock of demand elements based on defined constraints. The objective of the method is to optimise the process of identifying suitable matches between elements. The mapping is done through a Python script in Grasshopper using a first-fit strategy. The form-fitting problem of this Grasshopper component can be compared to a generalisation of the Bin Packing Problem described in Section 3.4.2. The supply elements represent the items, and the demand elements represent the bins. The function of the Grasshopper component is to match supply elements with demand elements, generating the minimum amount of cut waste based on length. It assumes that all available supply elements are of sufficient quality and can be safely reused.

5.2.1 Element Mapper: input

The Element Mapper component has three inputs, a supply DataFrame, a demand DataFrame and a boolean toogle. The boolean toogle can be set to either True or False. The DataFrames each contain relevant information about the elements in their corresponding structure, extracted as explained in Section 3.4.2.

Prior to DataFrames entering the Element Mapper, some additional operations are performed. The moment of inertia is calculated by based on IFC geometry that is imported directly to Grasshopper and converted to breps. This will be further explained in Section 5.4.1. The moment of inertia is found based on these breps and their center lines. The orientation of the cross section plane is located. Curves are found from the edges of the cross section. This geometry is the input of the "Area Moments" component that calculates the moment of inertia about all its axes. The moment of inertia about the y-axis is added to the demand and supply DataFrame for further use. The order of Grasshopper components used to find the moment of inertia is presented in Figure 5.4.

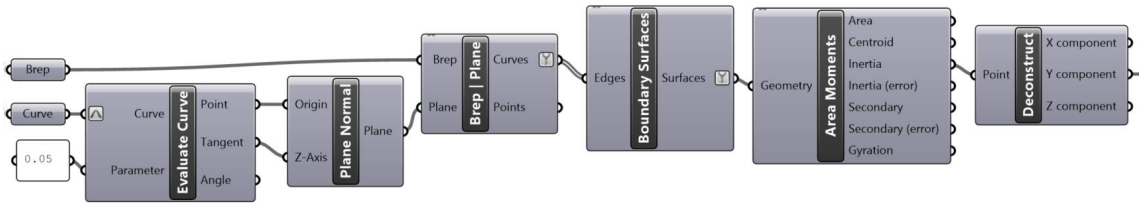


Figure 5.4: Calculating moment of inertia with Grasshopper components.

In addition, curved elements are identified in order to prevent the matching of curved and straight elements. This is achieved by using the Grasshopper component "ggIFC Decompose Extruded Area Solid" where the IFC geometry is utilised. The operation removes the curved elements for the list of breps. The indexes of the filtered out elements are used to remove the elements from the DataFrame as well. This ensures correspondence between the DataFrame and its IFC geometry. DataFrame and IFC geometry now only consist of straight elements.

Prior to connecting the DataFrames to the Element Mapper component, the sets are sorted based on their element lengths. The demand DataFrame is sorted in decreasing order, while the supply DataFrame is sorted in ascending order, making the algorithm at hand a First-Fit-Decreasing Bin Packing Problem. This reordering is also applied to the IFC geometry data to assure compliance between the DataFrames and their respective IFC geometry. This assures that the correct moment of inertia is assigned to the correct element and that the indexes between the DataFrames and IFC geometry correspond in the further procedures, as will be presented in Section 5.3 and Section 5.4. An illustration of the sorting of the elements is shown in Figure 5.5. This sorting causes the elements to retrieve new indexes within Grasshopper. However, since the DataFrames and IFC geometry are sorted based on the same condition, the element indexes will still correspond to each other. Since the GUID of an element is associated with a specific element and not with a specific index, this sorting does not cause problems in later tracking of the correct elements.

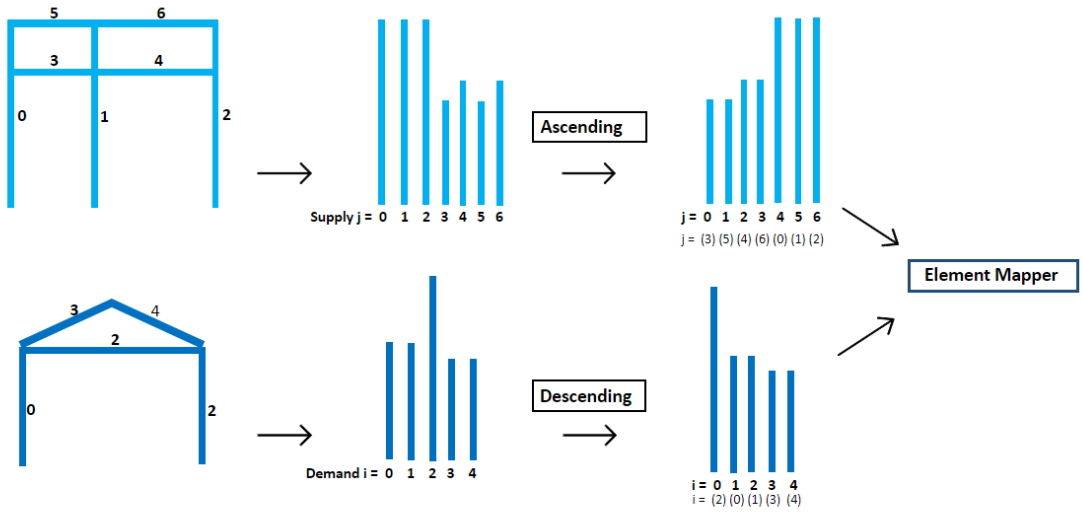


Figure 5.5: Visualisation of the processing of demand and supply structures in Grasshopper prior to entering the Element Mapper.

5.2.2 Element Mapper: matching conditions and process

A match between elements is found when the following property of the supply element is larger or equal to the corresponding demand property: length, 80% of the cross section area and 80% of the moment of inertia. In addition, the mapper only considers demand and supply elements that have assigned the material "Steel". In order to explore the implication of resource optimisation, the cross section area and moment of inertia requirement is set to 80% for a successful match. This criterion allows for greater design flexibility, potentially leading to an increased number of matched elements and the potential for material optimisation.

The first demand element, *i*, is compared with the first supply element, *j*, from each sorted DataFrame. If a match is not found, element *i* will be compared with element *j+1*, and so forth. Once a match is found between a demand element *i* and a supply element, the algorithm proceeds to compare the next demand element, *i+1*, with the remaining supply elements.

If the boolean toggle is set to False, the matched element in the supply list is deleted, regardless of how much of the element was utilised. Each element in the supply list can therefore only be assigned to a single element in the demand list. If the boolean toggle is set to True, the remaining length of the supply element is updated in the supply list. If the demand and supply lengths are a perfect match, the supply element is removed from the supply list. This allows multiple demand elements to be assigned to a single supply element. In the further method, the boolean toggle is set to True.

5.2.3 Element Mapper: output

The output of the Element Mapper is the indexes of each demand element and its matching supply element, with unmatched demand elements assigned a null value. Additionally, the component calculates and provides volumes of matched supply elements, unmatched demand elements, and cut waste, which are subsequently employed in the calculation of environmental and financial impact, as will be explained in Section 5.5. Figure 5.6 provides a visual representation of matched and unmatched volumes and the resulting cut waste. Notably, in the specific example being illustrated, all demand elements have successfully found a match.

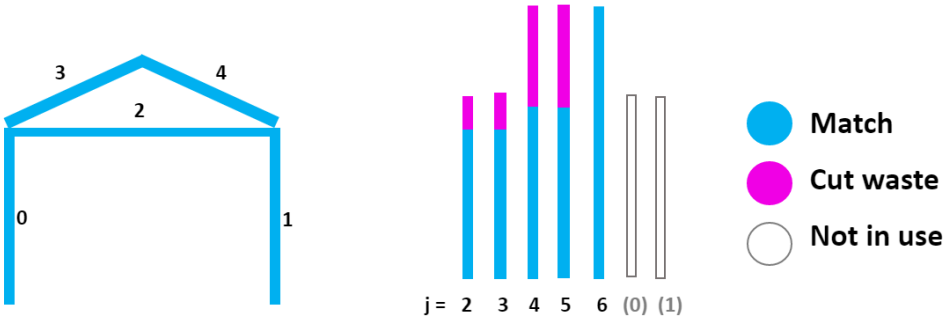


Figure 5.6: Structure to the left illustrated the demand structure. Elements to the right are the available supply elements. The colour-coding indicated matched, unmatched and cut waste volume.

The data structure of the Element Mapper output is changed through a C# component in Grasshopper. The new mapping is stored as a Data Tree, where the paths are the supply element index and the items within the branches are the corresponding matched demand indexes. If a supply element is utilised for several demand elements, the demand element indexes will be placed under the respective path of the supply element index. The supply and demand elements with no match are not included in the new Data Tree. This sorting of indexes is shown in Figure 5.7.

Sorting the mapping indexes as a Data Tree offers several benefits, one of which is the ability to maintain the index of each element for both demand and supply. This allows for the retrieval of the unique identifier of any element from any part of the script. Another benefit of the use of a Data Tree is the possibility of merging trees while remaining path indexes. It is then possible to work with separate paths of a Data Tree based on, for example, a dispatch pattern. Operations can be performed on the respective paths and later merged back together to the original Data Tree.

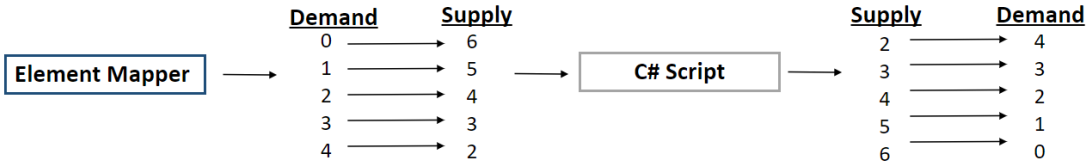


Figure 5.7: Sorting of element indexes in Grasshopper script following the Element Mapper.

The DataFrames created by the IFC Data Extraction are later utilised in the FEM analysis, which is further explained in Section 5.4. In order to perform a FEM analysis in Karamba3D, it is necessary to define the cross sections and material names of the elements. These property names in Karamba3D can be viewed using the "Cross section Range Selector" and "Material Selection" components in Grasshopper. The objective of the properties in the DataFrame is to match the respective presentation format in these components exactly, enabling Karamba to interpret the properties correctly. If the properties retrieved in the IFC Data Extraction, are not an identical match, these need to be manually edited.

5.3 Visual Mapping

In order to perform a visual mapping of the elements, the element geometry from the demand and supply IFC files is used as well as the output from the Element Mapper. The IFC geometry is sorted in compliance with the result from the Element Mapper and C# script. The successfully matched supply elements are flipped vertically to align with the same orientation and then positioned within a grid. The elements are trimmed accordingly to the demanded length. The remaining length of the element marks the cut waste. The demand and their corresponding matched supply elements are connected with a line. Figure 5.8 illustrates this visual mapping, providing a clear representation of the placement of supply elements in relation to their respective demand elements.

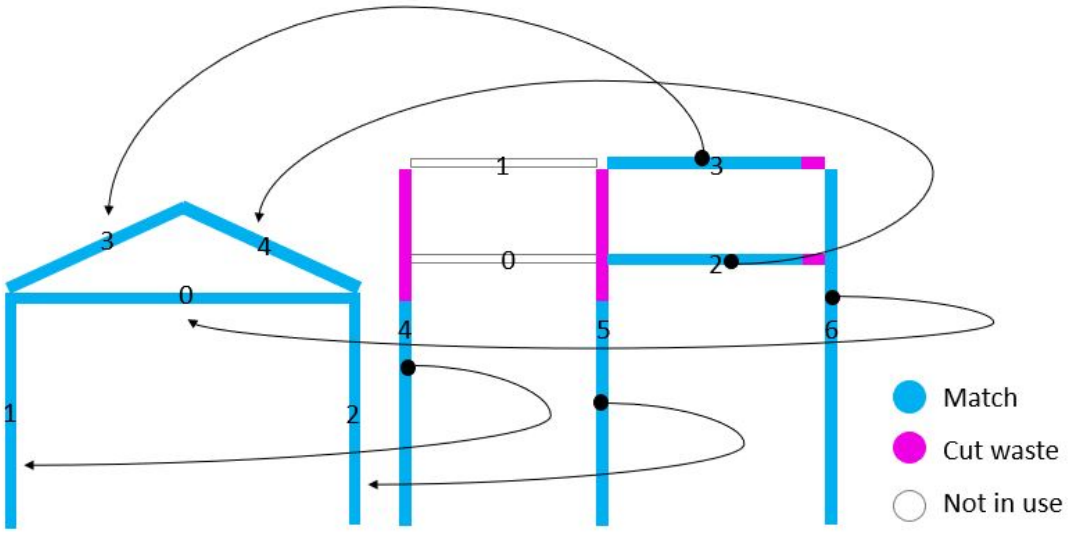


Figure 5.8: Visual mapping of demand and supply elements by using the Visual Mapper.

5.4 FEM Analysis

Following the successful matching of supply and demand elements, the demand elements are then replaced with their respective supply counterparts. To ensure the structural integrity and safety of the construction with the substituted elements, a structural analysis is conducted. The utilisation of FEM analysis within Karamba3D offers significant potential to optimise the structural verification process in a reuse project. This approach leverages the geometric data extracted directly from IFC files, offering a more efficient alternative compared to constructing a model from the ground up using conventional FEM design tools.

To ensure a precise and effective structural analysis using Karamba3D, it is essential to establish the correct assembly of the digital model. While importing an IFC file into Grasshopper provides the visual geometry of the model, certain adjustments are required to accurately depict the elements for analysis purposes.

Utilising the Karamba components, the structural analysis provides visualisations of deformation, deflections, bending moments, axial forces, and shear forces. These visualisations enable a comprehensive understanding of the structural behaviour and aid in assessing the overall performance and safety of the construction.

5.4.1 Establishing the analytical model

Import of IFC Geometry

The imported IFC geometry data is used as a base for establishing the analytical model. The data is sorted and converted to brep before further processing is performed. The breps are organised as a Data Tree with the same ordering as the DataFrame from the Element Mapper.

Extracting center lines

To meet the requirements of Karamba3D, the elements within the demand structure need to be represented by lines. This is done by deconstructing the breps and locating the end surfaces, cross sections, of each brep. The center point of each surface is located, and the points are subsequently connected with a line. The center lines are later assigned the appropriate properties, including cross section and material, based on the outcomes obtained from the Element Mapper. This is done using the "Line to Beam (Karamba3D)" component. This ensures that the elements are properly characterised and ready for the subsequent analysis.

Sorting based on level and orientation

To properly assign properties and loads to correct elements upon assembly of the model, the elements are sorted based on level and orientation. Sorting based on level is done by evaluating the z-value of the middle point of each line. The sorting is parametrically controlled through a "Number slider" component that defines the z-value. The components "Cull Pattern" or "Dispatch Pattern" is used to extract or remove desired elements from lists. An excerpt from the Grasshopper script performing this procedure is presented in Figure 5.9. Sorting of orientation is done by evaluating the vectors and curves representing the center lines and separating them based on their value. For example, a line representing a horizontal beam will have an angle of 0.5π from the z-axis. The components behind this extraction are presented in Figure 5.10. The same method is performed on the other lines, but with corresponding values to separate columns from diagonals.

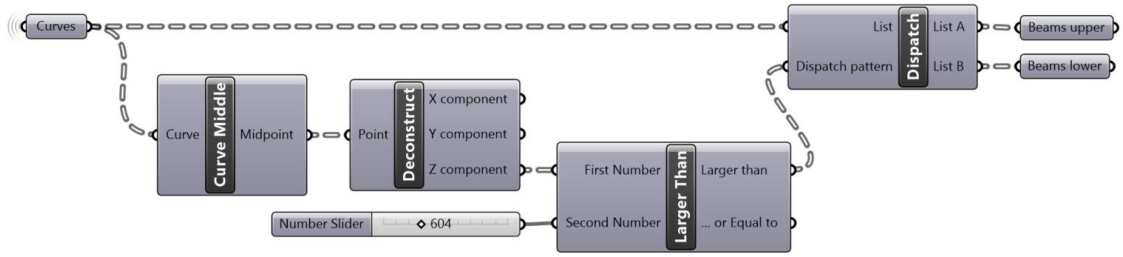


Figure 5.9: Grasshopper - sorting lines based on z-coordinate.

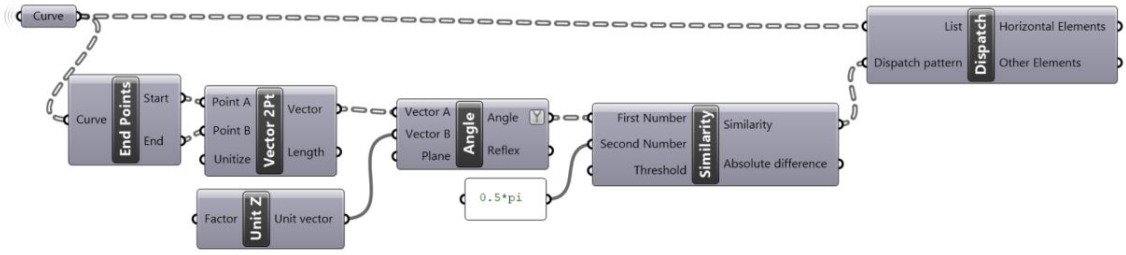


Figure 5.10: Grasshopper - sorting lines based on orientation.

Model consistency

Ensuring model consistency is important as Karamba3D requires the elements to intersect at all nodes. If this requirement is not met, Karamba3D will interpret the elements as disconnected and the model will provide inaccurate results during analysis. By examining the breps converted from the IFC geometry data, it

appears visually in Rhino that the elements intersect. However, when utilising the center lines instead of the breps, these represent the correct length but do not intersect. To establish connections, artificial members are implemented. This involves locating the node at the desired intersection location and connecting the node-point to the point of either the start or end of an element. The approach to locate the nodes is presented in Figure 5.11. The input of the "Perp Frames" component is lines of all elements that potentially are connected. The "Cull Duplicates" component allows for duplicate points to be removed from the list of points. It is also possible to control the *Tolerance* of this. In the intersection point between two beams and a column, there would originally be three points. Instead of creating three nodes, only one is created by removing the duplicates within 80 cm, in this specific illustrated example.

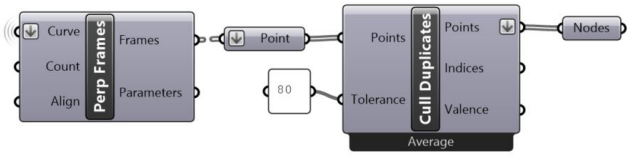


Figure 5.11: Grasshopper - locating nodes with a *Tolerance*

The approach to creating the artificial elements is presented in Figure 5.12 and a visual representation of an example of such elements is presented in Figure 5.13. The illustration shows how the artificial elements represent an extension of the already present elements. The artificial elements are assigned high stiffness and low mass to minimise their negative influence on the overall system response. This is done using the cross section type *Spring* and assigning it a high rotational and transnational stiffness, as shown in Figure 5.14. This created a weightless element that interferes minimum with the overall structural behaviour of the assembled model.

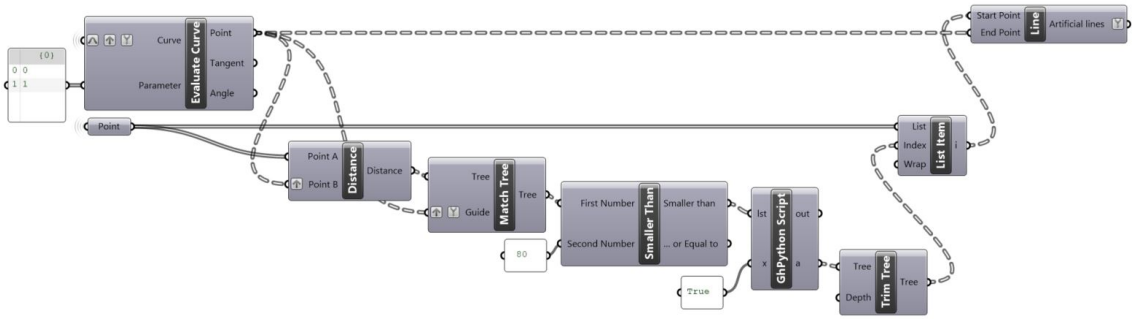
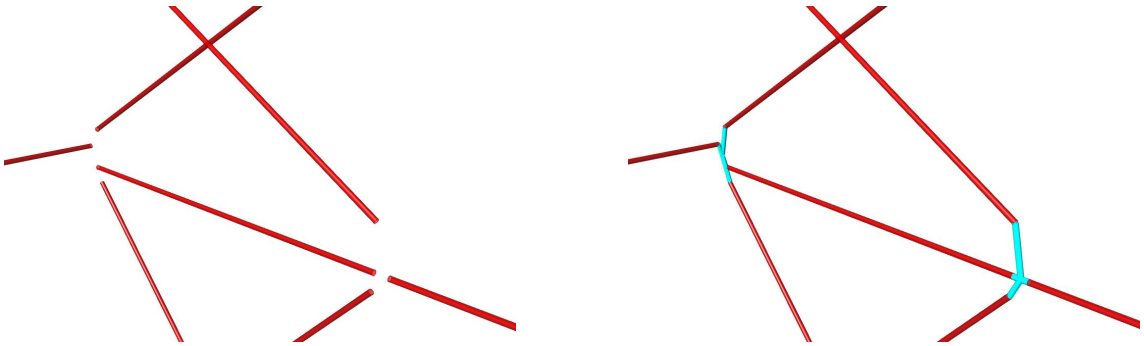


Figure 5.12: Grasshopper - creating artificial elements.



(a) Before.

(b) After.

Figure 5.13: Model before and after implementing artificial elements.

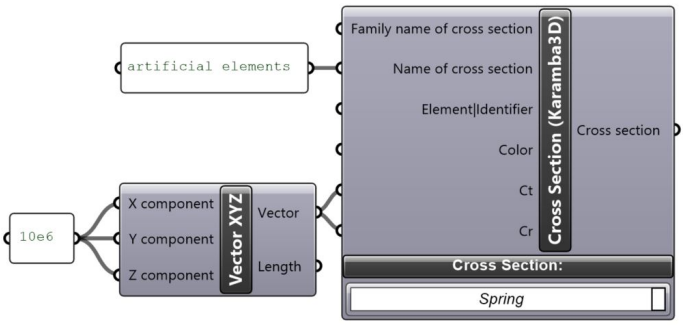


Figure 5.14: Grasshopper - artifical elements cross section.

In IFC files, columns are commonly represented as continuous elements that extend through multiple levels, with beams and other elements connected to them on each floor. While this representation aligns with the physical intent, it is necessary to split the columns at the points where they are connected to other elements in order to accurately depict the connections in Karamba3D. The same applies to other elements that are modelled as continuous but are connected to different elements. Consequently, all lines are divided into smaller segments based on the intersections with other elements. To achieve the desired outcome, the process involves locating a point on the line that is in close proximity to the node representing the intersection point. Subsequently, the curve is split based on the parameter associated with this point on the line. The specific implementation of this approach is outlined in Figure 5.15. The parameter "Second Number" will in this case define the allowed distance between the node and curve. The "Cull Patterns" component will remove the parameters of the points that are not in close proximity to the node.

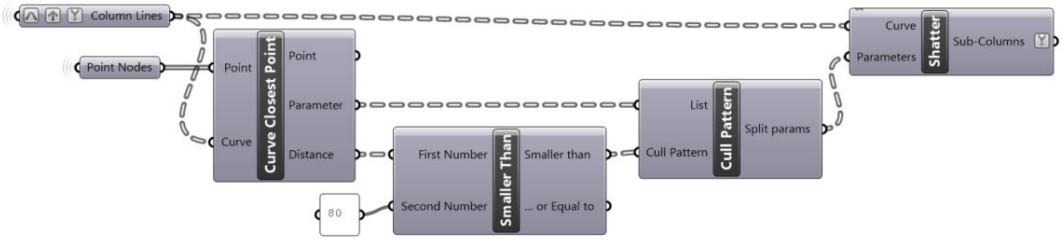


Figure 5.15: Grasshopper - splitting lines based on intersections.

Supports

When considering the supports for a structure in three-dimensional space, a body has six degrees of freedom, three translations and three rotations. For the accurate calculation of the deflected state in Karamba3D, the supports need to be designed in a manner where no translations or rotations are possible without causing a reaction force. Supports are implemented in the model with the component "Support (Karamba3D)", where the six degrees of freedom can be controlled. The component is depicted in Figure 5.16, configured as a simply supported that is constrained against torsion.

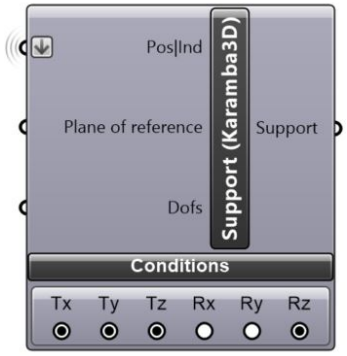


Figure 5.16: Simply supported supports in Karamba3D.

By default, all connections in Karamba are fixed unless otherwise specified. To create other types of connections the "Beam-Joint-Agent (Karamba3D)" component is utilised. This component enables control over six degrees of freedom and allows the designer to assign the connection to specific members based on their element identifiers. Using the component, it is possible to create pinned, semi-rigid, or rigid connections by defining the inputs of the component. This flexibility allows designers to customise the connections according to their specific requirements. Figure 5.17 shows how a "Beam-Joint-Agent (Karmaba3D)" component represents a pinned connection where the rotational stiffness is high, meaning that the joint is restricted against torsion.

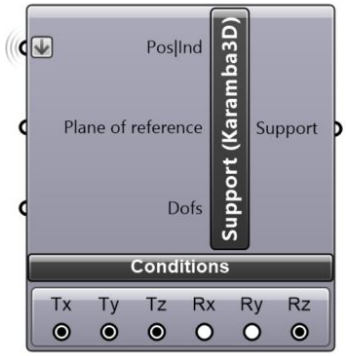


Figure 5.17: Pinned joint connections in Karamba3D.

Load case

In order to obtain a realistic outcome from the analysis, appropriate loads need to be applied to the structure. The magnitude of these loads can be adjusted by modifying the connected value parameter for that specific load component. The application of loads varies depending on the desired or demanded conditions.

The loads are applied using the component "Loads (Karamba3D)" that can be assigned a *Vector*, *Mesh*, and *Element Identifiers*. The *Vector* controls the magnitude of the load, while the *Mesh* parameter defines the area of which the load will be distributed. By using the *Element Identifiers* parameter, the user can specify the elements on which the assigned load will be applied.

Assembly

The model is assembled using the "Assemble Model (Karamba3D)" component by connecting all elements, supports, joints and loads together.

5.4.2 Analysis and results

For analysis purposes the "Analyze (Karamba3D)" component is employed to evaluate the model. Subsequently, the results are visualised using the "Model View (Karamba3D)" component. This component enables the visualisation of deformations, reactions, loads, supports, local axes, and joints. Section forces, utilisation and deformation are displayed with the "Beam View (Karamba3D)" components.

5.5 Quantifying Financial and Environmental Impact

Quantifying the financial and environmental impacts of the structural element reuse process is crucial for conducting a comprehensive evaluation of the implemented method in real-world scenarios. This assessment examines trade-offs associated with executing such projects and establishes the foundation for determining their financial and environmental pros and cons.

The environmental impact is measured in terms of Global Warming Potential, GWP. The following assumptions are made in the calculations for environmental impact and cost of reused and raw steel materials.

- The supply elements are cut at the demolition site.
- The transportation and other processing of the cut waste are not included in the calculations.
- The transportation of new elements from fabrication to the construction site is set to 100km.
- The structural integrity of supply elements after demolition is intact.
- The transport of elements to be reused is set to the distance between the demolition site and the construction site.

When quantifying the environmental and financial impact, the values presented in Table 5.2 are used.

Table 5.2: Environmental and financial impact key numbers.

	Unit	Value
Steel GWP [30]	[kg CO_2 per m^3]	9263
Steel reused GWP [21]	[kg CO_2 per m^3]	278
Steel cost [15]	[NOK per kg]	67
Steel reuse cost [15]	[NOK per kg]	67
Steel density [**EC3NS-EN1993-1-1:2005+A1:2014+NA:2015**]	[kg per m^3]	7850
Transportation cost [19]	[NOK per km per tonn]	4
Transportation GWP [31]	[gram per tonn per km]	89.6
Valuation GWP [33]	[NOK per kg CO_2]	0.7

The methodology and assumptions employed to calculate the financial and environmental impact is a simplified approach. The method does therefore not retrieve the precise values. Nonetheless, the obtained results serve as an indicative measure of the potential realistic outcomes and are sufficient for the purpose of this master's thesis.

6 Case Studies

In this chapter, two case studies are presented, offering a comprehensive examination of the method employed in each study and any modifications implemented throughout the process. To ensure the reliability of the obtained results from the structural analysis, a comparative study is subsequently conducted. The aim is to compare the results of the FEM Analysis made in Grasshopper with those obtained using a custom-constructed model developed with conventional FEM design tools.

Additionally, a detailed discussion is carried out regarding the results obtained from the Element Mapper and FEM Analysis, along with specific observations made during the testing phase. This discussion aims to provide a deeper understanding of the applied strategies of the specific case and underlying assumptions, thereby shedding light on the implemented strategies.

The "Original Structure" refers to the demand structure composed of the original cross sections and materials from the IFC file, while the "Reuse Structure" represents the updated structure with elements featuring reusable materials, as determined by the Element Mapper. These definitions will be used further in the case studies.

6.1 Case Study 1: Test Files

The first case study is based on two self-produced IFC files. They have been produced using Revit 2023 and then exported to IFC2X3 format. Both structures are situated in different locations in Norway and consist of the structural steel S275 and S355. The supply structure consists of 82 elements, while the demand structure consists of 30 elements. All *IfcBeams* and *IfcColumns* in the files are straight elements with typical dimensions expected for beams and columns. Figure 6.1 showcases the two structures, providing a visual representation of their relative sizes. Both demand and supply structures are non-symmetrical.

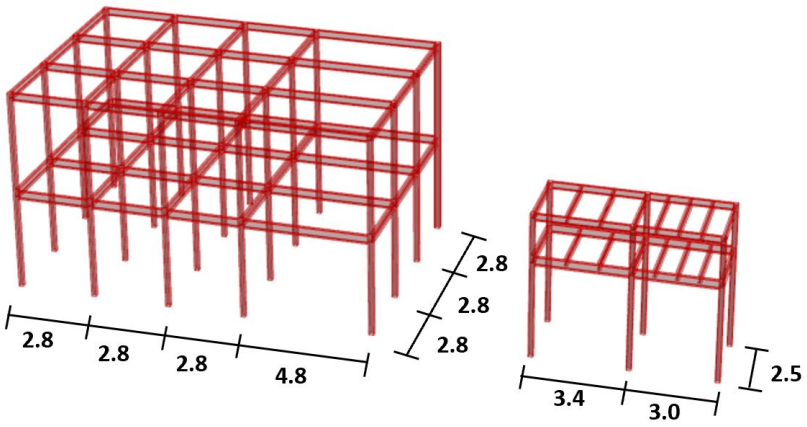


Figure 6.1: Structures in Case Study 1. Supply structure to the left, demand structure to the right. Unit in meters.

6.1.1 Case Study 1: IFC Data Extraction

The Python code developed in the IFC Data Extraction demonstrated satisfactory performance when applied to the two files created specifically for this case study. These test files effectively aligned with the functionality of the implemented method. However, some observations were made during the extraction of

the structure locations that shed light on an issue that may arise when handling IFC data structure and the use of IfcOpenShell.

Cross section and material names

The extraction process effectively retrieved the cross section types and material information for all elements within the demand and supply structures. Manual editing of the cross section names was performed to ensure recognition in the "Cross Section Range" component in Karamba3D. All the required cross section types and materials were already present in the database, but need to be formatted correctly to ensure compatibility.

Location

It was observed during the creation of the IFC files in Revit that the address of the structure was automatically associated with the *IfcPostalAddress* entity as a default. However, the sorting of the attributes within the entity was not as expected. The attributes of the *IfcPostalAddress* entity are intended to be sorted as mentioned in Section 3.3.9. The collected entity with associated attributes from the IFC files is given in Listing 4, for the supply and demand structure, respectively.

```
#111= IFCPOSTALADDRESS(\$,\$,\$,\$,('Enter address here'),\$,'','H\X2\00F8\X0\
    gskoleringen','5','7034 Trondheim');

#111= IFCPOSTALADDRESS(\$,\$,\$,\$,('Enter address here'),\$,'','Elizabeth','
    Stephansens vei 15','1433 \X2\00C5\X0\s');
```

Listing 4: The *IfcPostalAddress* of Case Study 1 for the supply and demand stucture respectivly. Collected by opening the IFC files in Visual Studio Code.

In both *IfcPostalAddress* entities, the expected and actual positioning of the attributes do not correspond. An examination was carried out to determine whether the expected attribute positioning was incorrect. This involved accessing the attributes both by their names and by their positions in the attribute list. The results obtained were consistent, leading to the conclusion that the default attribute assignment in Revit, does not necessarily align with the expected sorting as given in Section 3.3.9.

To ensure a clear illustration of the previously explained problem, an example will be provided. There are two ways to access the postal code information of the structure. One option is to use the attribute name "PostalCode" of the *IfcPostalAddress*, while another is to retrieve it by accessing the attribute in position seven. In the supply structure, calling the attribute name "PostalCode" returned "5", while position seven returned "Høyskoleringen". The expected return was "7034". However, having access to the *IfcPostalAddress* entity by using the command "get_info" as shown in Listing 3, the attributes assigned can be retrieved. The desired information can be extracted by referencing the position within the entity. It is important to note that this behaviour may vary depending on the software used to generate the IFC file.

Alternatively, manually adding the correct address into the Excel DataFrame is a simple solution.

6.1.2 Case Study 1: Element Mapper

The Element Mapper achieved a successful match for 28 out of the 30 elements in the demand structure, resulting in an approximately 90% match. For this match, the demand structure utilised 28 elements from the supply bank, indicating no supply element was used for more than one demand element. As a result, the mapping produces a total cut-waste of 0.13 m^3. Furthermore, the Reuse Structure obtained a volume approximately 1.1 times larger volume than the Original Structure. Key results of the Element Mapper for Case Study 1 are shown in Table 6.1.

Table 6.1: Element Mapper results in terms of volume given in m^3.

	Volume [m^3]
Total volume of Original Structure	0.46
Total volume of available supply elements	1.70
Volume of unmatched demand elements	0.03
Volume of total used supply elements	0.62
cut-waste volume	0.13
Utilised volume of supply in Reuse Structure	0.48
Reuse Structure	0.52

6.1.3 Case Study 1: Visual Mapping

Figure 6.2 provides a visual representation of the demand structure and the supply elements for Case Study 1. The colours assigned to the elements signify whether they are matched, not matched or cut-waste. Figure 6.3 highlights the visual mapping between one demand element and its respective supply element match. Figure 6.4 illustrates the visual mapping between all the matched demand and supply elements. Only the matched supply elements are visualised in the figures. After a match is made between the demand and supply elements, the cross sections are substituted accordingly. The new cross sections placed in the demand structure are illustrated in Figure 6.5.

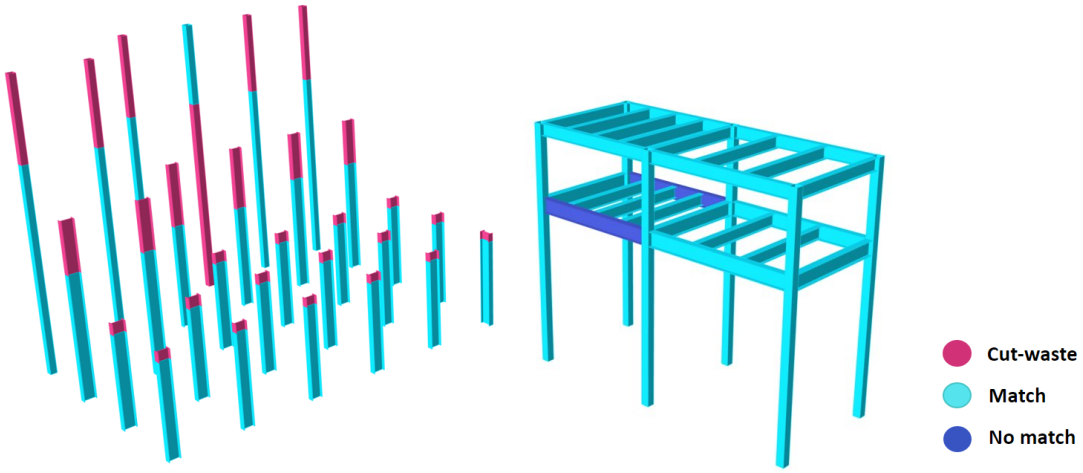


Figure 6.2: Supply elements and demand structure for Case Study 1. Match, no match and cut-waste elements are highlighted. Relevant supply elements are visualised to the left.

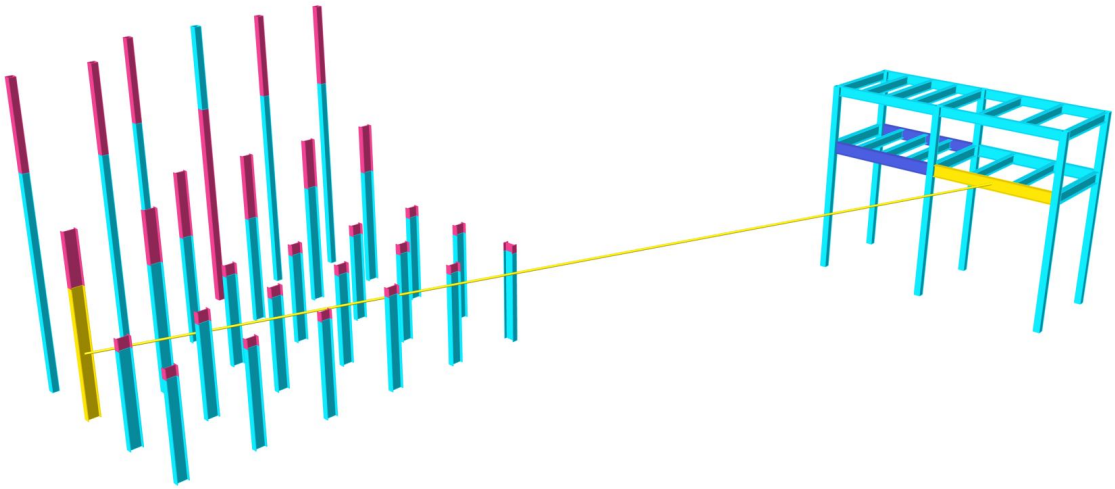


Figure 6.3: Visual mapping between one demand element and its respective supply element match.

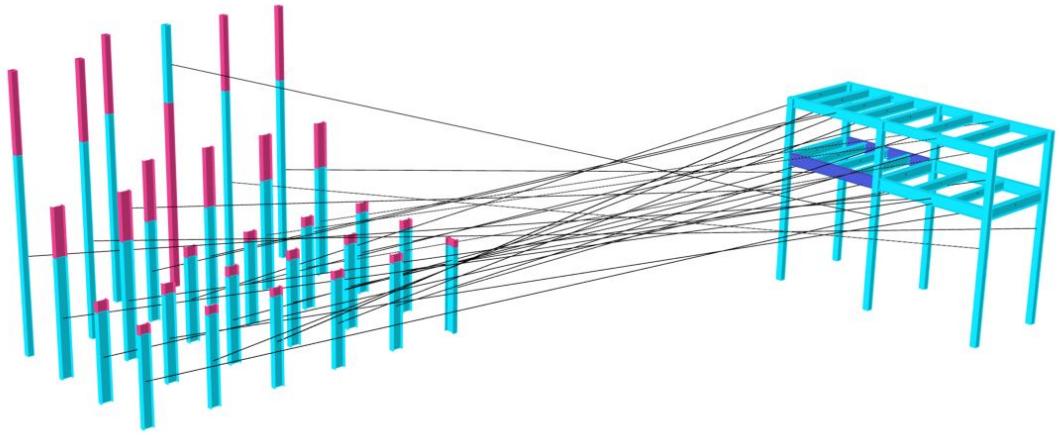


Figure 6.4: Visual mapping between all matches of supply and demand elements using the Visual Mapper.

Figure 6.5: The figures represent the matched elements in the demand structure with substituted and original cross section, respectively.

6.1.4 Case Study 1: FEM Analysis

In the Reuse Structure, it is evident that all substituted elements possess equal or greater cross sectional areas compared to the Original Structure. Additionally, all elements in the Reuse Structure have the same structural steel as before.

To ensure a connection between all elements before assembling the model in Karamba3D, artificial elements with spring cross sections are employed. The implementation of these is illustrated in Figure 6.6 and has been connected to a "Pipe" component to highlight their presence. Further, the nodes connecting artificial elements and beams are modelled as pinned, as explained in section Section 5.4.1. The model is subjected to two loads: the gravity load and an additional dead load of 20 kN/m, with a load factor of 1.35. This results in an additional dead load of 27 kN/m. This load is uniformly applied as a block load to all elements, except for the artificial elements. The load application is shown in Figure 6.7. Based on the mass retrieved from the "Assemble Model (Karamba3D)" component in Grasshopper, the Reuse Structure has a mass of approximately 1.13 times larger than the original structure.

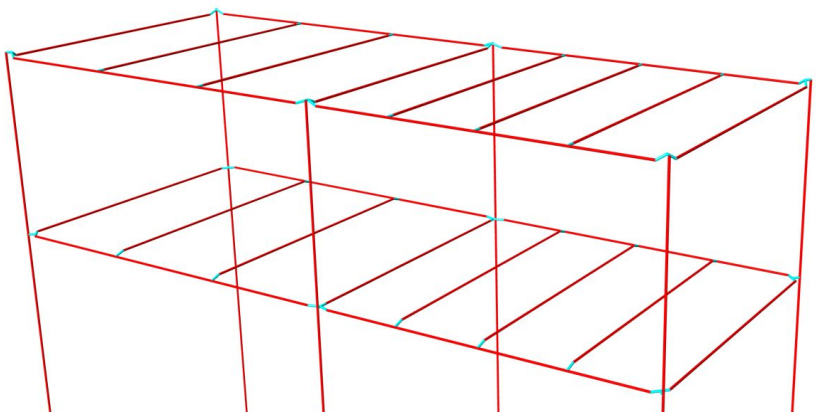


Figure 6.6: Artificial elements are implemented to ensure intersection between elements.

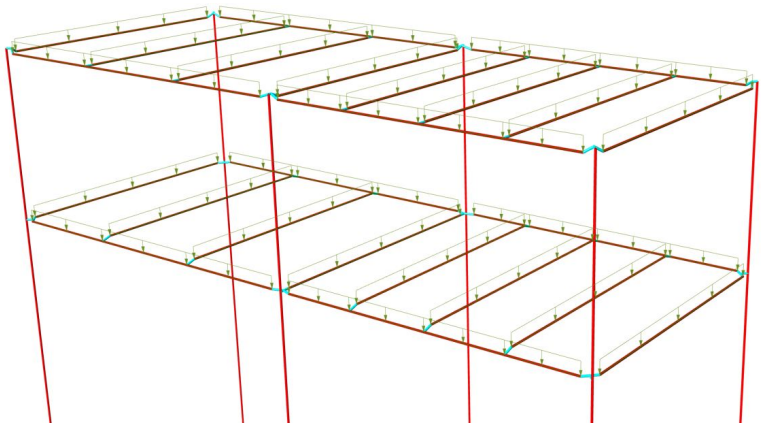
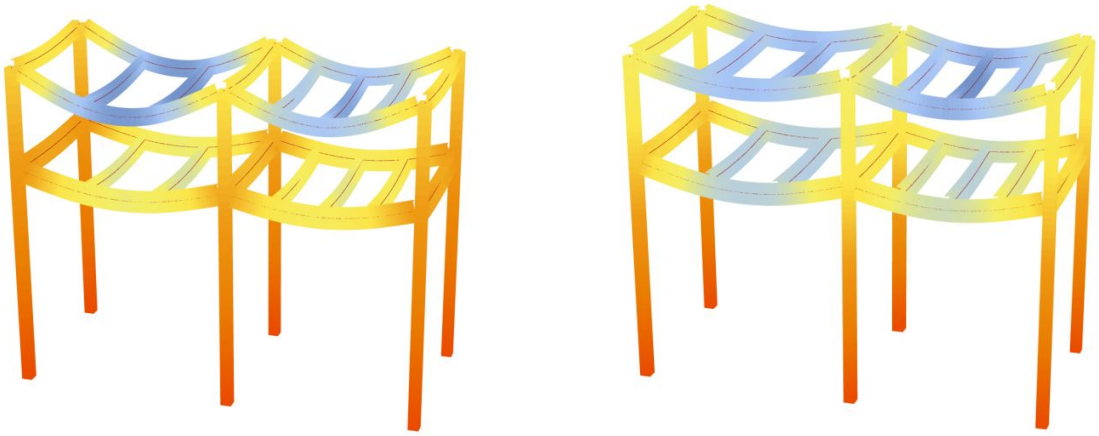


Figure 6.7: Load application for Case Study 1. The assigned load consists of the gravity load and an additional dead load of 27 kN/m, applied as a block load.

Results

The modification of cross sections presented several notable differences in the structural analysis. Firstly, the Reuse Structure exhibits enhanced stiffness, leading to reduced deformation, as seen in Figure 6.8. The deformation *Display Scale* set to 70, the *Upper Result Threshold* to 100 and *Lower Result Threshold* to 0. The utilisation from the "ModelView" component, returning the utilisation in terms of the ratio between the normal stresses and the yield stress for that specific material, is shown in Figure 6.9. As before, the *Upper Result Threshold* is set to 100 and the *Lower Result Threshold* is set to 0. Higher maximum bending moments are observed in the Reuse Structure, as seen in Figure 6.10 where the *Section Force* scale is set to 0.01. Table 6.2 presents the key findings of the analysis, highlighting the comparison between the Original Structure and the Reuse Structure. The values for utilisation are found using the "Utilization of Elements (Karamba3D)" component, which includes buckling according to Eurocode 3 [27].



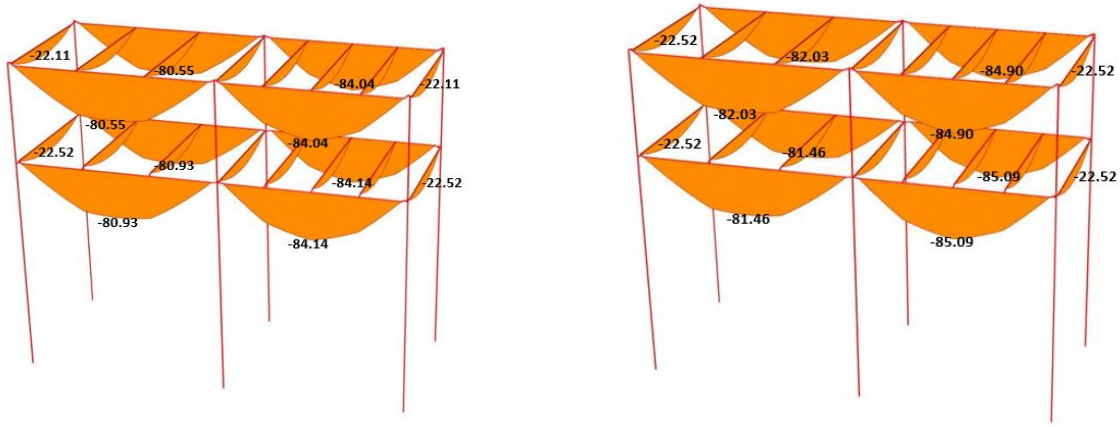
(a) Original Structure. Maximum displacement= 1.20 cm.

(b) Reuse Structure. Maximum displacement= 0.94 cm.

Figure 6.8: FEM Analysis results, Deformation. Deformation *Display Scale* set to 70.

(a) Original structure. (b) Reuse Structure.

Figure 6.9: FEM Analysis results, utilisation.



(a) Original structure. (b) Reuse Structure.

Figure 6.10: FEM Analysis results, Bending moments (My) in [kNm]. *Section Forces* scale set to 0.01.

Table 6.2: Key results obtained from the FEM Analysis of the Original and Reuse Structure.

	Max displacement [cm]	Max utilisation	Average utilisation	Max bending moment [kNm]	Mass [kg]
Original Structure	1.20	0.78	0.39	84.14	3729.779028
Reuse Structure	0.94	0.62	0.33	85.09	4219.764939

Supplementary analysis

Since the test files produced in Case Study 1 resulted in element substitutions with higher or equal cross sections than required, two additional analyses are performed: Test 1 and Test 2. The analysis aims to explore the potential outcomes of element substitutions, specifically focusing on instances where the cross sections are reduced. The intention behind performing these tests is to gain a more comprehensive understanding of the implications associated with element substitution and consequences for structural performance.

In Test 1 the cross sections for the Reuse Structure are substituted manually by assigning all elements a new cross section area between 80-100% of the Original Structure. The second analysis performed, Test 2, was

performed by giving all elements their original cross sections except for three elements. These were given a cross section between 80-100% of the Original Structure. The elements given a lower cross section area in Test 2, are highlighted in Figure 6.11. Figure 6.12, Figure 6.13 and Figure 6.14 illustrates the deformation, utilisation and moment distribution for Test 1 and Test 2. The settings for the display scales are as before. It is observed that with lower cross section areas, the deformation and utilisation decrease. Additionally, bending moments increase. Key results obtained from the FEM Analysis are shown in Table 6.3.

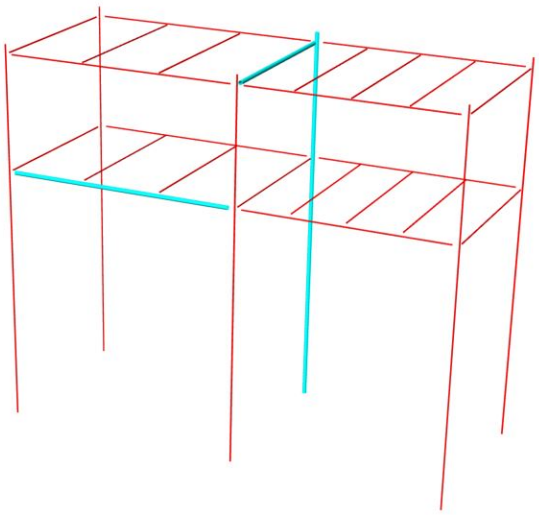
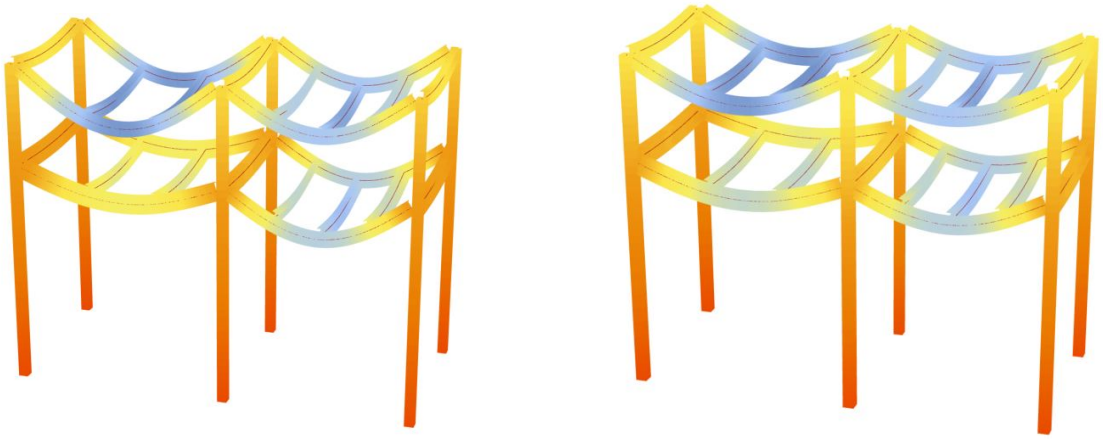


Figure 6.11: The highlighted elements have been given a new cross section area between 80-100% of the original cross section area in Test 2.



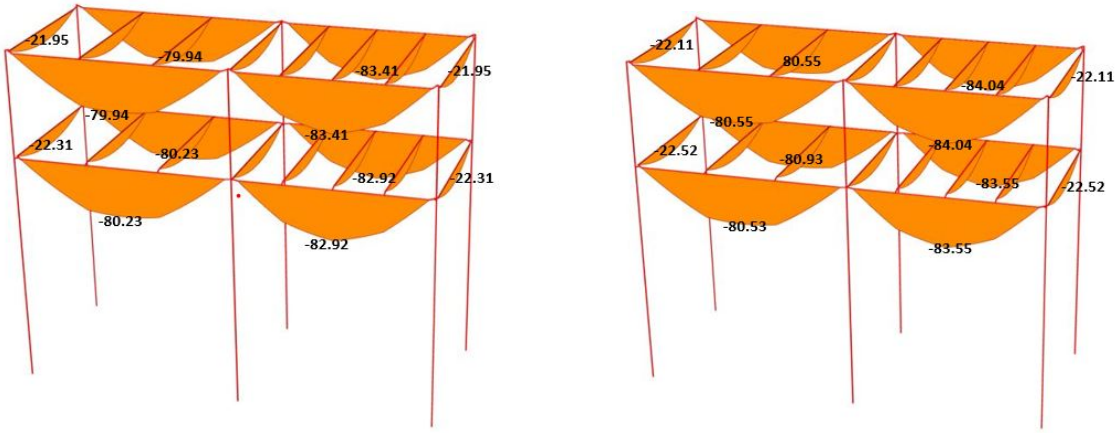
(a) Test 1. Maximum displacement= 1.66 cm.

(b) Test 2. Maximum displacement= 1.24 cm.

Figure 6.12: FEM Analysis results, deformation. Deformation *Display Scale* set to 70.

(a) Test 1. (b) Test 2.

Figure 6.13: FEM Analysis results, deformation. Deformation *Display Scale* set to 70.



(a) Test 1. (b) Test 2.

Figure 6.14: FEM Analysis results, Bending moments (My) in [kNm]. *Section Forces* scale set to 0.01.

Table 6.3: Key results obtained from the FEM Analysis of Test 1 and Test 2 in Case Study 1.

	Max displacement [cm]	Max utilisation	Average utilisation	Max bending moment [kNm]	Mass [kg]
Test 1	1.66	1.03	0.56	83.41	3150.09959
Test 2	1.24	0.78	0.44	84.04	3579.96058

In all scenarios, the applied dead load of 27 kN/m remains constant, while the gravity load varies according to the mass of the assembled elements. The findings reveal that an increase in mass resulting from larger cross section areas in the Reuse Structure leads to higher bending moments. Similarly, Test 1, characterised by smaller cross sections and thus lower mass, exhibits lower bending moments in the structure.

When decreasing the cross section areas in Test 1, the maximum and average utilisation increase. The maximum utilisation of some elements in the structure exceeds one, meaning the respective elements have exceeded their stress design capacity. The element substitution does therefore not fulfil structural requirements for this specific load case.

The effects of changing only three cross sections in Test 2 were minor for this case study. When some elements in the structure get a different stiffness, the distribution of forces within the structure will sub-

sequently change. However, given the relatively small changes in the element substitutions, this is not reflected in the results of the deflection, average and maximum utilisation, and bending moments.

6.1.5 Case Study 1: Financial and environmental impact

The steel density is given in Table 5.2 and has a value of 7850 kg/m^3. This is used in combination with the results presented in Table 6.1 to calculate the volume and weight of the Reuse Structure. The result is presented in Table 6.4.

Table 6.4: Weight and volume of the Reuse Structure consisting of reused and raw materials.

	Reused material	Raw material	**Total Reuse Structure**	**Original structure**
volume [m^3]	0.48248	0.03509	0.51757	0.461856
weight [kg]	3787.468	275.4565	4062.9245	3625.5696

The structures in the test files are located at the coordinates given in Table 6.5, which are extracted from the IFC file. The shortest distance by road between these locations is 521 km according to Google Maps [**Google.n.d.HgskoleringenMaps**]. Furthermore, it is assumed that the raw materials have a transportation length of 100 km from the fabrication site to the construction site, as stated in Section 5.5.Table 6.6 shows the results in terms of GWP and cost for the Reuse Structure, consisting of both raw and reused materials. A comparison between the Original and Reuse Structure in terms of total GWP and cost is shown in Table 6.7.

Table 6.5: Latitude and longitude coordinates of the demand and supply structure.

	Latitude	Longitude
Demand structure	59.6678390502778	10.7670049666667
Supply structure	63.4147377013889	10.4059600827778

Table 6.6: Use of raw and reusable material in Reuse Structure for Case Study 1.

	Raw materials	Reused materials
Production GWP [kg CO_2]	325.04	134.13
Transportation GWP [kg CO_2]	2.47	176.81
Production cost [NOK]	18 444.59	25 3760.36
Transportation cost [NOK]	110.18	7893.08
Valuation GWP [NOK]	229.25	217.65

Table 6.7: Financial and environmental impact of Case Study 1 for the Original and Reuse Structure.

	Original Structure	Reuse Structure
Production GWP [kg CO_2]	4278.17	459.17
Transportation GWP [kg CO_2]	32.49	179.27
Production cost [NOK]	242 913.16	272 215.94
Transportation cost [NOK]	1450.23	8003.26
Valuation GWP [NOK]	3017.46	446.91
Total GWP [kg CO_2]	4310.65	638.44
Total cost [NOK]	247 380.62	280 666.11

6.1.6 Case Study 1: Discussion

In Case Study 1, a match of 28 out of 30 demand elements was achieved. A total cut-waste of 0.13 m^3 was produced. The Reuse Structure was calculated to be approximately 30 000 NOK more expensive than a

structure made from only raw materials. The total GWP however is approximately 15% lower when using reusable materials. From the results obtained in the FEM Analysis before and after element substitution, the structural response of such a decision would not affect the structural integrity.

The method employed in the study exhibited highly efficient performance on the self produces IFC files. The method returned the desired characteristics and properties. Further examination of the location assignment should however be performed.

6.2 Case Study 2: Tennebekk and Buebygget

For the second case study, the IFC files for Tennebekk and Buebygget are utilised. Tennebekk is a bus hall project consisting of 1410 *IfcElements*, whereas 113 are assigned to the *IfcBeam* and *IfcColumn* entity. Buebygget is an office building of seven floors. The structure consists of 4600 *IfcElements*, whereas 2527 are assigned to the *IfcBeam* and *IfcColumn* entities. Both structures are located in Bergen. Tennebekk and Buebygget are representing respectively the demand and supply structure. Both the IFC files originate from Autodesk Revit 2023 (ENU) and are presented in IFC2X3 format. The structures are presented in Figure 6.15, where their relative size is illustrated.

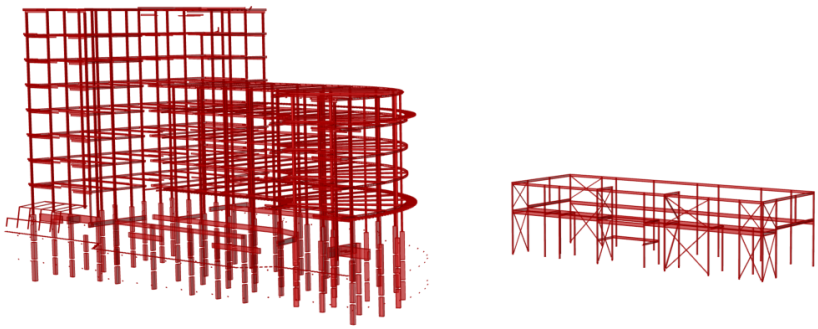


Figure 6.15: Structures in Case Study 2. Supply structure to the left, demand structure to the right.

6.2.1 Case Study 2: IFC Data Extraction

The IFC Data Extraction process for Case Study 2 brought forth various insights into working with IFC files and the Python scripts interpretation of these files. It provided valuable perspectives on the challenges and considerations associated with extracting data from IFC files within our specific context.

Cross section and material names

The extraction of cross sections from the IFC files was successful for both structures in Case Study 2. However, the property names extracted are not included in the "Cross Section Range" component in Karamba3D, therefore the DataFrame needed to be modified. In the cases where the cross section type is not found in the database, a substitute is implemented. Table 6.8 provides an overview of the substitutions made specifically for Tennebekk. Each individual cross section was carefully examined and assigned a substitute with comparable properties to accurately represent the behaviour of the element.

Table 6.8: Overview of the cross section substitutions manually performed for Tennebekk in order to comply with available cross sections in Karmaba3D.

Cross section IFC file	Cross section Karamba3D	Count
HEA100	HEA100	12
HEA220	HEA220	6
HEA260	HEA260	1
HSQ285/330	HEA300	2
HEA320	HEA320	1
IPE270	IPE270	6
IPE400	IPE400	25
HUP 200x120x10	RHS 200x120x10	2
HUP 150x150x10	SHS 150x150x10	2
SHS180x10	SHS 180x10	29
VL 200x200x16	SHS 200x200x12	2
HUP 80x80x5	SHS 80x80x5	24
HUP 200x150X10	RHS 200x150x10	1
	Total	113

Location

Buebygget and Tennebekk have the *IfcSite* entity as given in Listing 5 respectively. The entity is retrieved by opening the IFC file in Visual Studio Code.

```
#217= IFCSITE('0w2sGaZrz4fvFF21Cf8hu0', \#42, 'Default', \$, \$, \#216, \$, \$, .
    ELEMENT., (42,24,53,508911), (-71,-15,-2429,-58837), 19619.9999999999,\$,\$);

#5105= IFCSITE('05dEApX5f0MeApve83GT71', \#20, 'Surface:1559047', \$, \$, \#5104,
    \#5100, \$, .ELEMENT. (59,54,57,599999), (10,44,59,999999), 38600., \$,\$);
```

Listing 5: *IfcSite* of Case Study 2 collected from opening the IFC file in IfcOpenShell. The entities are given for the supply and demand structure respectively.

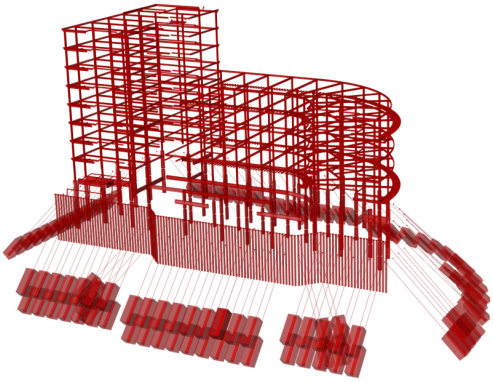
The resulting longitude and latitude coordinates for Buebygget are 42.414864 and -71.258072, coordinates to an address in the USA. The resulting longitude and latitude coordinates for Tennebekk are 59.916, 10.75, coordinates for an address in Oslo, Norway. Both of these coordinates are incorrect, as we know prior to the data extraction that both structures should return a location in Bergen, Norway. One potential reason for this deviation could be an incorrect assignment of the address for the *IfcSite* by the creator of the IFC file. Alternatively, it is possible that no address was assigned to the entity, resulting in it being given a default address instead.

The address retrieved from the *IfcPostalAddress* is Oslo, Norway for Tennebekk and Kronsparken, Bergen for Buebygget. Buebygget has the correct address, while Tennebekk is in reality located in Tennebekk, Bergen.

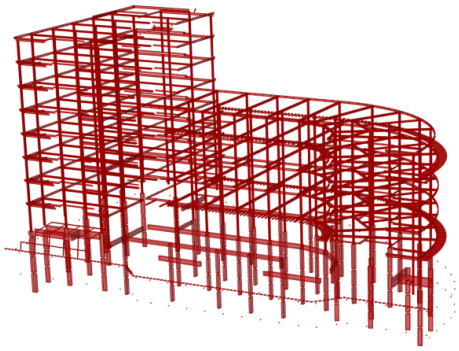
Extracting quantities

Noteworthy observations have been made regarding the quantities of non-typical beams and columns assigned to the entities *IfcBeam* and *IfcColumn* in Buebygget. Specifically, the observed lengths and volumes obtained from the IFC Data Extraction and the IFC geometry exhibited a lack of correspondence. To shed light on this issue, an example from Buebygget is provided.

It is observed that when importing IFC files into Grasshopper using the "ggIFC ReadFile" component, there is a significant difference in the geometry obtained from the display of breps and data. This is illustrated in Figure 6.16.



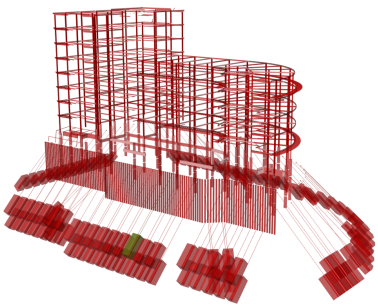
(a) Buebygget *IfcBeams* and *IfcColumns* data.



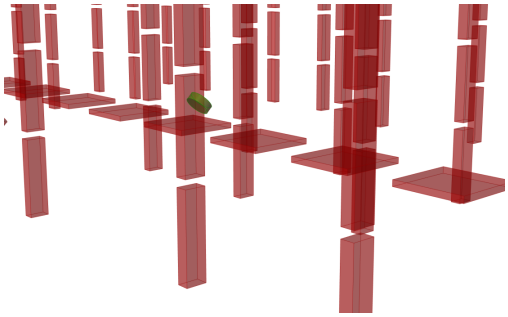
(b) Buebygget IfcBeams and IfcColumns breps.

Figure 6.16: Imported data and brep from Buebygget.

By looking at a specific element, the consequence of such a difference is observed. The element with GUID 0hSEg3wBj8c9CD2$Re070Q has the index 1562 in Grasshopper right after the importation of the file. The element is a ground anchor of steel. Figure 6.17a highlights the geometry from the data for index 1562. Figure 6.17b highlights a close-up of the brep for the same index.



(a) Buebygget data for index 1562.



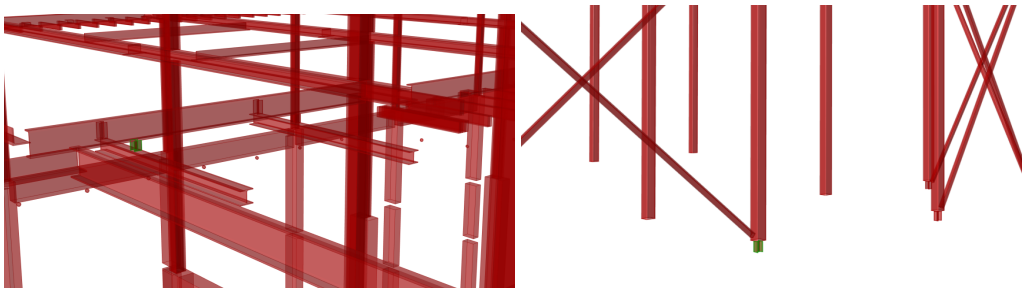
(b) Buebygget brep for index 1562.

Figure 6.17: Close up of IFC element 1562 viewed as data and brep.

The results obtained from the IFC Data Extraction for this given element is a length of approximately 15 m and a volume of 20 m^3. The lengths calculated in Grasshopper are based on the center lines of the breps and volume is found by using the "Volume" component where the breps are the input. The length obtained is 113 mm and the volume of approximately 0.002 m^3. It is therefore clear that the IFC Data Extraction calculates the quantities based on the data information and Grasshopper calculates the quantities based on the brep information. This was only observed for non-typical beams and columns assigned to *IfcBeams* and *IfcColumns*.

6.2.2 Case Study 2: Element mapper

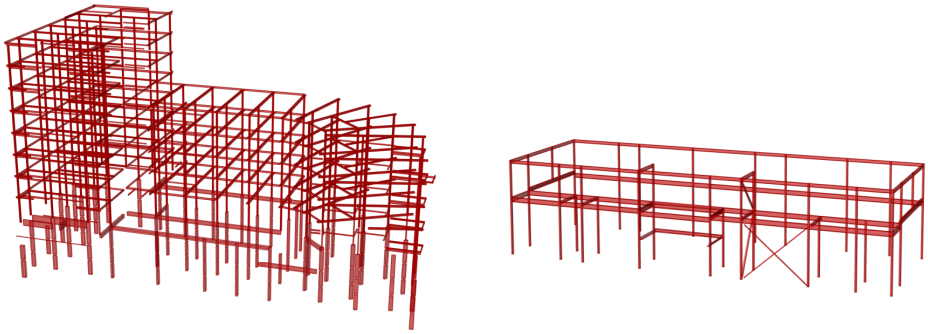
Prior to the mapping of demand and supply elements, some elements are filtered out. Elements of insignificant size are present in both Buebygget and Tennebekk represented as *IfcBeams* and *IfcColumns*. An example of an element with this quality is highlighted in Figure 6.18. The matching between elements of small volumes is not necessarily problematic for the Element Mapper, other than being an unrealistic approach for reusing elements in real practice.



(a) Small connecting element in Buebygget. (b) Small connecting element in Tennebekk.

Figure 6.18: Small connecting elements observed in Tennebekk and Buebygget.

The elements with a volume of under 0.005 m^3 were removed from both Buebygget and Tennebekk. This was done by utilising the brep obtained from the IFC files in Grasshopper. In addition, there are several curved elements in Buebygget. These are removed from the list of breps using the component "ggIfc Decompose Extruded Area Solid", as explained in Section 5.2.1. Figure 6.19 shows the updated collection of elements, excluding small and curved elements, to be examined further for mapping. However, the elements represented in the figures may not possess the material type steel, and if so, they are excluded in the Element Mapper.



(a) Relevant elements Buebygget. (b) Relevant elements Tennebekk.

Figure 6.19: Elements used in Element Mapper for Case Study 2.

There is a total of 101 in Tennebekk and 772 elements in Buebygget when looking at *IfcBeams* and *IfcColumns* that exclusively possess a volume greater than 0.005 m^3, exhibit straight characteristics, and have been assigned steel material. These are the elements used for further mapping.

Two element mappings are performed in Case Study 2. Tennebekk represents the demand structure in both mappings. The first mapping bases its results of the supply from the original Buebygget DataFrame, *Element Mapper with Buebygget cross sections*. For the second mapping, the cross sections in the supply DataFrame are substituted, making it represent a random supply bank, *Element Mapper with random supply bank*. The visual mapping of elements and the FEM Analysis is only performed for the random supply bank.

Case Study 2: Element Mapper with Buebygget cross sections

The Element Mapper provided a successful match for 97 out of 101 elements in the demand structure. The mapping process, therefore, produced approximately a 96% match of demand elements. The Reuse Structure utilised 89 elements from the supply bank, meaning eight supply elements fulfilled the requirements of 16 demand elements. As a result, the mapping process produces a total cut-waste of 1.31 m^3. Furthermore, the Reuse Structure obtained a volume of 7.88 m^3, which is double the volume of the Original Structure. Key results of the Element Mapper are shown in Table 6.9.

Table 6.9: Results of Element Mapper for Case Study 2 with Buebygget cross sections.

	Volume [m^3]
Total volume of demand elements	3.59
Total volume of supply elements	39.03
Volume of unmatched demand elements	0.59
Volume of used supply elements	8.61
Cut-waste volume	1.31
Utilised supply volume in Reuse Structure	7.29
Reuse Structure	**7.88**

Case Study 2: Element Mapper with random supply bank To address the challenge of a large number of elements in Buebygget and the unavailability of their cross section information in Karamba3D, a simplified approach was adopted to perform the element mapping. New cross sections were assigned to elements, not necessarily reflecting their original qualities in Buebygget. Some cross sections were given larger properties, while others were assigned smaller properties. Consequently, the resulting DataFrame for the supply elements does not represent the original DataFrame for Buebygget but rather resembles a random supply bank. The total volume of the random supply bank is now 31.45 m^3, while the total length of all elements remains the same.

Since the moment of inertia is originally calculated using breps, this property does not correspond to the property represented in the DataFrame anymore. Therefore, the moment of inertia is manually added to the DataFrame to ensure that the associated properties of an element are accurately reflected.

The total number of *IfcBeams* and *IfcColumns* in the demand and supply IFC files that fulfil the requirements to undergo matching is unchanged, meaning that there are 101 demand elements and 772 supply elements. The Element Mapper provided a successful match for 71 demand elements, utilising 71 supply elements, indicating that no supply element was used more than once. As a result, the mapping results in a total cut-waste of 0.43 m^3. Furthermore, the new structure obtained a volume of 5.27 m^3 which is approximately 1.5 times larger than for the Original Structure. Key results of the Element Mapper are presented in Table 6.10.

Table 6.10: Element Mapper results for Case Study 2 with random supply bank.

	Volume [m^3]
Total volume of demand elements	3.59
Total volume of supply elements	31.45
Volume of unmatched demand elements	1.51
Volume of total used supply elements	4.19
Cut-waste volume	0.43
Utilised volume of supply in Reuse Structure	3.76
Reuse Structure	**5.27**

6.2.3 Case Study 2: Visual Mapping

Case Study 2: Visual Mapping with Buebygget cross sections

An illustration of the demand structure and supply element is shown in in Figure 6.20, with the colour of the elements representing a match, no match and cut-waste as determined by the *Element Mapper with Buebygget cross sections*. Mapping between one specific demand and supply element is shown in figure Figure 6.21. Figure 6.22 illustrates the mapping between all matched demand and supply elements.

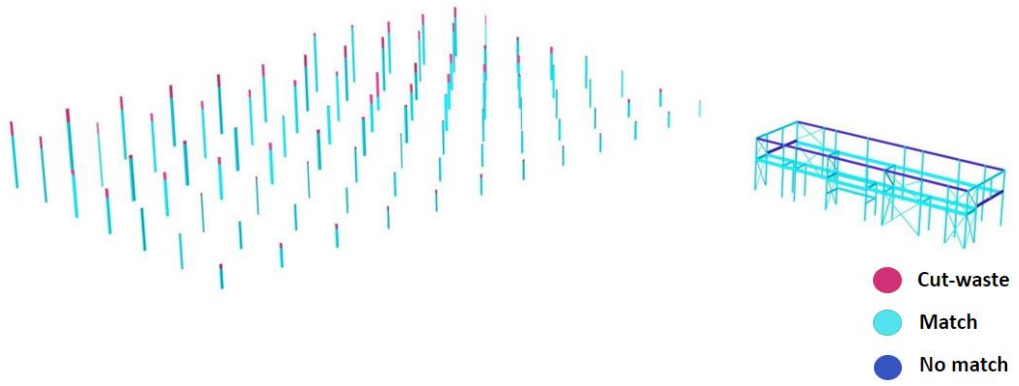


Figure 6.20: Supply elements and demand structure for Case Study 2 with *Buebygget cross sections*. Match, no match and cut-waste elements are highlighted. Relevant supply elements are visualised to the left.

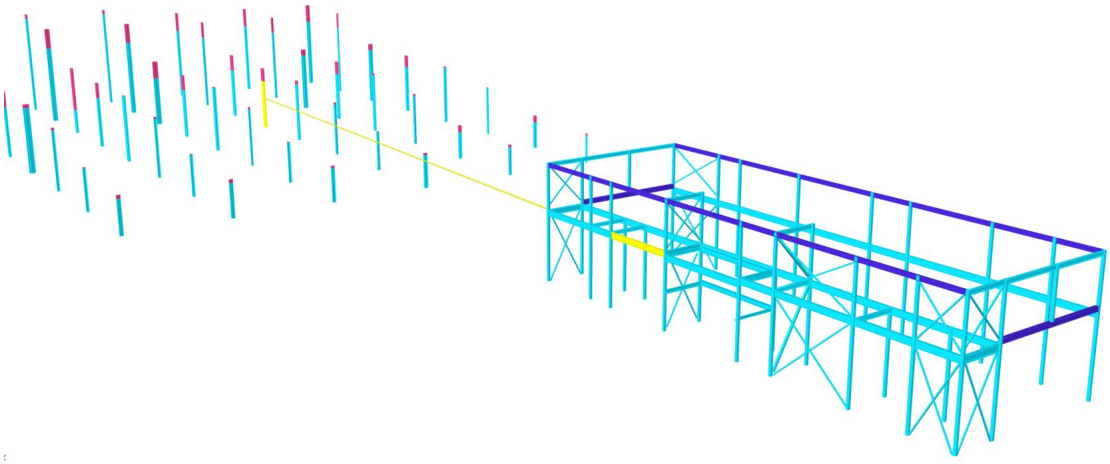


Figure 6.21: Visual mapping between one demand element and its respective supply element match for Case Study 2 with *Buebygget cross sections*.

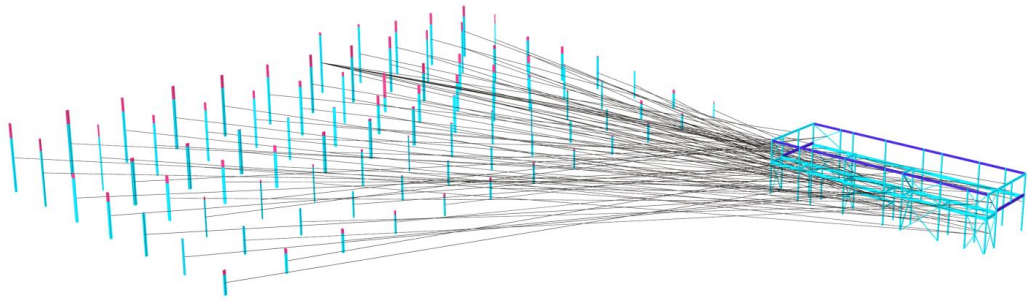


Figure 6.22: Visual mapping between all matches of demand and supply elements by for Case Study 2 with *Buebygget cross sections*.

Case Study 2: Visual Mapping with random supply bank

The new manually assigned cross sections and associated properties assigned in the DataFrame, may not align precisely with the actual cross sections of an element visual in Rhino. Nevertheless, this deviation does not pose any issues as the sole purpose of the Visual Mapping is to visually represent the outcome of the Element Mapper, without generating numerical results. The layout of the supply elements and demand structure is shown in Figure 6.23. Figure 6.24 illustrates the mapping between one matched demand element and its corresponding supply element, while Figure 6.25 illustrated all the matches. After a demand element has been matched with a supply element, the cross sections are substituted. This is illustrated in Figure 6.26. Figure 6.27 shows a close-up of the element substitution.

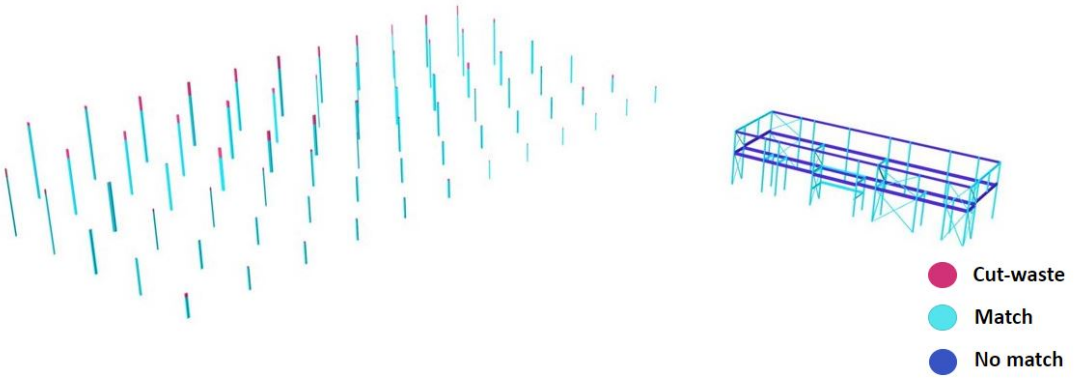


Figure 6.23: Supply elements and demand structure for Case Study 2 with *random supply bank*. Match, no match and cut-waste elements are highlighted. Relevant supply elements are presented to the left.

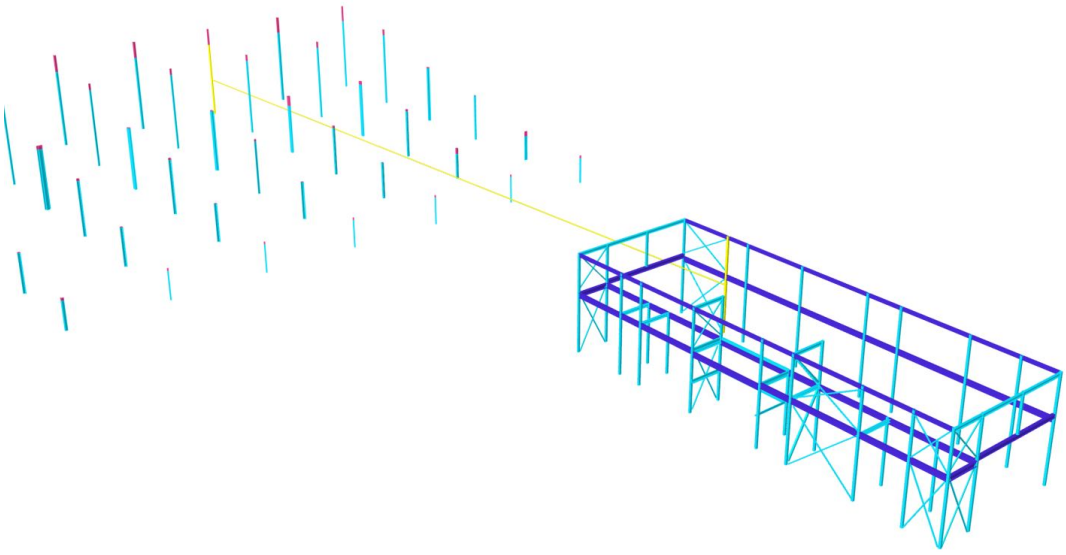


Figure 6.24: Visual mapping between one demand element and its respective supply element match for Case Study 2 with *random supply bank*.

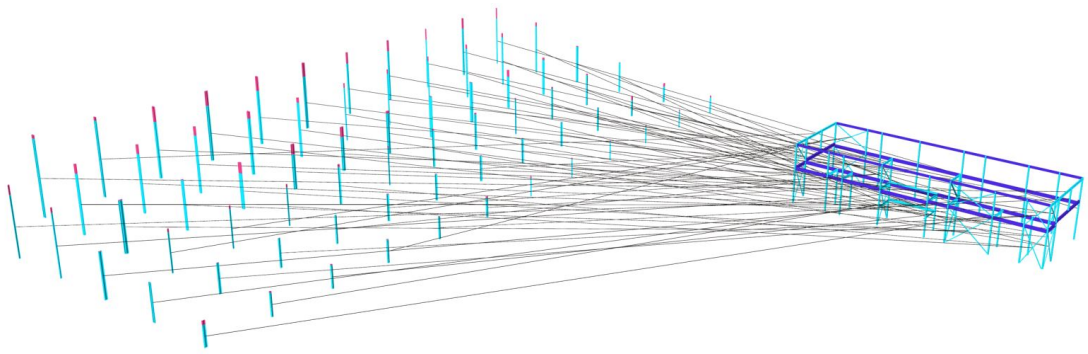


Figure 6.25: Visual mapping between all matches of demand and supply elements by using the Visual Mapper for Case Study 2 with *random supply bank*.

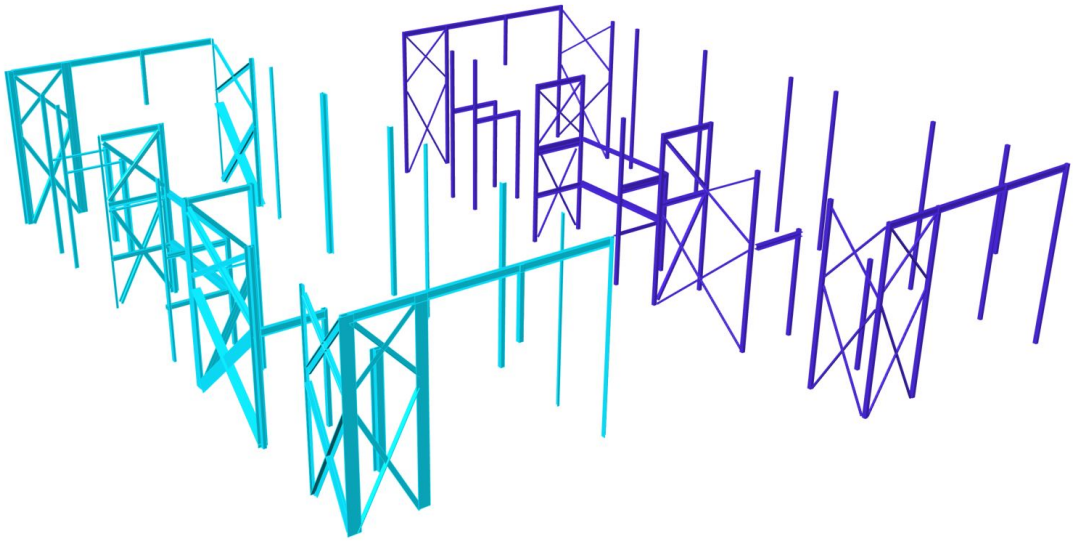


Figure 6.26: The figures represent the matched elements in the demand structure with substituted and original cross section, respectively, for Case Study 2 with *random supply bank*.

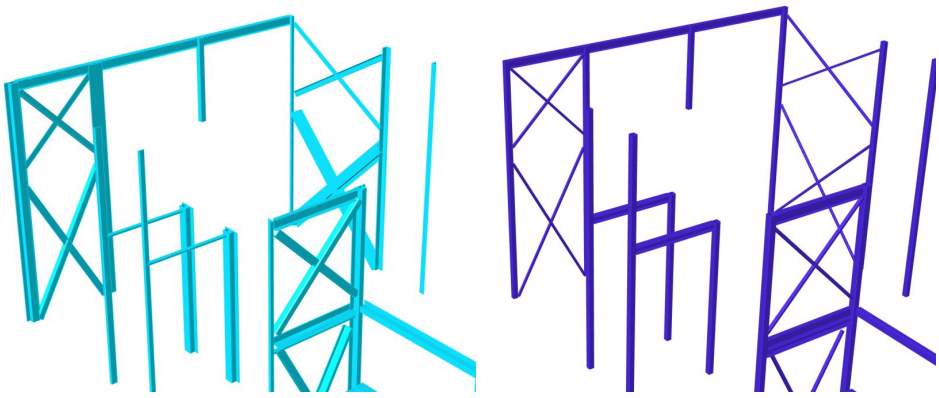


Figure 6.27: The figures represent a close-up for the matched elements in the demand structure with substituted and original cross section, respectively, for Case Study 2 with *random supply bank*.

6.2.4 Case Study 2: FEM Analysis

To perform the structural analysis on Tennebekk before and after the substitution of elements, a few modifications is done to the model. The six elements highlighted in Figure 6.28 are removed from the model as these were found to be non-relevant for the load-bearing of the structure. This is further explained in Section 6.3.

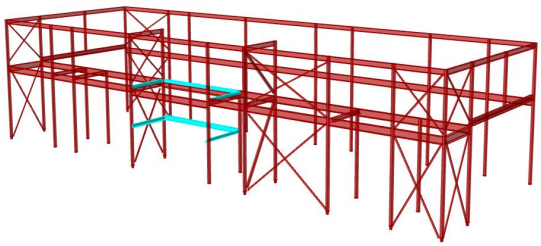
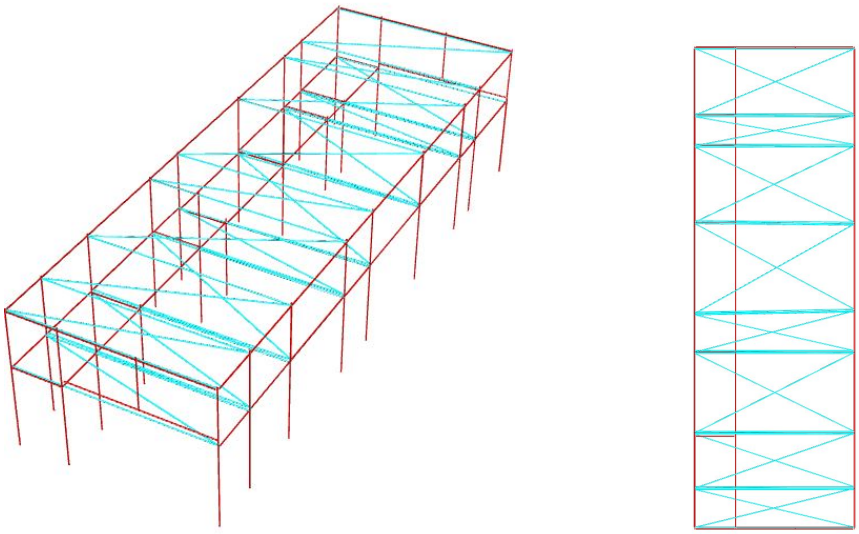


Figure 6.28: Removed elements from Tennebekk model.

The beams on the second floor in Tennebekk are interpreted as continuous when the IFC file is imported to Grasshopper. To ensure realistic connections between beams and columns, the beams are split into smaller segments. When this is done, all the elements are still represented under the same path, but with different indices. The same splitting procedure is performed on the columns. In the cases where center lines do not intersect, as described in Section 5.4.1, artificial elements are implemented. The model consists of a total of 95 elements after the splitting process, disregarding the artificial elements.

The Tennebekk structure consists of several hollow cores. These hollow cores will in reality distribute and transfer horizontal forces. As these are not a part of the FEM Analysis model, the distribution and transfer of such forces need to be considered. Elements with spring cross sections are implemented across the decks to simulate the function of hollow cores, resulting in better capture of the actual behaviour of the frame. This is accomplished by utilising a range of integrated components in Grasshopper, that locate the orientation of the center line axes of the beams and filter out those with the desired alignment. In this case, it refers to the beams situated at the horizontally longest sides of the structural frame. These new elements are connected to the intersection points between beams and columns. This can be seen in Figure 6.29.



Perspective view. Aerial view, *xy*-plane.

Figure 6.29: Spring elements distributing horizontal forces in Tennebekk.

The spring cross sections are created using the "Cross Section (Karamba3D)" component which allows for control of C_t and C_r, the translational and rotational stiffness respectively. The cross sections are restricted against rotation but allowed translational movement. The spring cross section has no weight, minimising the impact of the overall structural behaviour. The settings of the cross section are presented in Figure 6.30.

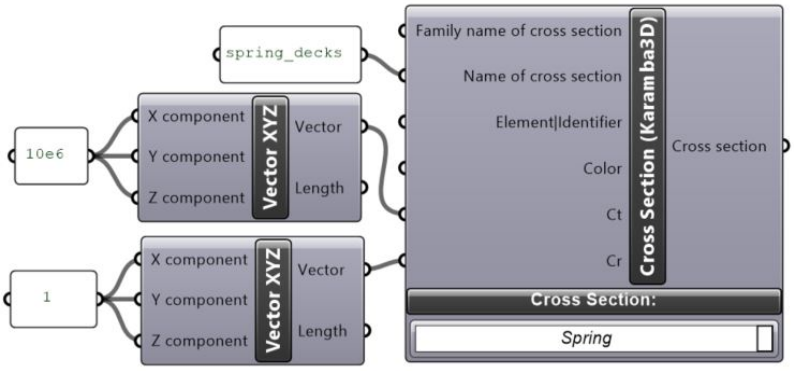


Figure 6.30: Spring cross section with high translational stiffness and low rotational stiffness.

Connections are modified by implementing beam joints at every relevant intersection point. This is done using the Grasshopper components presented in Figure 6.31, where all start and end points of beam elements are extracted. Duplicate points are removed before other non-relevant points are removed as well using "Set Difference". The output points are assigned beam joints. The joints are free to rotate about its y- and z-axis. Supports were designed as simply supported by restraining the translational movement, and rotation along the z-axis.

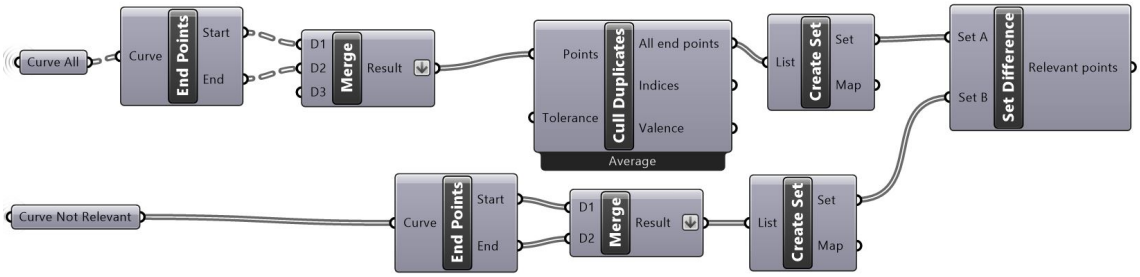


Figure 6.31: Locating points to locate beam joints.

The load cases and combination of loads used before and after substitution of cross sections, are presented in Table 6.11 and Table 6.12, respectively. Loads were applied using the "Loads (Karamba3D)" component. The settings applied were as followed: *Type of load*: MeshLoad Constant, *Generation*: Line loads, *Orientation*: Global. The load is applied as a mesh using *IfcSlab* as a basis. The *IfcSlab* is converted from *IfcExtrudedAreaSolid* to a brep. This is done using the "ggIFC ExtrudedAreaSolid" component. One brep is created for the second floor and one for the roof. These are converted to meshes using the component "Mesh Breps (Karamba3D)". To ensure proper distribution of weight and load application, it is essential to separate the elements oriented in the x-direction from the others. This separation allows the load to be correctly applied to these specific elements rather than affecting all the beams collectively. Consequently, beams are sorted in lists based on their orientation, whether it is in the x-direction or the y-direction.

Table 6.11: Load cases for Case Study 2.

Load cases	Value [kN/m^2]
Applied self-weight	1.0
Live load	4.0
Snow load	1.4
Self-weight (+struc. dead load)	5.4

Table 6.12: Combination of load cases for Case Study 2.

No.	Name	Type	Factor	Load cases
1	ULS	Ultimate	1.35	Applied self-weight
			1.050	Live load
			1.050	Snow load
			1.350	Self-weight (+struc. dead load)

The model was subsequently assembled as described in Section 5.4.1. The cross sections and steel types were extracted from the DataFrame and assigned to the correct "Line to Beam" component. The cross section and steel-type data are sorted in the same order as the line elements, ensuring a smooth and seamless assignment of these properties. An excerpt of the DataFrame with original and reuse cross sections is presented in Figure 6.32.

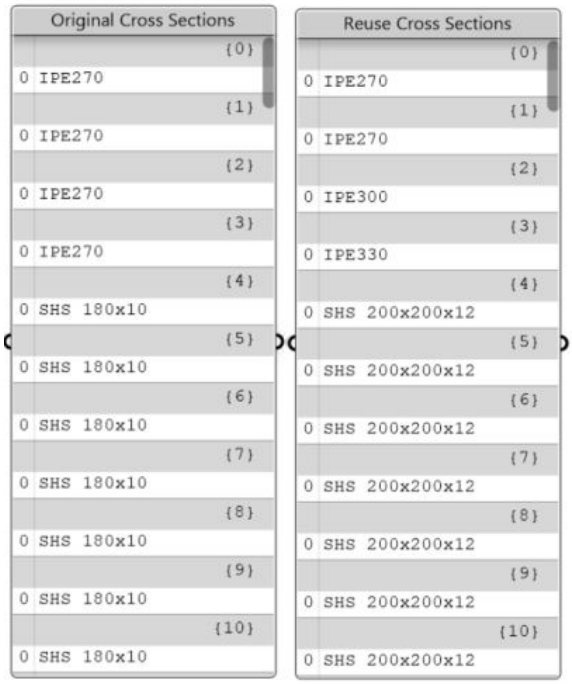


Figure 6.32: Part of the cross section DataFrame used for cross section assigning. Paths are represented with curly brackets.

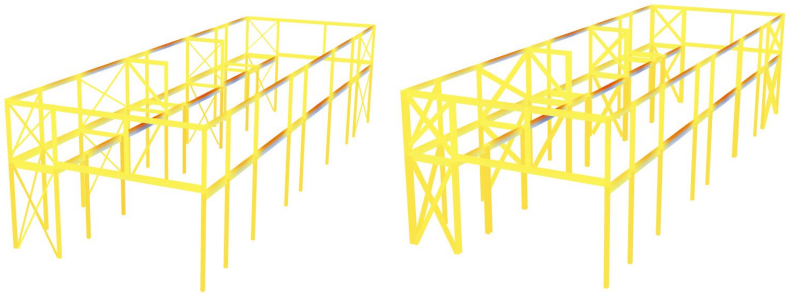
Results FEM Analysis

Key results from the structural analysis are presented in Table 6.13. The focus in this section will be on the effect the substituted elements have on the overall structural behaviour. Further evaluation of the structural integrity of the FEM model will be performed in Section 6.3.

Table 6.13: Key results obtained from the FEM Analysis of the Original and Reuse Structure.

	Max displacement [cm]	Max utilisation	Average utilisation	Max bending moment [kNm]	Max axial force [kN]	Mass [kg]
Original Structure	3.68	2.85	0.127	340	339	25931
Reuse Structure	3.65	1.84	0.091	340	334	38750

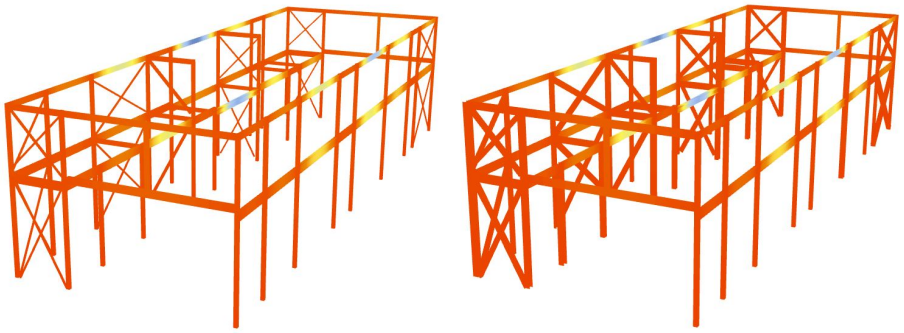
The structural analysis reveals decreased utilisation in the Reuse Structure. The utilisation is presented in Figure 6.33 in terms of the ratio between normal stresses and the yield stress for the specific material. *Upper Result Threshold* is set to 100 and the *Lower Result Threshold* is set to 0. The decrease in utilisation is expected as the Reuse Structure consists of elements with larger cross section areas and they will obtain lower stresses. It is observed that the utilisation is above one, indicating that the elements exceed their stress design capacity. This applies to several elements in both the analysis before and after substituting elements. The reason for this can be pointed to the design of the FEM Analysis model, which will be discussed further in the structural verification of the FEM Analysis in Section 6.3.



Original structure. Reuse Structure.

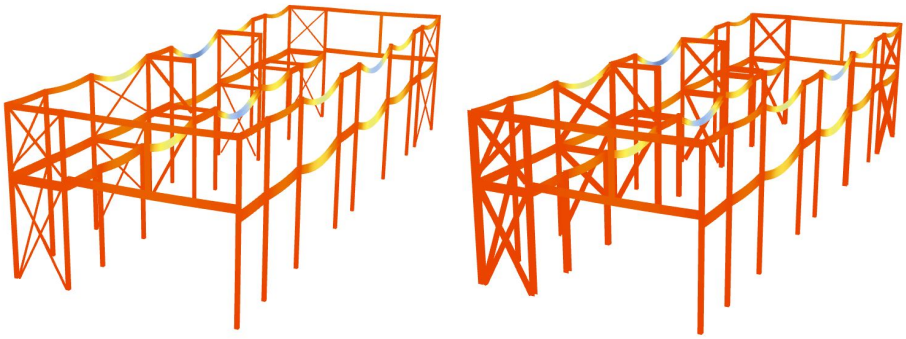
Figure 6.33: FEM Analysis result, utilisation.

The maximum displacement is greatest for the Original Structure. This is not as expected as steel behaves in a linear elastic manner, indicating an expected increase of deformation with the increase of load due to increased weight. However, the difference is insignificant and does not affect the structural integrity noteworthy.



Original structure, max = 3.68 cm, deformation *Display Scale* set to 0.

Reuse Structure, max = 3.65 cm, deformation *Display Scale* set to 0.

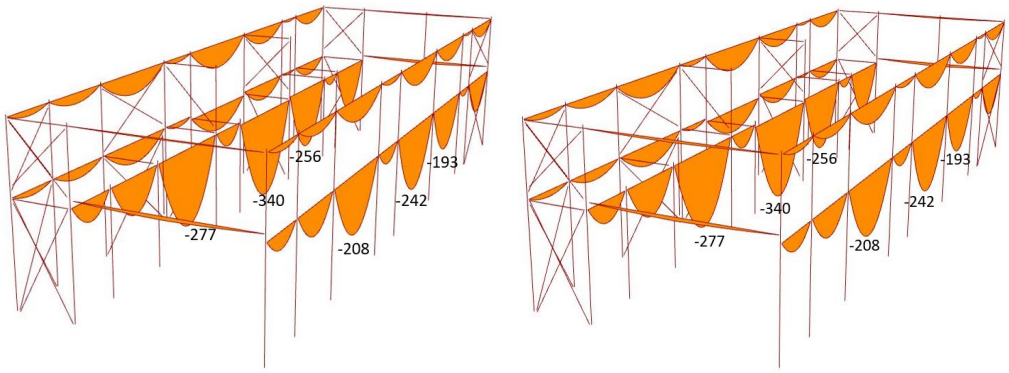


Original structure, deformation *Display Scale* set to 36.

Reuse Structure, deformation *Display Scale* set to 36.

Figure 6.34: FEM Analysis result, deformation.

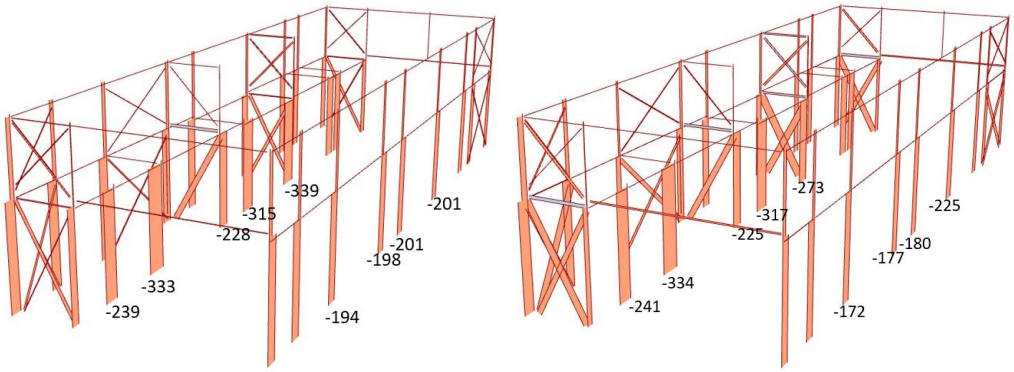
Furthermore, the Reuse Structure exhibits similar bending moments as the Original Structure, as shown in Figure 6.35. An increase in bending moments would not have been unrealistic as the weight of the structure increased.



Original structure, max = 340. Reuse Structure, max = 340.

Figure 6.35: FEM Analysis results, bending moments in [kNm]. Section Forces scale set to 0.016.

It is observed that maximum axial forces are lower in the Reuse Structure than in the Original structure. Most columns in the Reuse Structure have greater cross section area and moment of inertia than in the Original structure. This contributes to the distribution of axial forces over a greater area leading to a decrease in axial stresses. This is however not the case for all columns. In the areas where there are no matched elements and the original cross sections are used, there is minimum impact on the axial force distribution. The axial force distribution is presented in Figure 6.36.



Original structure, max = 339. Reuse Structure, max = 334.

Figure 6.36: FEM Analysis results, axial forces in [kN]. Section Forces scale set to 0.43.

6.2.5 Case Study 2: Financial and environmental impact

The financial and environmental impact is calculated for the matches obtained from the *Element Mapper with random supply bank*. The steel density, from Table 5.2, and results from Table 6.10, are used to calculate the volume and weight of the Reuse Structure. This is presented in Table 6.14.

Table 6.14: Weight and volume of the Reuse Structure consisting of reused and raw materials for Study Case 2 with *random supply bank*.

	Raw material	Reused material	**Total Reuse Structure**	**Original structure**
volume [m^3]	1.5066	3.327118	5.265094	3.589327
weight [kg]	11 826.81	26 117.8763	41 330.9879	28 176.21695

Tennebekk and Buebygget are both located in Bergen in Norway. As mentioned in section Section 6.2.1,

the coordinates in their corresponding IFC file imply otherwise. Therefore, the transportation distance from the demolition site to the construction site is added manually. The shortest distance by road between these locations is approximately 9 km according to Google Maps [18].

Table 6.15: Use of raw and reusable material in Reuse Structure for Case Study 2 with *random supply bank*.

	Raw materials	Reused materials
Production GWP [kg CO_2]	13 955.6358	924.938804
Transportation GWP [kg CO_2]	105.9682	21.06
Production cost [NOK]	792 396.27	1 749 897.712
Transportation cost [NOK]	4730.724	940.24
Valuation GWP [NOK]	9843.1228	662.199

Table 6.16: Financial and environmental impact from Case Study 2 with *random supply bank*.

	Original Structure	Reuse Structure
Production GWP [kg CO_2]	33 247.6859	14 880.5746
Transportation GWP [kg CO_2]	254.458	127.0782
Production cost [NOK]	1 887 806.536	2 542 293.982
Transportation cost [NOK]	11 270.486	5670.964
Valuation GWP [NOK]	23 451.5	10 505.32
Total GWP [kg CO_2]	33 502.1439	15 007.6528
Total cost [NOK]	1 922 528.522	2 558 470.271

6.2.6 Case study 2: Discussion

Observations from Case Study 2 prompted valuable insights into handling IFC files from the construction industry for reuse projects. These findings have contributed to a deeper understanding of the challenges and considerations associated with utilising IFC data in real reuse projects.

The IFC Data Extraction proved to be effective for Case Study 2, successfully extracting the desired quantities and properties, with the exception of the location information. However, during the analysis of the IFC files a specific noteworthy observation was made. This was the assignment of non-typical beams and columns to the *IfcBeams* and *IfcColumns* entities. As a consequence, the quantities obtained in the IFC Data Extraction did not correspond to the values obtained in Grasshopper. This issue was resolved by filtering out the small elements in Grasshopper based on their volume. Although these excluded elements had large volumes in the DataFrame, they possessed small volumes in Grasshopper, leading to their exclusion. Consequently, this issue did not cause wrongful mapping of elements. However, it raises the question of potential inconsistencies that may exist in other IFC files.

Several of the utilised cross sections in Buebygget were unavailable in the "Cross Section Range" database in Karamba3D. Random cross sections were assigned to these elements, transforming the supply bank of Buebygget to a random supply bank. The objective of the case studies is to compare two structures rather than precisely matching real structures, as Tennebekk and Buebygget. It is therefore justifiable to edit the cross sections for the most efficient testing of the method. In the future development of the methods presented in this thesis, it is essential to utilise the correct cross section areas. This can be achieved by creating custom cross sections that accurately represent the properties of the elements and incorporating them into the Karamba3D software. By employing these custom cross sections, it ensures precise and reliable results that align with the characteristics of real structures.

As indicated in Table 6.9, the Reuse Structure achieved using Buebygget cross sections exhibits a total volume that is more than twice that of the Original Structure. Likewise, the Reuse Structure employing the random supply bank shows a volume 1.5 times that of the Original Structure, as demonstrated in Table 6.10.

Upon analysing the elements in Tennebekk, it was noted that several brace elements were replaced with elements possessing significantly larger cross sections. Since Buebygget did not have brace elements similar to those in the Original Structure, there were limited alternatives available for substitution. Due to the absence of an upper limit requirement for cross section area in the script, there is no mechanism preventing or addressing this issue. However, if desired, an upper limit can be incorporated into the Element Mapper.

The total GWP linked to the Original Structure was more than twice the amount compared with the Reuse Structure, suggesting a significantly more sustainable solution. However, in terms of cost, the Reuse Structure was approximately 635 000 NOK more expensive. This cost difference is caused by the increased weight of the Reuse Structure. The structure had approximately 1.5 times larger volume compared to the Original Structure, leading to an increase in self-weight of around 13 tons.

As stated in Section 5.5, it is important to acknowledge that the method used to calculate the financial and environmental impact is a simplified approach. Amongst others, specific simplifications are made to transportation distances. In a practical reuse scenario, considerations may encompass not only the transportation of materials from the demolition site to the construction site, but also transportation to and from the fabrication site and, potentially, to and from an intermediate storage facility. Nevertheless, despite these simplifications, the obtained results still provide valuable insights into the potential consequences of the reuse process.

The FEM Analysis provided valuable insight into the structural behaviour of the Reuse Structure compared with the Original Structure. The structural integrity was not significantly affected by the substitution of elements. However, a few anomalies were observed, which can be attributed to the design of the FEM Analysis model rather than the substitution of elements.

6.3 Verification of FEM Analysis

The following section presents a comparison between the model created using IFC in Grasshopper as the base, and a conventionally developed model using other FEM design tools. The comparison is based on results from the FEM Analysis on Tennebekk and a structural analysis report provided by COWI. This report will further be referred to as the Calculation Report. The primary objective is to highlight the significant differences and potential strengths and weaknesses of modelling with IFC as the base in Grasshopper. The significance of this component lies in its ability to facilitate an analysis based on geometry extracted from IFC files in a more efficient manner, compared to constructing an analysis model from the ground up.

6.3.1 Modelling

The Calculation Report presents an analysis conducted on a model with modified cross sections. Hence, the same modifications are made to the FEM Analysis model to ensure the utmost similarity between the two. The model is modified to have the cross section types as presented in Table 6.17. All cross section types used in the Calculation Report is found in the "Cross Section Range Selector" component in Grasshopper, except for the custom-made "DR 26-290" cross section. The respective elements were assigned the cross section type HEA300 instead. All elements in the Calculation Report have the material type steel S355. This is not the case in the IFC file as 11 elements are assigned the steel type S235. This is subsequently changed for the modified model as well.

Table 6.17: Substituted cross sections in modified model.

Cross section Calculation Report	Cross section FEM Analysis	Count
KKR 180x180x10	SHS 180x10	27
KKR 80x80x5	SHS 80x5	26
DR 26-290	HEA300	2
IPE400	IPE400	25
IPE270	IPE270	22
HEA220	HEA220	6
KKR 150x150x10	SHS 150x10	2
	TOTAL	110

According to the Calculation Report, the columns in the structure are modelled as continuous beams from bottom to top, and are simply supported. The beams have two nodes and are connected to the columns with pinned connections. The bolts between the hollow cores and the beams on the second floor, as well as between the roof and the beams on the roof level, serve as bracing elements. These bolts transfer the bending moments as shear forces in the compression zone. The diagonal members are modelled as brace elements and are simply supported. The node between two diagonals is modelled as clamped and functions as a beam element. To ensure that the brace elements only experience tension stresses, they are assigned defined plastic capacities. For compression, the plastic capacity is set to 0, while it is infinite for tension. This enables the brace elements to behave in a plastic manner if subjected to compression. The software used for analysis employs non-linear calculations to determine if a load should be transferred to the diagonals or not. The roof is modelled using a "Cover" element, and vertical weight is applied on top of this cover. To distribute horizontal forces to the diagonals, a "Diaphragm-Rigid Membrane" is implemented. This function does not have any weight or vertical stiffness but possesses infinite horizontal stiffness, ensuring the forces are distributed based on the relative stiffness of the structure [13].

The FEM Analysis incorporates the loads as specified in the Calculation Report. The load case used in the conducted analysis is presented in Table 6.18. The load combinations applied in the Ultimate Limit State (ULS) are given in Table 6.19. The load is applied through the use of meshes in Grasshopper, similarly to what was done in the previous analysis in Case Study 2, Section 6.2.4. Additionally, a mesh is created for the wind load in order for the load to be most accurately applied. The wind load mesh is created manually using the "4Point Surface" component. The basis for this is the top and bottom points in the two outer columns on the side of the structure where the wind load is simulated. These points are extracted and provided as input data for the component. Subsequently, the surface is converted to mesh using "Mesh Breps (Karamba3D)".

Table 6.18: Load case in the Verification of FEM Analysis of Tennebekk.

Load cases	Value [kN/m^2]
Applied self-weight	1.0
Live load	4.0
Snow load	1.4
Wind	2.0
Self-weight (+struc. dead load)	5.4

Table 6.19: Load combinations in the Verification of FEM Analysis of Tennebekk.

No.	Name	Type	Factor	Load cases
1	ULS, wind	Ultimate	0.900	Applied self-weight
			0.000	Live load
			0.000	Snow load
			1.500	Wind load
			0.900	Self-weight (+struc. dead load)
2	ULS, no wind	Ultimate	1.35	Applied self-weight
			1.050	Live load
			1.050	Snow load
			1.350	Self-weight (+struc. dead load)

To properly assign loads to the correct elements, the beams and columns are sorted accordingly into *Columns Wind*, *Column No Wind*, *Beams First Storey*, *Beams Second Storey*, *Bracing Compression* and *Bracing Tension*. In addition, beams are sorted based on their orientation as described in Section 6.2.4. This is done by accessing the correct paths belonging to each element. This method of locating elements is possible due to the Data Tree structure that is implemented from the beginning of the script.

In the Calculation Report the loads was applied on all elements except for one are, this is illustrated in Figure 6.37. This was not taken into account in the FEM Verification model, hence potentially leading to somewhat different results.

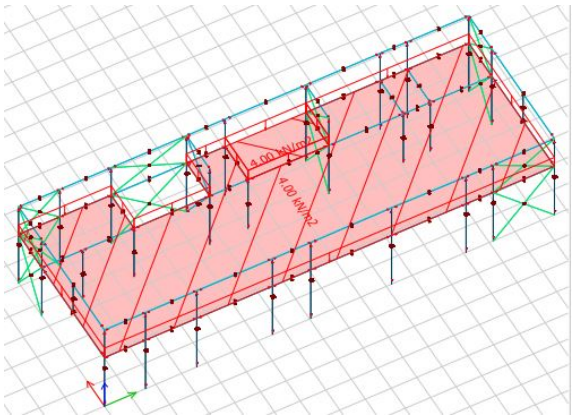


Figure 6.37: Load application on second floor in the Calculation Report [13].

6.3.2 Results

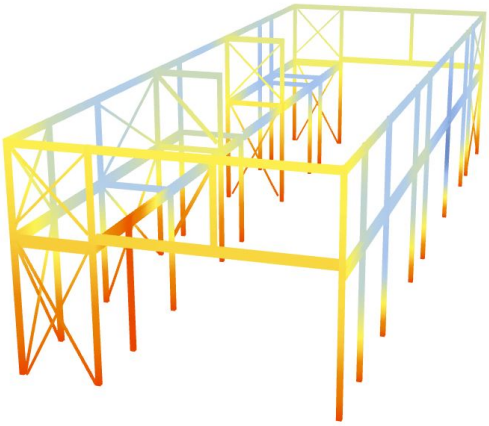
Key results from the FEM Analysis of modified Tennebekk are presented in Table 6.20. Load combination number one is further referred to as LC1, and load combination number 2 is referred to as LC2.

Table 6.20: Key results obtained from the Verification of FEM Analysis of Tennebekk for LC1 and LC2.

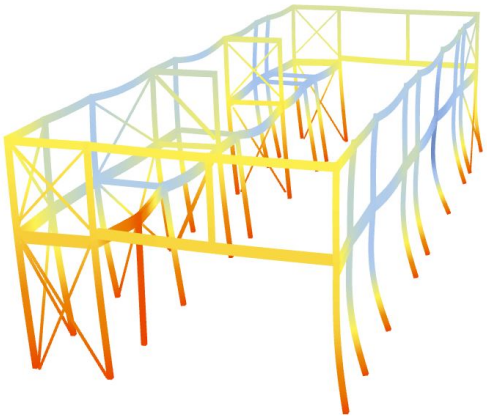
	Max displacement [cm]	Max utilisation	Average utilisation	Max bending moment [kNm]	Max axial force [kN]	Max torsional moment [kNm]
LC1	3.3	2.98	0.079	181	240	45
LC2	4.6	2.73	0.093	424	382	-

The deformation of LC1 is presented in Figure 6.38. The maximum displacement in LC1 can be observed in the column on the right side of the figure, with translation in the horizontal direction. In the structures to the left, the deformation is shown only in terms of colour display, while in the structure to the right, the

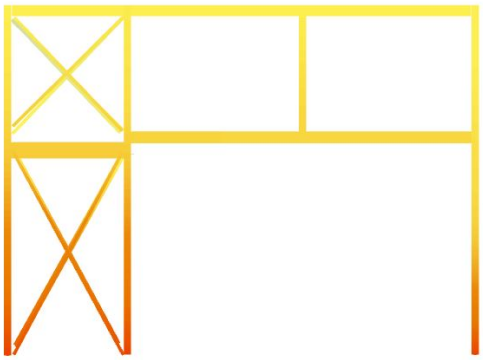
structure is visually deformed. The deformation *Display Scale* is set to 36, *Upper Result Threshold* is set to 100 and *Lower Result Threshold* is set to 0.



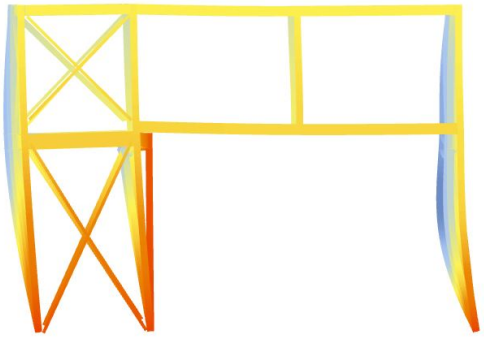
Deformation *Display Scale* set to zero.



Deformation *Display Scale* set to 36.



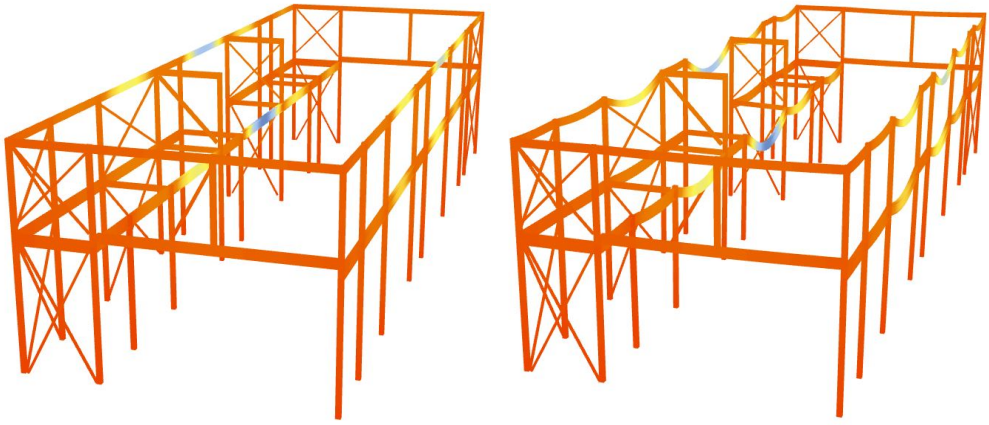
Deformation *Display Scale* set to zero.
Seen in the yz-plane.



Deformation *Display Scale* set to 36.
Seen in the yz-plane.

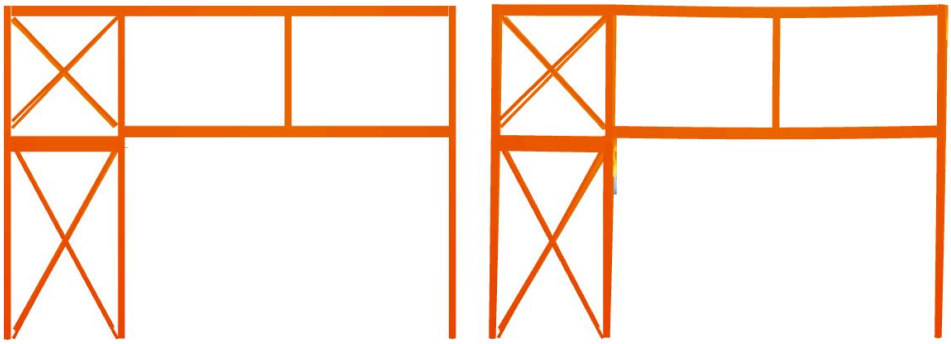
Figure 6.38: Deformation in LC1 in the Verification of FEM Analysis of Tennebekk. *Upper Result Threshold* is set to 100 and *Lower Result Threshold* is set to 0.

The deformation when LC2 is applied is presented in Figure 6.39. A particularly large displacement is observed on one of the beams located in the middle of the structure. This was not observed in the Calculation Report. However, this particular element does not intend to be a load-bearing element, even though it is a part of the analysis. In the Calculation Report, this element was not applied load, while in this case study the load was applied on all beam elements.



Deformation *Display Scale* set to zero.

Deformation *Display Scale* set to 36.



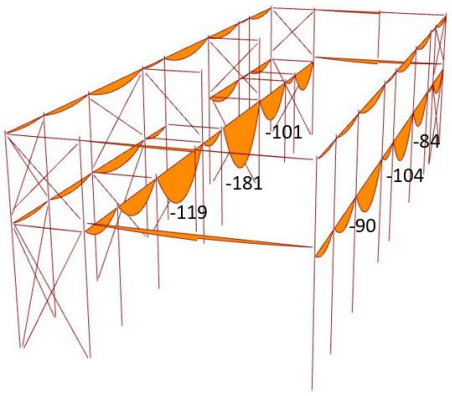
Deformation *Display Scale* set to zero.
Seen in the yz-plane.

Deformation *Display Scale* set to 36.
Seen in the yz-plane.

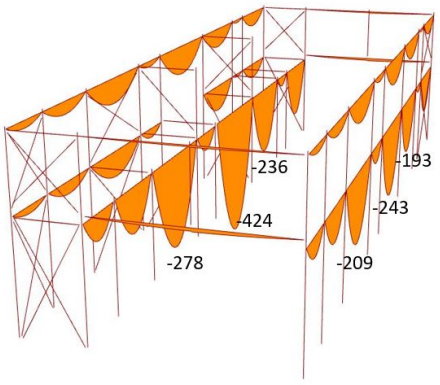
Figure 6.39: Deformation in LC2 in the Verification of FEM Analysis of Tennebekk. *Upper Result Threshold* is set to 100 and *Lower Result Threshold is set to 0*.

Moment about y- and z-axis, in addition to axial force distribution is presented in Figure 6.40 for LC1 and LC2. The greatest moment occurred when LC2 was applied. The maximum moment is found in the mid-span of the beam located in the middle of the structure, M_{max} = 424 kNm.

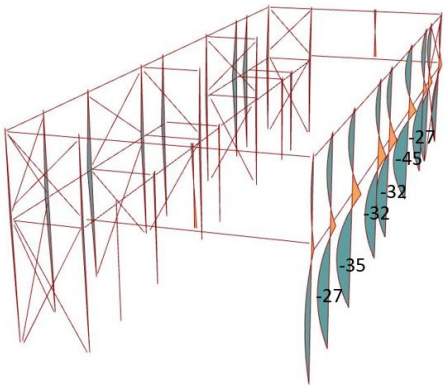
Moment diagrams, axial forces and displacement of the model presented in the Calculation Report is presented in Appendix C for comparison.



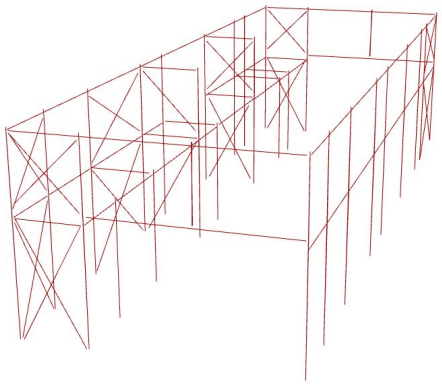
M_y LC1, scale=0.016



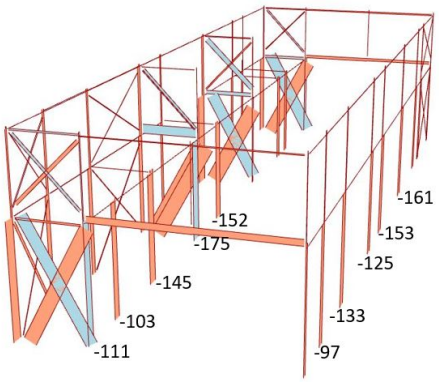
M_y LC2, scale=0.016



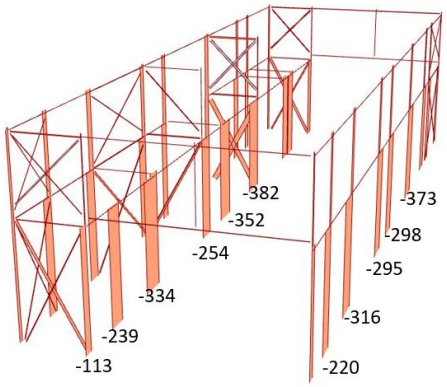
M_z LC1, scale=0.016



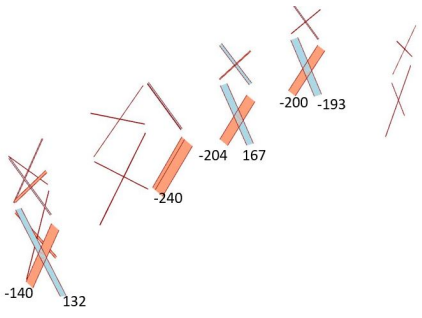
M_z LC2, scale=0.016



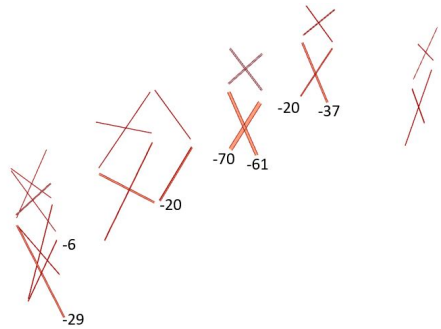
N_x LC1, scale=0.40



N_x LC2, scale=0.40



N_x LC1, scale=0.40



N_x LC2, scale=0.40

Figure 6.40: Results from the Verification of FEM Analysis of Tennebekk.

6.3.3 Discussion

The main differences between the FEM model in Karamba3D and in the Calculation Report is how the intersections, diagonal members, and hollow cores are modelled. In addition the on the second floor in is applied slightly different in the Calculation Report and in the Karamba3D model.

The Calculation Report designed the hollow cores and roof by using "Cover" and "Diaghragm- Rigid Membrane". This function was in Karamba3D modelled with the implementation of horizontal elements with spring cross sections. The elements with spring cross sections introduced stiffness and allowed for the horizontal forces to be effectively transmitted from the columns to the beams in the structural frame. The diagonal horizontal members provided additional stability by distributing the forces to areas with brace elements. These serve the same purpose as the diagonals in a truss system, thereby providing additional restraint, see principle in Figure 6.41. By incorporating elements with spring cross sections, the structural behaviour was better captured, as the horizontal force transmission was accounted for, without explicitly modelling the hollow cores. It provides a practical approximation of the behaviour of the structure under wind load, allowing for a more comprehensive analysis and evaluation of its performance. Even though the maximum horizontal displacement in the verification analysis was greater than for the Calculation report, the method served well as an alternative.

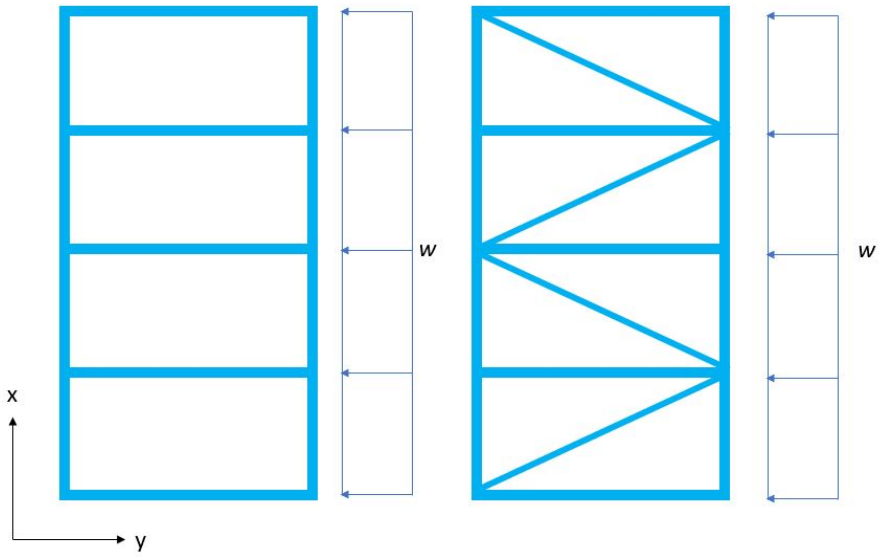


Figure 6.41: Truss system principle.

Using spring connections or elements with spring cross sections in Karamba3D allows for great control of flexibility and movement at the connection point of the elements. The stiffness of the spring determines the level of restraint and the amount of displacement allowed. Parametrically controlling this value allows for easy adaption and design of connections. This was sufficiently utilised to develop the artificial elements. Rigid links were modelled as springs with high stiffness. The artificial members with their assigned infinite rotational and translational stiffness allow for moments and forces to be sufficiently transferred, without contributing to any additional effect on the structural behaviour. The implementation of artificial elements as rigid links sufficiently connected elements from the IFC file that did not intersect when imported to Grasshopper. The effect of the artificial elements is illustrated in Figure 6.42.

Bending moment distribution from point load. Shear force distribution from point load.

Figure 6.42: Rigid links are modelled as springs with infinite stiffness to transfer forces between elements that have no connection when extracted from IFC.

During the process of obtaining a connected model from IFC to Karamba3D, the columns was split in areas were beams intersect. This splitting does however not affect the structural behaviour of the column, as default settings in Karamba3D create fixed connections unless specified otherwise as mentioned in Section 5.4.1. This will apply to all connections between artificial elements and their respectively connected beams or columns, and for columns that are split at intersection points. The fully fixed connections will restrict all degrees of freedom making them unable to rotate or translate at these locations. As a result, all forces are transferred and therefore representing a continuous column.

The beam joints creates pinned connections, as implemented in the Calculation Report. By connecting these at all nodes were beams intersect with columns, similar connections as in the Calculation Report is achieved. The principle of implementing beam joints are illustrated in Figure 6.43 where a simple frame is uniformly loaded.

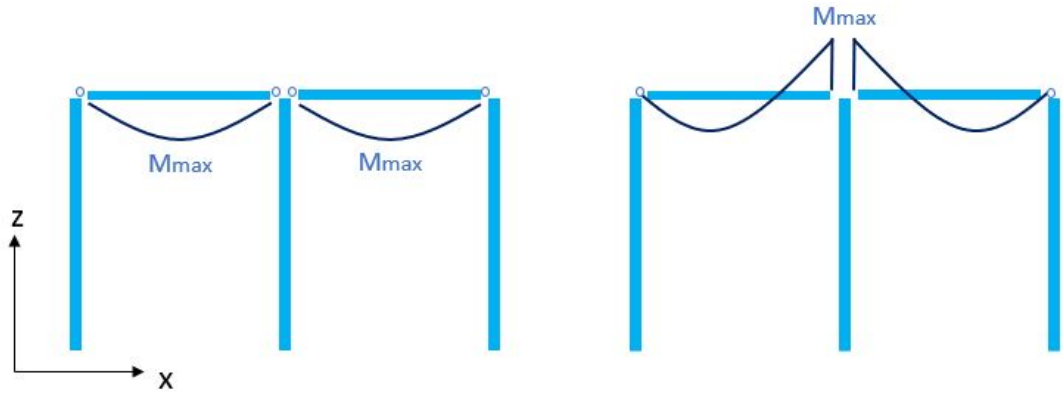


Figure 6.43: The effect of fixed and pinned connections on moment diagram.

In the Calculation Report, only two brace elements were designed to handle compression forces. While the remaining 24 brace elements were allowed tension forces. This constraint was not taken into consideration in this FEM Verification. Hence, compression forces occurred in more than two brace elements. Most of the brace elements did however only take tension. The maximum axial force when wind load was applied, occurred in one of the diagonal members as presented in Figure 6.40.

The occurring compression forces in the brace elements could be assessed by manipulating the cross sections to achieve a lower moment of inertia and hence making the element more prone to buckling. This would limit the element from distributing force, and buckle instead. The cross section of these typical qualities are circular non-hollow ones. This could serve a similar purpose as applying plastic qualities to members that experience compression, as in the Calculation Report.

The FEM Verification Analysis showed generally similar results as the Calculation Report, indicating that the same abilities were assigned in Karamba3D as in the software used in the Calculation Report. However, improved modelling of brace elements would likely improve the similarity further.

7 Discussion

The following chapter presents a comprehensive discussion encompassing various observations and reflections made during the development of the proposed method. These insights shed light on important aspects and may be valuable for further understanding and refinement of the method.

The IFC files used in Case Study 1 were specifically customised for this master's thesis, with a clear understanding of their intended purpose. In Case Study 2, two IFC files from the construction industry were employed. The IFC Data Extraction demonstrated satisfactory performance in both case studies. It successfully collected the desired quantities and properties of elements from IFC files using IfcOpenShell, with the exception of the location of the structure in Case Study 2. However, observations from the case studies and prior testing of IFC files have revealed notable inconsistencies in their data structure. Variations in naming conventions, data structures, and relationships between elements pose challenges in constructing a generic script that can seamlessly extract the desired information from all IFC files. Encouraging a more standardised data structure in IFC files, regardless of company procedures or software utilised, will enhance the efficiency of information extraction.

The IFC Data Extraction shows considerable potential and could be further developed to handle a supply bank containing elements from various buildings, rather than being limited to a single building as it currently is. This concept of creating a comprehensive material database holds great potential for streamlining the process of mapping elements, leading to substantial time- and effort-savings in future reuse projects.

The creation of the DataFrame is performed using the IfcOpenShell library in Python, which is continuously being developed and improved. While this development ensures advancements in functionality, it also presents certain challenges. The frequent updates and improvements imply that the documentation may not always be comprehensive or up-to-date, making it difficult to find solutions to specific problems. Additionally, due to the evolving nature of IfcOpenShell, new issues and challenges often arise that have not yet been addressed or resolved by the development team. To overcome these difficulties, seeking assistance from online forums and discussion boards was an essential part of the extraction process. Engaging with the community and exchanging knowledge and experiences with other users proved invaluable in finding workarounds and solutions to issues that may not have official documentation available.

The development of the Element Mapper aimed to utilise the established Bin Packing Problem to address multiple aspects of the reuse problem. In a potential further development of the method, notable improvements to the optimisation algorithm should be examined. The current bin packing algorithm in the Element Mapper aims to reduce the cut waste material in terms of length rather than volume. To enhance the functionality of the method, it is advisable to include the consideration of volume-based cut waste in the mapping process. Additionally, exploring optimisation objectives such as waste reduction, cost minimisation, or a balanced combination of both, would provide valuable insights into the trade-offs involved. Furthermore, it would enable a more comprehensive analysis of the methods' performance and the relevance for application in the construction industry.

In Case Study 2, it was observed that the elements used in the Reuse Structure resulted in a significantly larger volume. Currently, the Element Mapper does not have an upper limit for matching elements based on their cross section properties. This implies that a smaller element can be paired with a significantly larger element in relation to cross section area. As a result, the overall weight of the structure may increase significantly, leading to higher costs and a greater environmental impact in terms of GWP rather than if smaller cross sections would have been used. To address this issue and optimise the outcomes in terms of cost, GWP, and material waste reduction, it is recommended to further investigate the incorporation of an upper limit for the cross section area.

To significantly enhance the applicability of the presented reuse method in real projects, it would be es-

sential to expand the range of cross section types available in Karamba3D. This is necessary in order to assure accurate results in the structural analysis. The production of a shared database or a Grasshopper plugin containing custom-made cross sections, should ideally be created and shared within the Grasshopper community. This collaborative approach will contribute to streamlining the process and reduce the overall workload of incorporating custom cross sections.

Grasshopper demonstrates its efficiency as a valuable tool in the development of establishing a reuse method of structural elements based on IFC files. As the environment is integrated with the CAD application Rhino, this offers a visual interface that improves comprehension and visualisation of the structure. Using an IFC-based structural analysis in Karamba3D, introduced an effective method of performing a structural analysis without having to build the model up from scratch. This can be time-consuming and is currently the common practice in structural engineering. The analysis model from Karamba3D was compared with a calculation report of the same structure in order to verify the implemented method. The findings highlight that the IFC-based model performed similarly to the results obtained in the Calculation Report. However, certain adjustments were required in the assembled model within Karamba3D. This method presents a promising avenue for conducting accurate structural analyses using IFC in a more generic manner.

8 Concluding remarks

This master's thesis proposes a method for effectively reusing structural elements in construction projects. The method utilises IFC files to extract information on available building materials and identifies suitable integration into new structures based on predefined requirements. The structural integrity of the constructions using reused elements is ensured through a thorough structural analysis performed on an IFC-based model in Grasshopper.

The two comprehensive case studies conducted in this master's thesis have provided valuable insights into the functionality and effectiveness of the proposed method. The successful extraction of material data from IFC files has facilitated the creation of a comprehensive database of reusable elements. The mapping process between elements has demonstrated the ability to create predefined structures by utilising a substantial portion of reused materials while reducing cut waste.

Furthermore, the implementation of structural analysis in Karamba3D has confirmed the preservation of structural integrity when incorporating reused elements for the case studies. The results obtained from the analysis were compared to an external calculation report. With the level of accuracy achieved, the implemented method has demonstrated significant potential in simplifying the analysis process by directly utilising IFC files.

Overall, the findings of this master's thesis demonstrate that the proposed method offers a viable approach to reusing structural elements in construction projects. By reducing the demand for new materials, conserving resources, and minimising environmental impacts, this method contributes to the goal of achieving a more sustainable construction industry. Additionally, the methods introduced show great potential in streamlining the design process of future reuse projects, making such a choice more favourable in terms of time- and cost-effectiveness. Through further research and implementation of these methods, a more circular economy can be promoted within the construction sector.

Bibliography

[1] Autodesk Inc. *Revit IFC manual. Detailed instruction for handling IFC files*. 2018. URL: https://damassets.autodesk.net/content/dam/autodesk/draftr/2528/180213_IFC_Handbuch.pdf.

[2] I. Bertin et al. 'Construction, deconstruction, reuse of the structural elements: The circular economy to reach zero carbon'. In: *IOP Conference Series: Earth and Environmental Science*. Vol. 323. 1. 2019. DOI: 10.1088/1755-1315/323/1/012020.

[3] Jan Brütting, Gennaro Senatore and Corentin Fivet. 'Optimization formulations for the design of low embodied energy structures made from reused elements'. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10863 LNCS. 2018. DOI: 10.1007/978-3-319-91635-4{_}8. URL: https://link.springer.com/chapter/10.1007/978-3-319-91635-4_8.

[4] buildingSMART International. *IFC Formats*. 2023. URL: https://technical.buildingsmart.org/standards/ifc/ifc-formats/.

[5] buildingSMART International. *IFC Specifications Database*. 2023. URL: https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/.

[6] buildingSMART International. *Industry Foundation Classes (IFC)*. 2023. URL: https://technical.buildingsmart.org/standards/ifc/.

[7] BuildingSMART International Ltd. *Industry Foundation Classes 4.0.2.1*. 2020. URL: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/.

[8] buildingSMART International Ltd. *IFC GUID*. URL: https://technical.buildingsmart.org/resources/ifcimplementationguidance/ifc-guid/.

[9] Aurimas Bukauskas et al. 'Form-Fitting strategies for diversity-tolerant design'. In: *Proceedings of IASS annual symposia*. Vol. 2017. 17. 2017.

[10] Gaochuang Cai and Danièle Waldmann. 'A material and component bank to facilitate material recycling and component reuse for a sustainable construction: concept and preliminary study'. In: *Clean Technologies and Environmental Policy* 21.10 (2019). ISSN: 16189558. DOI: 10.1007/s10098-019-01758-1.

[11] Catherine De Wolf. 'Low Carbon Pathways for Structural Design: Embodied Life Cycle Impacts of Building Structures'. PhD thesis. 2017.

[12] Clemens Preisinger. 'Linking Structure and Parametric Geometry'. In: *Architectural Design* 83 (2013), pp. 110–113. URL: https://onlinelibrary.wiley.com/doi/epdf/10.1002/ad.1564?src=getftr.

[13] COWI. *Tennebekk buss hall Calculation Report*. Tech. rep. Bergen, 2023.

[14] Mauro Dell'Amico, Fabio Furini and Manuel Iori. 'A branch-and-price algorithm for the temporal bin packing problem'. In: *Computers and Operations Research* 114 (2020). ISSN: 03050548. DOI: 10.1016/j.cor.2019.104825.

[15] Entra. *Erfaringsrapport ombruk KA13*. Tech. rep. 2021.

[16] European Environment Agency. *cradle to grave*. URL: https://www.eea.europa.eu/help/glossary/eea-glossary/cradle-to-grave.

[17] *GeometryGym*. URL: https://geometrygym.wordpress.com/.

[18] Google. *Kronstad, Bergen til Tennebekk, 5171 Bjørndalstræ - Google Maps*. URL: https://www.google.com/maps/dir/Kronstad,+Bergen/Tennebekk,+5171+Bj%C3%B8rndalstr%C3%A6/@60.3753687,5.2162278,12z/data=!4m14!4m13!1m5!1m1!1s0x463cf94c19ffbe19:0x392be3a8a92fdbde!2m2!1d5.3448942!2d60.3771533!1m5!1m1!1s0x463cfb9feffd2eb7:0x1990fc1b19cc9f30!2m2!1d5.246889!2d60.363472!3e0?entry=ttu.

[19] Grønland Stein Erik. 'Kostnadsmodeller for transport og logistikk'. In: (2022). URL: https://www.toi.no/getfile.php?mmfileid=73913.

[20] Linnea Harala et al. 'Industrial ecosystem renewal towards circularity to achieve the benefits of reuse - Learning from circular construction'. In: *Journal of Cleaner Production* 389 (Feb. 2023), p. 135885. ISSN: 0959-6526. DOI: 10.1016/J.JCLEPRO.2023.135885.

[21] Vilde Høydal and Hanna Walter. 'Ombruk av byggematerialer og - produkter i et bærekraftperspektiv'. PhD thesis. NTNU, 2020.

[22] Eleni Iacovidou and Phil Purnell. *Mining the physical infrastructure: Opportunities, barriers and interventions in promoting structural components reuse*. 2016. DOI: 10.1016/j.scitotenv.2016.03.098.

[23] *IFC Certified Software - buildingSMART International*. URL: https://www.buildingsmart.org/compliance/software-certification/certified-software/.

[24] IfcOpenShell Contributors. *IfcOpenShell 0.7.0 documentation*. 2022. URL: https://blenderbim.org/docs-python/index.html.

[25] *IfcPatch - IfcOpenShell 0.7.0 documentation*. URL: https://blenderbim.org/docs-python/ifcpatch.html.

[26] Eg Coffman Jr, M.R. Garey and Ds Johnson. 'Approximation algorithms for bin packing: A survey'. In: m (1996). ISSN: 1098-6596. URL: https://www.labri.fr/perso/eyraud/pmwiki/uploads/Main/BinPackingSurvey.pdf.

[27] Karamba3D. *Karamba3D*. 2022. URL: https://manual.karamba3d.com/3-in-depth-component-reference/3.3-cross-section/3.3.3-spring-cross-sections.

[28] McNeel Europe. *food4Rhino*. 2023. URL: https://www.food4rhino.com/en.

[29] Linfeng Mei and Qian Wang. *Structural optimization in civil engineering: A literature review*. 2021. DOI: 10.3390/buildings11020066.

[30] Norsk Stål AS. *Norsk stål*. 2022. URL: https://www.norskstaal.no/.

[31] Norwegian Institute of Bioeconomy Research Norwegian Environment Agency Statistics Norway. 'Greenhouse Gas Emissions 1990- 2018, National Inventory Report'. In: (). URL: https://www.miljodirektoratet.no/globalassets/publikasjoner/m1643/m1643.pdf.

[32] Martin N. Nwodo and Chimay J. Anumba. *A review of life cycle assessment of buildings using a systematic approach*. 2019. DOI: 10.1016/j.buildenv.2019.106290.

[33] OECD. *Organisation for Economic Co-operation and Development*. URL: https://www.oecd.org/norway/.

[34] Robert McNeel & Associates. *Rhinoceros*. 2023. URL: https://www.rhino3d.com/.

[35] Shahrzad Saremi, Seyedali Mirjalili and Andrew Lewis. 'Grasshopper Optimisation Algorithm: Theory and application'. In: *Advances in Engineering Software* 105 (2017). ISSN: 18735339. DOI: 10.1016/j.advengsoft.2017.01.004.

[36] Patricia Schneider-Marin and Werner Lang. 'Environmental costs of buildings: monetary valuation of ecological indicators for the building industry'. In: *International Journal of Life Cycle Assessment* 25.9 (2020). ISSN: 16147502. DOI: 10.1007/s11367-020-01784-y.

[37] A. van Stijn et al. 'A Circular Economy Life Cycle Assessment (CE-LCA) model for building components'. In: *Resources, Conservation and Recycling* 174 (2021). ISSN: 18790658. DOI: 10.1016/j.resconrec.2021.105683.

[38] United Nations Environment Programme. *Global Status Report for Buildings and Construction*. Tech. rep. 2022. URL: www.globalabc.org..

Appendix

A Grasshopper Files

A *case1_mapper_and_fem_analysis.gh*

B *case2_mapper_buebygget_cross_section.gh*

C *case2_mapper_random_cross_section.gh*

D *case2_fem_analysis.gh*

E *verification_fem_analysis.gh*

F Python component in Grasshopper, Element Mapper

```
"""Provides a scripting component.
    Inputs:
        demand: testsetste
        y: The y script variable
    Output:
        a: The a output variable"""

__author__ = "Eier"
__version__ = "2023.02.01"

import rhinoscriptsyntax as rs
import ghpythonlib.treehelpers as th
import Rhino

""" METHODS """

def tree2lol(input_tree):
    # https://developer.rhino3d.com/guides/rhinopython/grasshopper-datatrees-and-
        python/    ]
    lol=[]
    for i in range(input_tree.BranchCount):
        sublist = []
        branchPath = input_tree.Path(i)
        branchList = input_tree.Branch(i)
        for j in range(branchList.Count):
            sublist.append(branchList[j])
        lol.append(sublist)
    return lol

def lol2tree(input_lol):
    # https://developer.rhino3d.com/guides/rhinopython/grasshopper-datatrees-and-
        python/
    tree = th.list_to_tree(input_lol, source=[0,0])
    return tree

""" INPUT """

# Turn to list of lists
demand_list = tree2lol(demand)
supply_list = tree2lol(supply)
supply_list_original = tree2lol(supply)
```

```
# Add original index (branch path) because later we delete some
[demand_list[i].append(i) for i in range(len(demand_list))]
[supply_list[i].append(i) for i in range(len(supply_list))]
[supply_list_original[i].append(i) for i in range(len(supply_list_original))]
a = lol2tree(demand_list)

""" CALCULATION """

mapping_id = []
logs = []

total_length_demand = 0 #Total length of all demand elements (matched and unmatched
    )
total_volume_demand= 0 #Total volume of all demand elements (matched and unmatched)

total_length_supply = 0 #Total length of all supply elements (matched and unmatched
    )
total_volume_supply= 0 # Total volume of all supply elements (matched and unmatched
    )

matched_length_demand = 0 #Total length of demand elements that have matched
unmatched_length_demand= 0 #Total length of demand elements that have not been
    matched

matched_volume_demand = 0 #Total volume of demand elements that have matched
unmatched_volume_demand=0 #Total volume of demand elements that have not been
    matched

matched_length_supply = 0 #The total length of all supply elements that have been
    matched (includes the cut_waste)
total_used_volume_supply = 0 #The total volume of all supply elements that have
    been matched (includes the cut_waste)

no_match_indices = []
matched_indices=[]
used_supply_indices = []
duplicate_indexes=[]

nr_demand_elements=0
nr_supply_elements=0

#Total demand length and volume
for i in range(len(demand_list)):
    if demand_list[i][5]=="Steel":
        nr_demand_elements+=1
        total_length_demand+=int(demand_list[i][0])
        total_volume_demand+=int(demand_list[i][0]*demand_list[i][1]) *(10**(-9))


#Total supply length and volume
for j in range(len(supply_list)):
    if supply_list[j][5]=="Steel":
        nr_supply_elements+=1
        total_length_supply+=int(supply_list[j][0])
        total_volume_supply+=int(supply_list[j][0]*supply_list[j][1]) *(10**(-9))


#Matching process
for i in range(len(demand_list)):
    match=False
    for j in range(len(supply_list)):
```

```
        if demand_list[i][0] <= supply_list[j][0] and 0.8*(demand_list[i][1]) <=
            supply_list[j][1]  and demand_list[i][5]==supply_list[j][5]=="Steel"
            and 0.8*(demand_list[i][8]) <= supply_list[j][8]:
            match=True
            mapping_id.append(supply_list[j][-1])
            matched_length_demand+=int(demand_list[i][0])
            matched_volume_demand+=(int((demand_list[i][1])*(demand_list[i][0])))
                *(10**(-9))
            unique_indices=set([index for index in mapping_id if index is not None
                ])
            dup_indexes=[index for index in mapping_id if mapping_id.count(index) >
                 1 and index is not None]
            duplicate_indexes=list(set(dup_indexes)) #gives us the indexes in the
                supply list of the elements used for more than one demand element.
            used_supply_indices = (unique_indices)

            break

    if match:
        if plural_assign:

            logs.append("#"+str(i)+" Found element #"+str(j)+" and utilized only "+
                str(demand_list[i][0]/1000)+"m of "+str(supply_list[j][0]/1000)+"m
                .")
            supply_list[j][0] = supply_list[j][0] - demand_list[i][0]
            matched_indices.append(demand_list[i][-1])

        else:
            print("null")
            del supply_list[j]
            logs.append("#"+str(i)+" Found element #"+str(j)+" and utilized fully.
                Demand: L="+str(demand_list[i][0]/1000)+"m.")
    else:
        unmatched_length_demand+=int(demand_list[i][0])
        print("null")
        mapping_id.append(None)
        unmatched_volume_demand+=(int((demand_list[i][1])*(demand_list[i][0])))
            *(10**(-9))
        no_match_indices.append(demand_list[i][-1])
        logs.append("#"+str(i)+" Not found. Demand: L="+str(demand_list[i][0]/1000)
            +"m.")



matched_demand_lengths=[]
cut_waste_length= []
cut_waste_volume=[]

for j in range(len(supply_list)):
    matched_demand_length=sum(demand_list[i][0] for i in range(len(demand_list)) if
         mapping_id[i]==supply_list[j][-1])
    matched_demand_lengths.append(matched_demand_length)

for i in range (len(supply_list_original)):
    if (matched_demand_lengths[i] != 0):
        cut_waste_length.append(supply_list_original[i][0] - matched_demand_lengths
            [i])
    else:
        cut_waste_length.append(0)

for i in range(len(supply_list_original)):
    cut_waste_volume.append(cut_waste_length[i]*supply_list_original[i
        ][1]*(10**(-9)))
```

```
#Total cut-waste
total_cut_waste_volume=sum(cut_waste_volume)
total_cut_waste_length=sum(cut_waste_length)

for k in used_supply_indices:
    matched_length_supply +=int(supply_list_original[k][0])
    total_used_volume_supply +=((int(supply_list_original[k][1]))*(int(
        supply_list_original[k][0])))*(10**(-9))

nr_used_supply_elements=len(unique_indices)
nr_matched_demand=len(matched_indices)

""" OUTPUT """

mapping_id = lol2tree(mapping_id)
```

B Visual Studio Code

A IFC Data Extraction

```
#IFC Data Extraction

#Import packages
import pandas as pd
import ifcopenshell
import ifcopenshell.geom
import ifcopenshell.util.unit
import ifcopenshell.util.shape
import ifcopenshell.util.element


#Store IFC file as a variable
ifc_file = ifcopenshell.open(r"ifc_path.ifc")

#Settings
settings=ifcopenshell.geom.settings()
settings.set(settings.DISABLE_OPENING_SUBTRACTIONS, True)

#Units
unit_scale=ifcopenshell.util.unit.calculate_unit_scale(ifc_file)

global_unit_assignments = ifc_file.by_type("IfcUnitAssignment")
global_length_unit = [u for ua in global_unit_assignments for u in ua.Units if u.
    is_a() in ('IfcSIUnit', 'IfcConversionBasedUnit') and u.UnitType=='LENGTHUNIT
    '][-1]
#print(global_length_unit)

global_volume_unit = [u for ua in global_unit_assignments for u in ua.Units if u.
    is_a() in ('IfcSIUnit', 'IfcConversionBasedUnit') and u.UnitType=='VOLUMEUNIT
    '][-1]
#print(global_volume_unit)

global_area_unit = [u for ua in global_unit_assignments for u in ua.Units if u.is_a
    () in ('IfcSIUnit', 'IfcConversionBasedUnit') and u.UnitType=='AREAUNIT'][-1]
#print(global_area_unit)

global_mass_unit = [u for ua in global_unit_assignments for u in ua.Units if u.is_a
    () in ('IfcSIUnit', 'IfcConversionBasedUnit') and u.UnitType=='MASSUNIT'][-1]
#print(global_mass_unit)
```

```
#Extract IfcBeams and IfcColumns
LB_elements = []
for element_type in ['IfcBeam','IfcColumn']:
    LB_elements.extend(ifc_file.by_type(element_type))

quantites=[]
for element in LB_elements:
    elem = ifcopenshell.geom.create_shape(settings, element)

    #psets = ifcopenshell.util.element.get_psets(element, qtos_only=True) #quantity
        sets
    #psets = ifcopenshell.util.element.get_psets(element) #property sets

    #Quantites
    volume=ifcopenshell.util.shape.get_volume(elem.geometry)
    x=ifcopenshell.util.shape.get_x(elem.geometry) /unit_scale
    y=ifcopenshell.util.shape.get_y(elem.geometry) /unit_scale
    z=ifcopenshell.util.shape.get_z(elem.geometry) /unit_scale


    guid=element.GlobalId
    name=element.Name
    owner_history=element.OwnerHistory
    moment_of_interia=float("nan")
    object_type=element.ObjectType

    #Material
    #material=ifcopenshell.util.element.get_material(element, should_skip_usage=
        True)
    #material_name=material.Name

    material = None
    if element.HasAssociations:
        for association in element.HasAssociations:
            # Check if the association is of type IfcRelAssociatesMaterial
            if association.is_a("IfcRelAssociatesMaterial"):
                # Get the associated material
                if association.RelatingMaterial:
                    if hasattr(association.RelatingMaterial, "Name"):
                        material = association.RelatingMaterial.Name
                break


    if element.is_a("IfcBeam"):
        cross_section_area=(volume/ x) *1000 #m^2
        length= x #mm
        height= z #mm
        width= y #mm

    elif element.is_a("IfcColumn"):

        cross_section_area=(volume/z) *1000 #m^2
        length= z #mm
        height=y #mm
        width= x #mm


    #GEOLOCATION
    postal_address=ifc_file.by_type("Ifcpostaladdress")
    site=ifc_file.by_type("IfcSite")
```

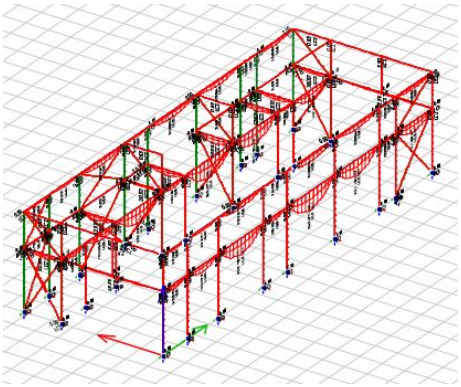
```
    for j in postal_address: #Note: The name of the attribute may not neccessarily
        correspond to the string or value attached
        region = j.Region if hasattr(j, "Region") else None
        addresslines = j.AddressLines if hasattr(j, "AddressLines") else None
        country = j.Country if hasattr(j, "Country") else None
        postal_code = j.PostalCode if hasattr(j, "PostalCode") else None
        postal_box = j.PostalBox if hasattr(j, "PostalBox") else None
        town= j.Town if hasattr(j, "Town") else None

    #address_attributes=postal_address[0].get_info().keys()
    #site_attributes=site[0].get_info().keys()

    location=[region, postal_code, country]
    location_str=",".join(location)

    """
    if all(item is None for item in location):
        location_str = ""  # Assign an empty string if all elements are None. No
            location assigned.
    else:
        location_str = ",".join(str(item) if item is not None else "" for item in
            location)
    """

    for s in site:
        long = s.RefLongitude if hasattr(s, "RefLongitude") else None
        lat = s.RefLatitude if hasattr(s, "RefLatitude") else None
        elev = s.RefElevation if hasattr(s, "RefElevation") else None
        ad = s.SiteAddress if hasattr(s, "SiteAddress") else None


        if long:
            long_float = long[0] + (long[1] / 60) + (long[2] / 3600) + long[3] /
                (3600 * 1000000)

        if lat:
            lat_float = lat[0] + (lat[1] / 60) + (lat[2] / 3600) + lat[3] / (3600 *
                 1000000)

    #Add desired quantites and properties to the DataFrame
    element_dict={"Guid": guid, "Name": name, "Material": material, "Length [mm]":
        length,"Height [mm]": height, "Width [mm]": width,"Cross section area [m
        ^2]": cross_section_area, "Volume [m^3]": volume, "Cross section name":
        object_type,"Location": location_str, "Latitude":lat_float, "Longitude":
        long_float}
    quantites.append(element_dict)

elements_df = pd.DataFrame(quantites)

#Export DataFrame to Excel
elements_df.to_excel(r"desired_excel_pat.xlsx")
```

B IfcPatch

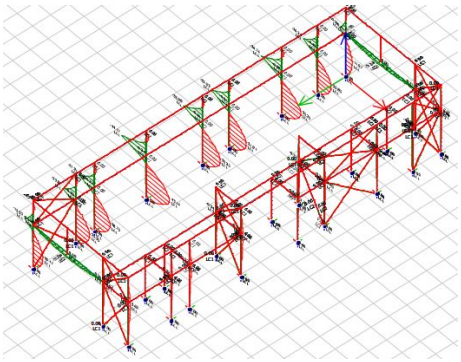
```
import ifcopenshell
import ifcpatch

ifc_path = r"C:\Users\elisemun\Documents\GitHub\vsc_test\18030_Modell_RIB_Som
    Bygget.ifc"
ifc_file = ifcopenshell.open(ifc_path)
```

```
output = ifcpatch.execute({
    "input": ifc_path,
    "file": ifcopenshell.open(ifc_path),
    "recipe": "ExtractElements",
    "arguments": [".IfcBeam|.IfcColumn"],
})
ifcpatch.write(output, "Desired_name_of_new_IFC.ifc")
```

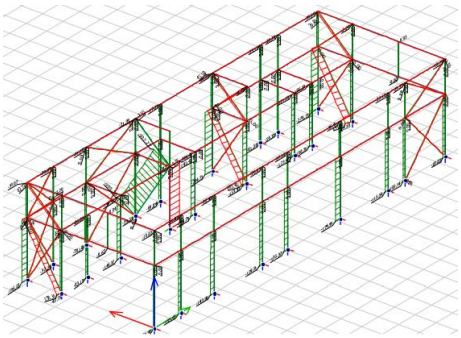
C Excerpt from Calculation Report



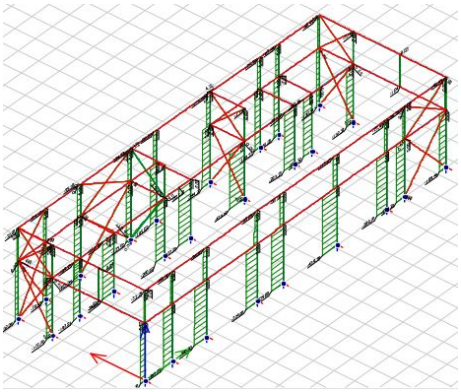
M_y LC1



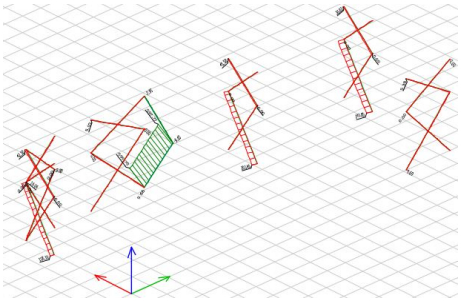
M_z LC1



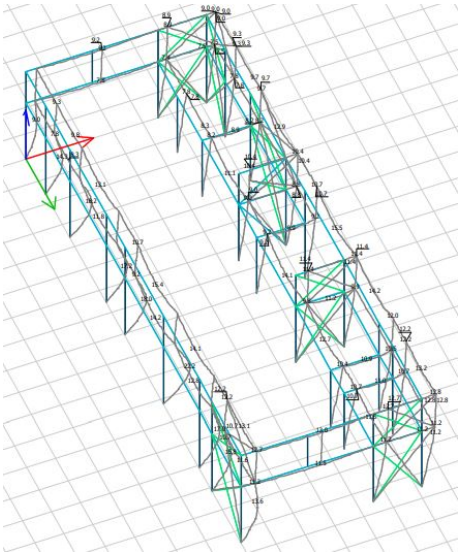
N_x LC1



N_x LC2

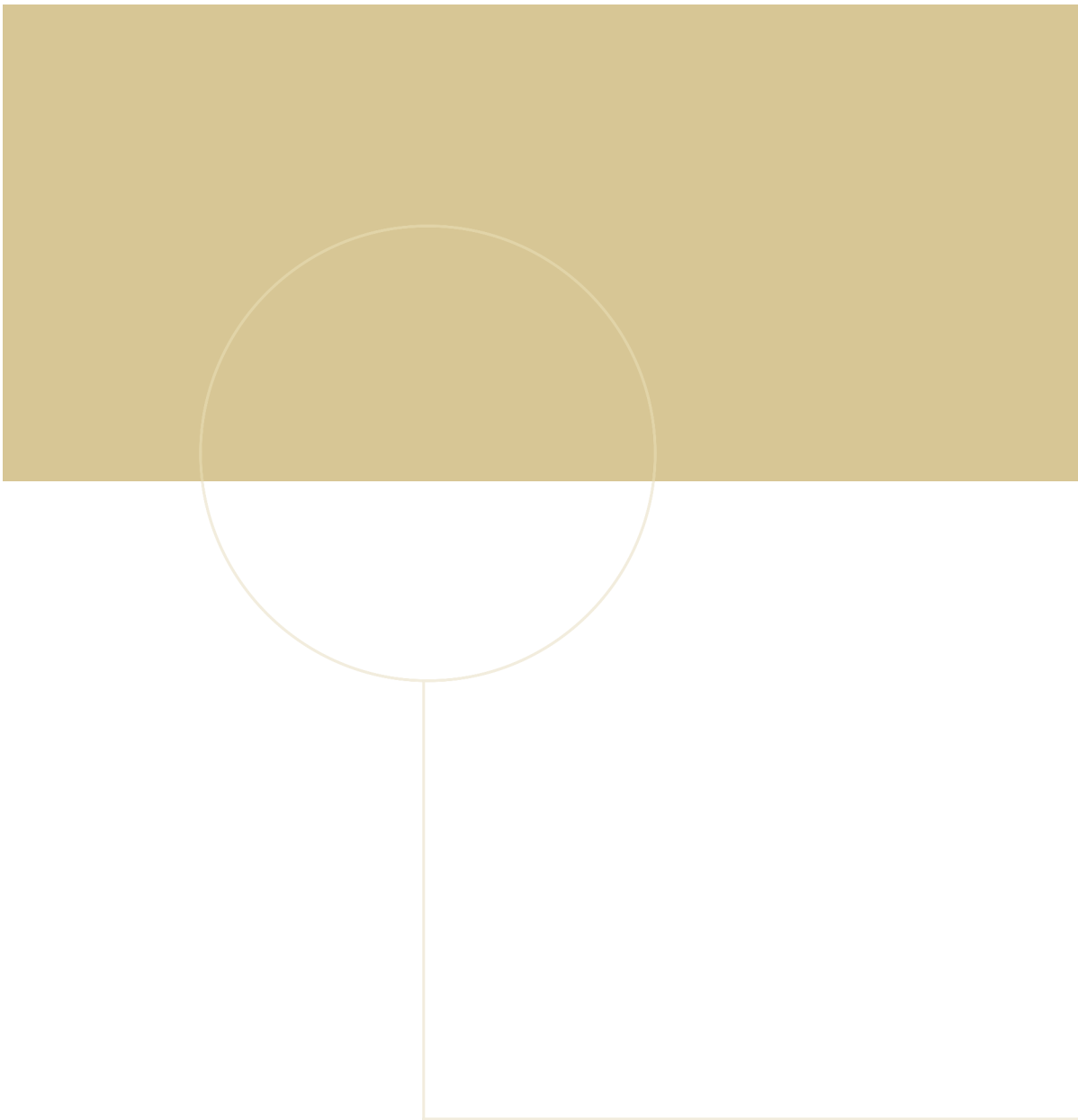


N_x LC1



Displacement, LC1

Figure C.1: Results from the Calculation Report.



NTNU

Norwegian University of
Science and Technology