T.C.S. van Oers

Master Thesis

Automated Reclaimed Steel Integration: A Machine Learning application within the Structural Design Process

Applying machine learning to evaluate automated structural designs of data centres which are based on available reclaimed steel elements

NOVEMBER 2024

TU Delft ARUP

Automated reclaimed steel integration: A machine learning application within the structural design process

Master Thesis

by

T.C.S. van Oers

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number:	4953479	
Project duration:	April 1, 2024 – December 1, 2024	
Thesis committee:	Dr. ir. F. Kavoura	TU Delft, supervisor
	Ir. A. C. B. Schuurman	TU Delft
	Dr. ir. R. Oval	TU Delft
	Ir. R. Crielaard	Arup

TU Delft

ARUP

Preface

I hereby present my thesis, the final step in order to conclude my Master's in Civil Engineering, thus marking the end of my journey at the TU Delft as a student. I started at this faculty in September 2018, wholeheartedly believing that my favourite courses would be anything with water, which very quickly turned out not to be true. I did not expect to be so interested in the design process of buildings, but courses like Construction Mechanics (especially 3) and Design for Construction and Foundations just worked for me. During my second year, I first got to work with Python and I was immediately intrigued by the endless possibilities it seemed to offer. So much so, that I did my Bachelor's thesis at the Remote sensing department. My interest in reuse of elements came later, during my master's. Where I participated in a joint interdisciplinary project with the municipality of Rotterdam. Our job there was to find the reason why reuse is not readily applied yet and propose a solution. While this research was more based on policies than I was used to, I was intrigued by the environmental and even monetary advantages reuse could have when applied correctly. So as we say in Dutch: *zo geschiede.*

I would like to thank my committee for the support and feedback they provided throughout the project. Florentia for helping me set up the project and answering all my steel- and organisation-related questions. Marco for the tips on the structure of both my designs and my report, and thorough feedback throughout the project. Robin for the tips on recent literature and developments on computational design and ML. I would also like to thank the team at Arup - particularly Rick, Simon, Emily, Stefan, Lotte, Mike, Roel, Predrag, Guus and most of all Roy - who have been very helpful every step of the way, always making the time to answer my questions or listen to my new ideas.

Next to the people helping me on a professional level, I would also like to express my gratitude to all my friends from JvB35, my current house, Dames 5 and everybody I met as a student, who stood by me even though I did not have a lot of time to catch up with them. Also thanks to the lovely Board 9 for working and studying alongside me during all the ups and downs a thesis endures. Lastly, I would like to thank my family. Mom, Dad, Suus, my brothers and of course Pepijn, with whom I could always spar and vent and think of new ideas. Thank you all.

Have fun reading my report!

T.C.S. van Oers
Delft, November 2024

Abstract

This research explores the integration of machine learning to support sustainable design in data centres by incorporating reclaimed steel elements into automated structural design workflows. The study begins with a literature review addressing the impact of steel reuse, the availability of reclaimed sections, strategies for automated, stock-constrained design and the applications of machine learning within the structural design process. Three reclaimed steel element databases are considered and three cross-section selection methods are evaluated and validated, establishing the optimal basis for a machine learning application within the design process. The cross-section selection method is used to gather data on generated design configurations for braced structures and moment-tight frame structures. Per element, the element location, length, type and profile are recorded for four separate design choice configurations. The data collected acts as a basis for a machine learning application for cross-section prediction.

Central to this research is the development and training of a Recurrent Neural Network (RNN) for sequential classification, aimed at predicting cross-sections based on design parameters. Four models were trained, each tailored to different combinations of design choices and reclaimed steel profiles. The second step in this workflow is the reclaimed steel integration which is an optimization model that integrates the RNN's predictions to refine grid-spaces in response to available reclaimed steel databases. Finally, the models and the resulting designs were validated to assess performance and applicability.

The trained RNN models reached a test accuracy ranging between 82% to 93% and the optimized steel integration resulted in design configurations which are slightly over-dimensioned, but all have an overall steel utilisation between 0.2 and 0.8. These findings underscore the feasibility of applying machine learning within the structural design domain to promote reclaimed steel integration at the early stages of the design process. A practical workflow is established that creates the possibility for adaptation to other building design typologies, making reuse-oriented design more accessible and efficient.

Front page image from Target Recycling Services inc. (2023)

Contents

List of Figures

List of Tables

1

Introduction

Following the Paris Agreement in 2015, a global reduction of 50% of all greenhouse gas (GHG) emissions is mandatory by 2030, followed by the goal to be completely carbon neutral by 2050 (Broer et al., 2022). As the construction sector is estimated to be responsible for 37% of all GHG emissions (UNEP, 2022), a shift towards environmentally conscious practices within this sector is necessary. To execute this transition, the integration of reclaimed materials into structural design has emerged as a promising area of exploration (Alaux et al., 2024). This research presents a solution to reduce environmental impact while enhancing resource efficiency in construction projects. Simultaneously, the up-rise of Machine Learning (ML) and Artificial Intelligence (AI) within research has been significant in recent years as almost ten per cent of all published papers now incorporate ML or AI in the title (Van Noorden and Perkel, 2023). However, combining these two relevant topics to create an automated design workflow, implementing ML to effectively incorporate reclaimed steel elements directly into the design has been under-explored (Liao et al., 2024).

1.1. Design Process

In structural engineering projects, the early phases of design are critical to influencing the project's outcomes while requiring relatively low expenditure (Bakker and de Kleijn, 2014). Figure 1.1 shows the development of influence versus the cost of change during the lifetime of a project, where the Final Investment Decision is the tipping point where changes within the project become too expensive.

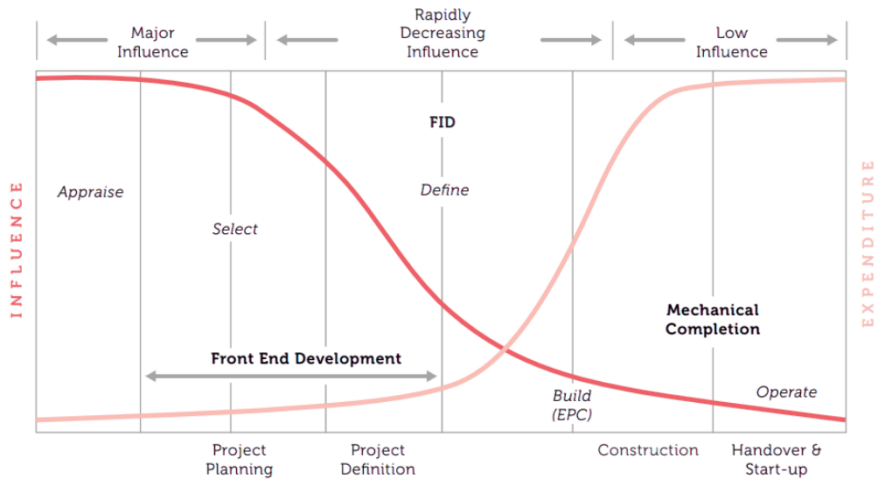


Figure 1.1: The development of influence versus expenditure during the progression of a structural design project (adapted from Bakker and de Kleijn, 2014)

The current workflow of incorporating reclaimed steel in a design is a time-consuming process. Figure 1.2 shows the steps needed to be taken, which form an inefficient feedback loop between engineering firms and steel distribution companies, as the design needs to be aligned with the available stock not known beforehand. This separated process creates a possibility for improvement. The feedback loop can be made faster, or even disregarded when an automated reclaimed steel integration method is applied within the design process that can quickly generate an initial design based on available stock. This would result in a more accurate initial concept design and time reduction between the concept design and the preliminary stage.

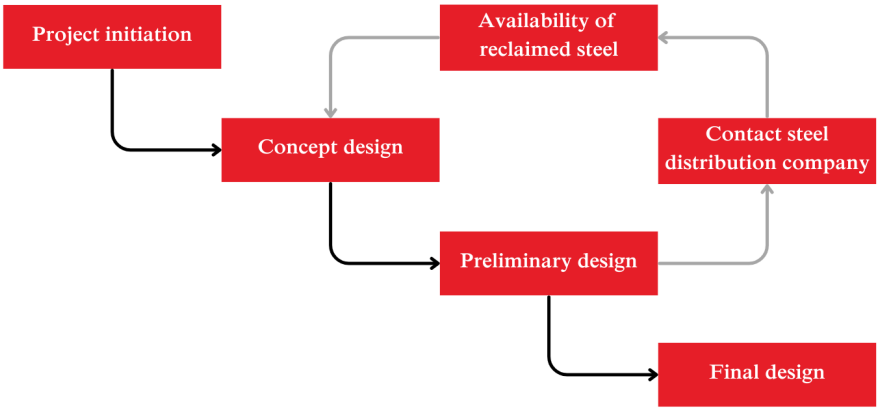


Figure 1.2: Current design process for reclaimed steel integration where a feedback loop between the engineering firm and the steel distribution companies is shown

1.2. Problem Statement

The core challenge addressed in this research is the adaptation of automated design processes to integrate reclaimed steel elements into the structural design of data centres. Although existing research has looked at the feasibility and benefits of incorporating reclaimed materials, methodologies to effectively implement these elements within automated design algorithms and tools remain underdeveloped. The main research question posed to tackle this problem is as follows:

"How can the incorporation of reclaimed steel elements be made more accessible within the design process of datacentres when using Machine Learning applications?"

To answer the main research question, the following sub-questions are set up:

1. What effect has the integration of reclaimed steel on the design approach and sustainability calculations of data centres?
2. What stock-constrained automated design processes have been researched and which apply best within this application?
3. What machine learning applications that assist in the structural design process have been researched?
4. How can machine learning techniques be applied to effectively incorporate reclaimed steel within the design process of data centres?
5. What are the key optimization variables for incorporating reclaimed steel into a design?

This research aims to bridge the knowledge gap on automatically integrating reclaimed steel by exploring innovative approaches with machine learning and combining previous research. The result of this research is a workflow that makes reclaimed steel integration more accessible for sustainable structural design practices.

1.3. Scope

The integration of automated design processes can be applied in many different aspects and stages of the design process. To narrow the scope of this research, this study focuses on the load-bearing structure of data centres with a focus on the data hall (the room that houses the server racks). This building function has the highest impact on the effectiveness of the data centre as poor floorplan layouts can lead to 50% more energy consumption due to inefficient cooling conditions (Rasmussen and Torell, 2015). Only the load-bearing structure is considered as most material used in the design is located there, leaving the most room for optimization of the total embodied carbon. The design variable chosen to be automated is the cross-section selection process and the floorplan grid spacings of the data hall. The variable grid spaces allow for an adaptive design to find the configuration with the most reclaimed steel applied. These boundaries result in a comprehensive and insightful study of the application of reclaimed steel elements.

1.4. Approach

The approach to answer the research questions can be split up into three parts: the literature study which acts as the theoretical framework, the application process and the validation of the research outcome. The overview of objectives and their respective approach, algorithm and tool are shown in Table 1.1 for the parts on application and validation.

Theoretical framework

- *Effect of reclaimed steel application to the design approach and sustainability calculations*: A literature study where the embodied carbon of reclaimed steel has been researched, as well as the design strategies needed to be implemented in order to incorporate reclaimed steel within the design process.
- *Data centre building design and design automation*: A literature study on the specific design requirements of data centres, combined with research on stock-constrained automated design.
- *Machine learning applications in the structural design process*: A literature study on the differences in algorithms, previously researched applications and the best fit for automated topology optimization.

Application

- *Initial design set-up and reclaimed steel element database generation*: This is based on the specific design requirements for data halls in data centre buildings. Three different types of reclaimed steel element databases are considered: one collected from open-source databases, one based on existing design and one generated based on Eurocode design standards.
- *Cross-section selection*: Two different approaches for the cross-section selection were combined to create a reliable cross-section profile selection result. This is applied for two configurations of cross-section profiles and two types of stability systems, resulting in four design choice combinations. The data has been recorded for almost five thousand different floorplan sizes and grid spacings for which the element ID, element type, element length and profiles for the four design choice combinations are saved.

- *Automated cross-section selection*: The generated dataset formed a basis for training and validating a Recurrent Neural Network (RNN) for sequential classification that is applied as an automated cross-section selection. Four separate models were set up and trained for the different design choice combinations.
- *Reclaimed steel integration*: The addition of reclaimed steel is done by combining the RNN for cross-section selection and a component matcher to combine the design with a reclaimed steel element database into a Grasshopper model. Where a matching algorithm is applied to find perfect matches, matches with cut-off length or matches with a bigger profile size are applied. The shape of the grid is then optimized with an evolutionary optimization using a genetic algorithm to find the configuration with the most matches to the reclaimed steel database.

Research outcome

- *Cross-section selection validation*: The validation of the combined cross-section selection method is done with hand calculations in Python and a structural analysis within Oasys GSA.
- *RNN model validation*: The RNN models are validated by running the model on unseen test data and plotting the predictions against the actual labels in a confusion matrix, from which the accuracy of the model can be calculated.
- *Design validation*: The resulting design configuration with the reclaimed steel integrated is validated with a final structural analysis done with finite element modelling in Oasys GSA.

Table 1.1: Objectives for the automated reclaimed steel integration with their respective approach, algorithm and tool.

Objective	Approach	Algorithm	Tool / Implementation
Initial design set-up	Data centre design requirements	-	Grasshopper
CS profile selection	Eurocode steel profile calculations	-	Karamba3D / Python
Automated CS selection	Machine Learning	Recurrent Neural Network	Python
Reclaimed steel integration	Object-oriented programming	-	Magpie (Grasshopper)
Shape optimization	Evolutionary optimization	Genetic algorithm	Opossum (Grasshopper)
CS selection validation	Hand calculations / Structural analysis	Finite Element Modeling	Python / GSA
RNN model validation	CS prediction for unseen test-dataset	-	Python
Design validation	Structural analysis	Finite Element Modeling	Oasys GSA

1.5. Report Outline

The structure of the report aligns with the order of tasks from subchapter 1.4. The first part of the report analyses the theoretical aspects of reuse, automated design, and machine learning. The second part presents the overall workflow with further explanation of the two model types: a Machine Learning model and an evolutionary optimization model. The report concludes with the final results, discussion and proposed further research. An overview of the report structure can be found in Figure 1.3.

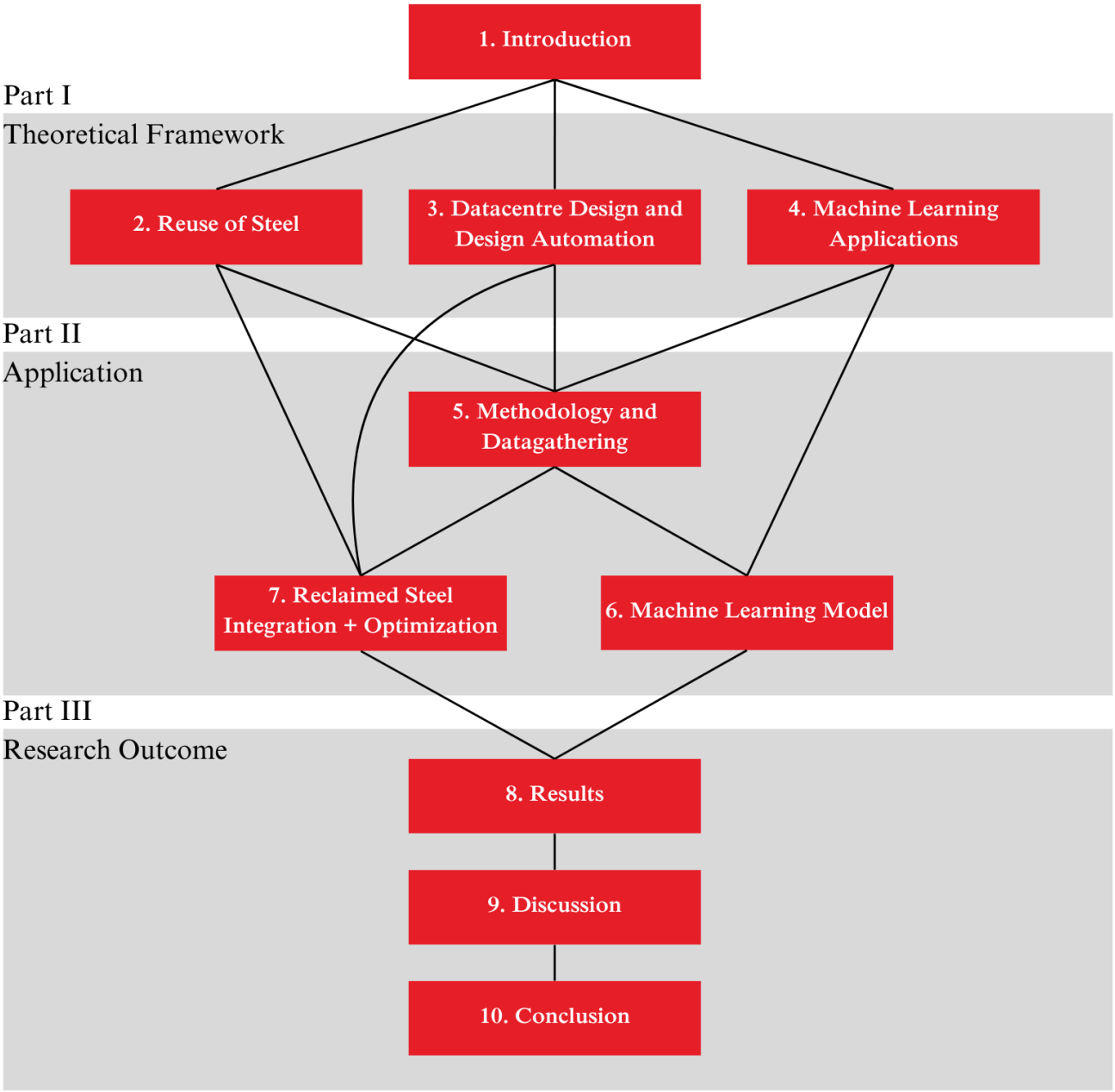


Figure 1.3: Overview of report structure which shows the links between theoretical chapters and application.

I

Theoretical Framework

2

Reuse of Steel Elements

The reuse of structural steel has emerged as a key strategy for reducing carbon emissions in construction, aligning with the European Commission's goal of climate neutrality by 2050 (EuropeanCommission, 2020). A pivotal development is the Dutch Technical Agreement NTA 8713 (2023), which provides clear guidelines for reusing structural steel, including dismounting, assessing material properties, adapting design methods, and preparing elements for reuse (NTA 8713, 2023). This report highlights that steel produced after 1972 can be considered new steel in terms of application and strength, provided it meets strict visual and structural checks, reinforcing its potential for long-term circular use. By streamlining reuse practices, such initiatives make it possible to reduce raw material use and GHG emissions, presenting a promising approach for sustainable design and life cycle calculations in the building industry. In this chapter, the first sub-question is further explored:

1. *What effect has the integration of reclaimed steel on the design approach and sustainability calculations of data centres?*

2.1. Impact of Steel Reuse

To explain the impact of steel reuse, clear life cycle stages of the material need to be identified. This is generalized within the Life Cycle Assessment (LCA) calculations of structures as described by Broniewicz and Dec (2022):

- **A1-3: Product Stage** - This stage involves the extraction and processing of raw materials, as well as the manufacturing of building products. It includes all the environmental impacts from the point of resource extraction to the delivery of the product to the construction site.
- **A4-5: Construction Process Stage** - This stage covers the transportation of building materials to the site and the construction of the building itself. It includes the energy used in construction and the waste generated on-site.
- **B1-7: Use Stage** - This is the longest stage in a building's life cycle and includes the use, maintenance, repair, replacement and refurbishment of the building. It also accounts for the energy used for heating, cooling and lighting during the building's operational phase.
- **C1-4: End of Life Stage** - This stage involves the deconstruction or demolition of the building, waste processing and disposal. It also includes the transportation of the waste and any potential recycling or recovery processes to its new location.

- **D: Benefits and Loads Beyond System Boundaries** - This stage considers the potential future generated or avoided environmental impacts. It includes the benefits of recycling materials from the demolished building into new products or the environmental load of waste disposal. As this model shows the benefits of reusing material it is interesting within this research. Therefore, the sub-modules, according to the Eurocode (15804+A2, 2019), are explained as well.
  - **D1**: Export of secondary materials.
  - **D2**: Export of secondary fuels.
  - **D3**: Export of energy as a result of waste incineration.
  - **D4**: Export of energy as a result of landfilling.

Each of these stages, or modules as they are known in the Netherlands, combined give the total overview of the environmental impact of a structure. To correctly identify the impact of reuse on the total carbon emissions, it is important to clarify its exact meaning and to keep the calculations constant.

2.1.1. Module D

The first four modules have been thoroughly assessed and researched for building structures, creating little room for discussion. But Module D on the other hand has been a controversial topic in recent years, as stated in an article by den Hollander (2024). In the article, the negative image of Module D is addressed. It is even mentioned by the writer that certain politicians accused the Module of being a tool for green-washing building designs. This is based on the fact that carbon emissions would be able to be reduced by promised actions made at the end-of-life (EoL) stage of the structure.

To clarify the application and differences of Module D when applied in different circumstances, MRPI conducted three thorough calculations on different design approaches. The calculations are done for a standard new-steel design, a design including reused material and a design for reuse at end-of-life. These calculations were done as a way to show the differences in Modules for these alternate design considerations. Since Module B would be the same for each iteration, it is left out of the calculations. Table 2.1 shows the resulting Global Warming Potential (GWP) expressed in embodied carbon per kg material for each of the life cycle stages. By comparing these values for the three design approaches, a clear overview is created of the specific stages that lower the total embodied carbon when reclaimed materials are applied. The table shows the highest differences in modules A1-A3 and D.

Table 2.1: GWP expressed in equivalent kg CO_2 per kg of steel which is emitted during the lifecycle of a building. Here three different design approaches are compared, one for raw material use, one with 90% reuse in the design and one structure which is designed to be reused at EoL. For the first two calculations, 16% reuse at EoL is taken into account as this is the current average of reuse at the EoL stage of buildings.

[$kgCO_2e/kg$]	Raw material, 16% reuse EoL (Kraaijenbrink et al., 2022c)	90% Reuse, 16% reuse EoL (Kraaijenbrink et al., 2022b)	Raw material, 80% reuse EoL (Kraaijenbrink et al., 2022a)
A1-A3	1.12	0.198	1.12
A4	0.0201	0.0201	0.0201
A5	0.0477	0.0477	0.0477
C1-C4	0.0807	0.0807	0.0808
D	−0.219	−0.0171	−0.761
Total	1.049	0.329	0.508

The high differences within module D can be further explained by looking at the formula for stage D1 in more detail, which is shown in equation 2.1.

$$e_{\text{moduleD1}} = \sum_{i} \left(M_{\text{MR,out}|_i} - M_{\text{MR,in}|_i}\right) * \left(E_{\text{MR,afterEoW,out}|_i} - E_{\text{VMSub,out}|_i} * \frac{Q_{\text{R,out}}}{Q_{\text{sub}}}|_i\right) \tag{2.1}$$

$M_{\text{MR,out}}$ = Amount of output material that will be recycled or reused

$M_{\text{MR,in}}$ = Amount of input material that is recycled or reused

$E_{\text{MR,afterEoW,out}}$ = Emissions from recovered material previous system

$E_{\text{VMSub,out}}$ = Emissions from material acquisition to substitution in a subsequent system

$\frac{Q_{\text{R,out}}}{Q_{\text{sub}}}$ = Quality ratio of recovered material versus otherwise applied material

The equation describes the material flow with parameter M, where the difference of recovered material entered into the system versus the amount of material that will be reused at end-of-waste point. These material flows are then multiplied by their relative emissions, as shown with parameter E. According to NTA 8713 (2023), it can be assumed that the quality of steel elements will not degrade over its reuse cycles when it passes the prescribed visual checks, resulting in the $\frac{Q_{\text{R,out}}}{Q_{\text{sub}}}$ being equal to one. The biggest benefits of the D module will be in the step from raw material use to design for reuse at EoL. However, since reusing elements will lead to a decreased value for A1-A3, the overall LCA calculation of a material per service life application will stagnate over its life cycles.

2.1.2. Literature on Effect of Reuse

Many different research groups have explored the topic of quantifying the positive effects of steel reuse by using LCA calculations. By looking at the effect on carbon emissions and costs, a complete picture can be painted to highlight the benefits and create a feasible business case for reuse. Buzatu et al. (2023) presents a comparative analysis of the environmental and economic impact of steel structures, through life cycle assessment (LCA) and life cycle cost (LCC) assessment. It was found that utilizing reclaimed steel structures leads to a 29-35% reduction in emissions compared to new steel. The study evaluates scenarios involving different levels of reuse, showing that even partial reuse of steel elements yields significant environmental and economic benefits, with the greatest gains observed in the production stage.

While Buzatu et al. (2023) focused on the benefits of steel reuse, Broniewicz and Dec (2022) took a more comparative approach by examining different design strategies based on LCA calculations calculated with GaBi software. Their research highlights the favourable Design for Deconstruction (DfD) strategy, which results in about 70% energy savings and an 80% reduction in CO_2 equivalent emissions compared to the scenario where the existing structure is remelted to fit a new design. Designing for deconstruction will lead to a more cost-competitive option to dismount the structure compared to demolition, as more elements can be recovered for a reasonable market value. However, it is important to note that this solution focuses on greenhouse gas reduction in the future while minimising emissions now is urgent as well.

A bit closer to home, Aardoom (2023) created a LCA and cost calculation tool called Steel-IT that builds on the previously mentioned papers together with the thesis of van Maastrigt (2019), where transportation costs are analysed in detail as well. Aardoom further facilitates the application of reuse as the tool creates a fast and thorough insight into the steel reuse possibilities and advantages. The key distinction between new and reclaimed steel lies in the design approach and stakeholder engagement. Although reclaimed steel designs are often more costly due to additional processes like re-fabrication, material

testing, and storage, a 25% environmental impact reduction can be achieved with minimal cost increase when there's a good match between donor elements and building geometry. The Steel-IT tool creates a fast and easy overview of reuse possibilities, making donor steel designs also more cost-competitive.

In summary, this section has explored the life cycle stages of buildings and the significant potential of reclaimed steel to promote sustainable construction practices. Research by Buzatu et al. (2023) and Broniewicz and Dec (2022) has highlighted the environmental and economic benefits of using reclaimed steel, demonstrating reductions in emissions and potential energy savings. Which are visualised effectively by Aardoom (2023) in the Steel-IT tool. Table 2.2 shows the total embodied carbon per kg of steel listed per source. This overview gives insight into the differences found in the literature.

Table 2.2: Greenhouse Gas emissions for new and reused steel as per three different sources. These account for the opening of connections, usage of the crane and preparation of the element.

[kgCO_2e/kg]	Brütting et al., 2020	van Maastrigt, 2019	Aardoom, 2023
New steel	0.894	0.985	1.0624
Direct reuse	0.447	0.109	0.3294
Indirect reuse	-	0.147	0.3423
Indirect reuse (waste)	-	0.085	0.0129
Transport	$1.1 * 10^{-4}[1/km]$	$1.32 * 10^{-4}[1/km]$	-

2.2. Design Strategies

Following a very recent study done by Alaux et al. (2024), a variety of design strategies have been developed to minimize the environmental footprint of buildings throughout their life cycles. These strategies emphasize optimizing material use and reducing energy demand, which collectively contributes to lowering the embodied and operational emissions of buildings. The report provides a comprehensive framework of 11 design strategies that outline the key approaches and their potential impact on carbon reduction and their effects are visualised in Figure 2.1.

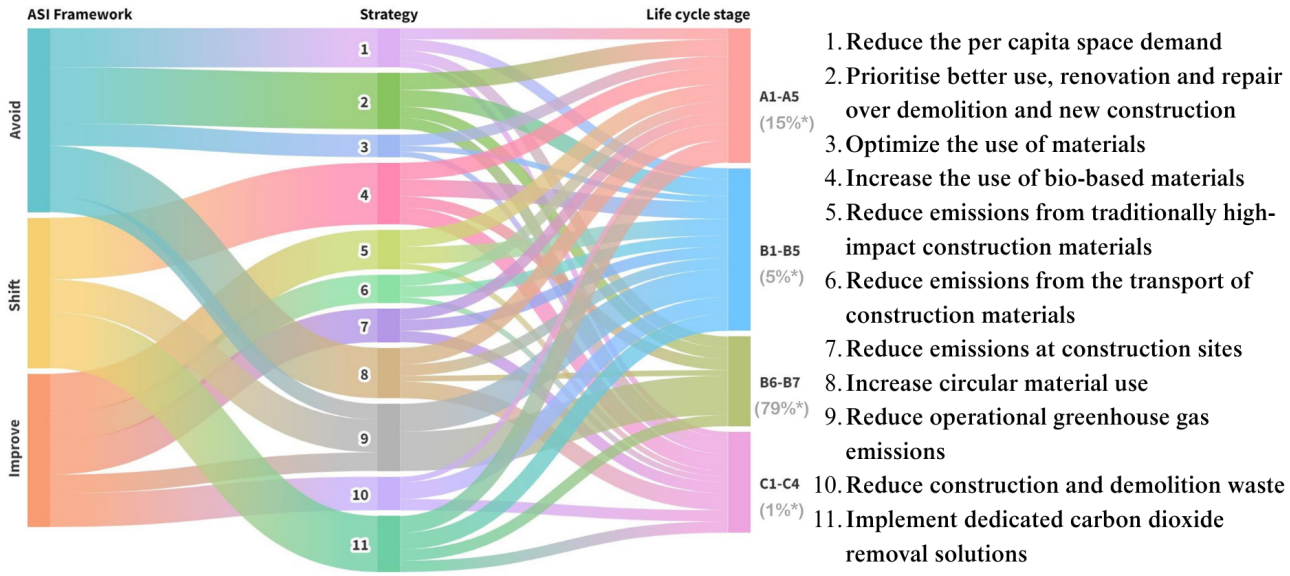


Figure 2.1: The figure from Alaux et al. (2024) shows an overview of the proposed design strategies and the respective impact on carbon reduction per life cycle stage of the building. * Shows the share of affected life cycle GHG emissions, based on average annual life-cycle related emissions of the EU building stock for the year 2020 from Le Den et al. (2023).

Within this research, the focus will go to design strategy eight, which includes the reuse of existing building components. Despite the environmental benefits of this design strategy, several obstacles hinder its widespread adoption. Rakhshan et al. (2020) identify key challenges, including the questionable state of many reusable components, which may no longer meet modern building standards. Additionally, there is a lack of clear regulations and uniform standards, creating uncertainty in the application. Finally, the labour-intensive process of deconstructing and re-purposing elements leads to higher costs, further complicating the feasibility of reuse in large-scale construction projects.

In order to effectively incorporate reclaimed elements into the design of new structures, a different design approach should be adopted. A Joint Inter-disciplinary Project (JIP) was established in collaboration with a municipality to determine the necessary changes to adopt a circular design and construction workflow (Keranis et al., 2023). They stated that several challenges need to be overcome for the successful transition from a linear to a circular construction sector. These include the perception that the circular approach is complicated, expensive, or even impossible to adopt and the lack of coordination between different departments. Following the release of new standard codes regarding reuse, Vergoossen et al. (2023) stated that the lack of knowledge on the possibilities of reuse created unnecessary discussions which haltered the distribution of knowledge. The following steps to overcome these barriers were identified within both studies:

- A Clear Vision: A clear and detailed vision for circular construction needs to be established. This vision should outline the goals and objectives of adopting a circular approach.
- Improve Coordination: There needs to be improved coordination and information sharing among various disciplines involved in the construction process. Creating a more efficient way of working between demolition, architectural design, structural design and execution.
- Establish a Circular Construction Process: A detailed circular construction process needs to be developed. This process should outline the steps involved in the efficient application of donor steel elements within the design process.
- Develop a Database of Reusable Elements: This database should include information on the availability, condition, and potential uses of each element.

By following these steps, the transition to a circular construction industry can be facilitated, contributing to the set sustainability goals.

These steps create a pathway for the application of reclaimed elements to enter the design process, which now needs to be implemented further. Kavoura and Veljkovic (2023) highlighted three distinct phases in the design process that are crucial for the successful reuse of structural elements. The first phase involves an assessment for potential reuse before deconstruction, creating a need for a systematic approach to inspection and documentation. Following deconstruction, the second phase entails the sampling and testing of elements, which ensures the reliability of the structural member. These aspects are elaborated further in Chapter 2.3. The third and final phase requires modifications to the design process to include reuse, taking into account the accurate load-bearing capacities and structural integrity of the elements. This is explained extensively in chapter 3.

2.3. Preparation Process

The release of NTA 8713 (2023) has introduced a standardized methodology for assessing steel elements in existing buildings to prepare them for reuse. This standard represents a significant step forward in streamlining reclaimed steel applications in the construction industry. Earlier research as done by Gordon et al. (2023), explored innovative ways to evaluate buildings for reuse. Their research, which included sensing and scanning technologies, Scan-to-BIM and computer vision, was applied in a Geneva case study and demonstrated the potential of advanced digital tools for deconstruction planning and recovery analysis. However, the comprehensive criteria outlined in NTA-8713 now provide a more actionable framework for the industry which is more widely applicable and cost-effective.

The NTA-8713 process begins with a detailed visual inspection of steel elements, resulting in a thorough inspection report. This document includes key assessments such as archival document status, conservation evaluations, material property verification, geometric deviations, potential consequence class applications, and weldability tests. A critical recommendation from the NTA is to classify all reused steel as S235 to account for potential variability in material properties, ensuring consistency and safety.

After passing inspection, reused steel elements must comply with the same structural codes as new materials. While the reuse of nuts and bolts is prohibited, steel profiles with pre-existing bolted connections are allowed if they meet visual and structural design requirements. The standard also emphasizes the importance of coating removal, particularly where toxic substances are involved. This step is crucial for making elements safe and viable for reuse, as confirmed by industry feedback (“Beschikbaarheid en process hergebruik constructiestaal. Interview by Tessel van Oers”, 2024).

2.4. Availability

The availability of steel beams and columns is a critical factor in the construction industry as mentioned in section 2.2, influencing both the design and execution phases of projects. One of the biggest hurdles when it comes to reclaimed steel is the cost of storage, due to their size, quantity and need for quality retention, according to a steel distribution company (“Beschikbaarheid en process hergebruik constructiestaal. Interview by Tessel van Oers”, 2024). Instead, they primarily source steel elements from donor structures after receiving assignments from engineering firms. This approach underscores the importance of efficient resource management and the need for a streamlined process to meet project demands.

The current workflow begins when a developer proposes a project, for example a data centre. The initiator then engages an engineering firm and an architect to develop an initial design based on the desired floor plan and power requirements. Once the preliminary design is completed, they approach a steel distribution company to check the design for possibilities to apply reused elements. Who in their turn will start looking for elements that will become available. This establishes a feedback loop between the engineering firm to adapt the design based on the availability of specific steel sections. This iterative process can be time-consuming, as it relies on separate parties who do not have complete insight into the possibilities.

Designing based on cross-sections that are often readily available shortens this procedure. By doing so, engineering firms can streamline the design phase, reducing the time spent on sourcing specific steel elements. The steel distribution company provided a range of cross-sections that are more frequently available (“Voorraad constructiestaal. Interview by Tessel van Oers”, 2024). This data can be combined with the application of basic steel design requirements from the Eurocode, generating a starting point for initial designs. This approach not only accelerates the design process but also ensures that the materials used are readily accessible, promoting efficiency and sustainability in construction projects.

2.4.1. Dataset

In order to effectively find the supply of reclaimed steel elements, Opalis.eu (2024) launched a platform listing all websites that offer reclaimed steel. From this platform, several sources can be found that have a small stock provided by local demolition companies like Snellen (2024), GBW (2024) and van Liempd (2024). These companies typically store a limited range of steel elements at their facilities, which can be accessed for reuse. While smaller in scale, such datasets present a reliable resource for specific projects where immediate availability is a priority. Figure 2.2 shows such a storage facility and a few of their available pieces. Another viable option is leveraging data from buildings which are up for decommissioning, creating an inventory of the used elements as a basis for a new design.

Figure 2.2: Pictures from van Liempd (2024) their available stock of HEA300 and IPE220 beams. As can be seen in the image, the elements are stored outside which will result in the need for a thorough visual check as per NTA8713. For the sake of this research, the quality of the beams is assumed to be fit for reuse, but it is important to note this is not a guarantee with this type of source.

2.5. Summary

This chapter shows the growing importance of reusing steel elements in building construction to reduce carbon emissions and contribute to a circular economy. Reuse of structural elements aligns with the European Commission's climate goals and the new Dutch Technical Agreement (NTA 8713), which outlines guidelines for reusing structural steelwork. This chapter also shows that reuse of steel significantly reduces carbon emissions and can offer economic benefits, as highlighted in several studies. LCA calculations demonstrate the potential for emission reductions across all life cycle stages, particularly when Module D, but Module A as well, is carefully considered. Research shows that incorporating reclaimed steel into designs can reduce emissions by up to 35%, with the most substantial benefits realized during the production stage. However, challenges such as unclear regulations, the labour-intensive nature of deconstruction and coordination barriers limit widespread adoption. Design strategies, including the reuse of building components and designing for deconstruction, offer actionable pathways to overcome these barriers, promoting a circular construction approach. Ultimately, the reuse of steel presents a viable solution for minimizing the environmental footprint of buildings while aligning with sustainable development goals.

3

Automated Design of Data Centres

Due to the increase in electricity demand and internet use, data centres have become more and more important as they power everything from cloud storage to streaming services. As a result, they are projected to be the biggest global energy consumer by 2030 (Jones, 2018). However, designing these critical infrastructures is a complex task that requires careful consideration of various factors, from energy efficiency and cooling systems to security and scalability (50600-2-1, 2021). Despite the complexity, the load-bearing structure of a data centre is relatively straightforward, making it an ideal candidate for automated design.

Automated design processes offer numerous benefits, including increased efficiency, reduced errors and the ability to quickly explore a wide range of design options. This chapter analyses the research on automated design processes, examining their findings and applications. Concluding with an overview of optimization tools available in the visual programming application Grasshopper, which is applied in the final step of reclaimed steel integration as described in chapter 7. This chapter aims to answer the second sub-question:

2. *What stock-constrained automated design processes have been researched, and which apply best within this application?*

3.1. Data Centre Design

In designing data centres, the Dutch norm NEN50600-2-1 (2021) states several key requirements to ensure structural integrity. In the document, guidelines and design recommendations are made for this specific building type, which will be referenced throughout this subchapter. This building type consists of many different functions that help the data centre function normally, each consisting of different loads and floor plan requirements. Within this research, the focus lies on the main hall which houses the data racks and cooling system, leaving out the energy storage, access ways and equipment rooms. This decision is based on the fact that the floor plan layout of the data racks has the most impact on the efficiency of the overall performance of the data centre (Rasmussen and Torell, 2015), as poor layout planning can lead to a decrease of data rack effectiveness of 50% due to inefficient cooling.

One of the recommendations made in the NEN50600-2-1 is to design for future expansion possibilities; the layout of the load-bearing structure should be designed to accommodate additional loads that may arise from future expansions. Another important requirement is to implement floors with a total height of around 4.5 meters to facilitate optimal air circulation and accommodate the installation of necessary equipment like raised floors and suspended ceilings. The Eurocode also recommends that roofs be designed to be able to have solar panels placed on them, promoting the use of renewable energy and

enhancing the data centre's sustainability as it generally is a building with high energy consumption. Lastly, the loads on floors and ceilings should align with the specifications provided in Table 3.1, ensuring these structures can withstand the weight of the equipment and infrastructure housed within the data centre. The specific load combinations used within this research can further be found in Appendix A. These design requirements contribute to the creation of robust, sustainable, and adaptable data centres.

Table 3.1: Floor and ceiling loads for electrical and mechanical spaces, and computer room spaces for data centres according to the Eurocode on information technology - Data centre facilities and infrastructure (50600-2-1, 2021)

		Minimum [a]	Recommended
Floor	Uniform load [b] $[kN/m^2]$	7,2	12,0
	Point load $[kN]$	5,0	7,5
Ceiling	Uniform load [b] $[kN/m^2]$	1,2	2,4

[a] If the minimum requirements are not met, it can be necessary to provide structural measures to distribute the loads.
[b] For inter-floor structures the uniform load for floor and ceiling shall be added.

In order to create the optimal design of the data halls, a basic floor plan layout as seen in Figure 3.1 can be considered. One data rack is typically 0.6 meters wide and 1.2 meters long and a row consisting of those racks is typically spaced 1.2 meters with an overall perimeter clearance of again 1.2 meters to create access ways. Since the data racks have a cold air intake on one side and a hot air exhaust on the other, the rows will create cold- and hot aisles within the floor plans. To optimize the performance of the data centre, it is important to keep these aisles separate, as the cooling system can then work on a more concentrated area. This means creating long rows of at least 5 data racks, no columns in the aisles and symmetry on the floor plan. An overview of the mentioned design criteria can be found in Table 3.2, which are used as a basis for the automated design of data centres in chapter 7.

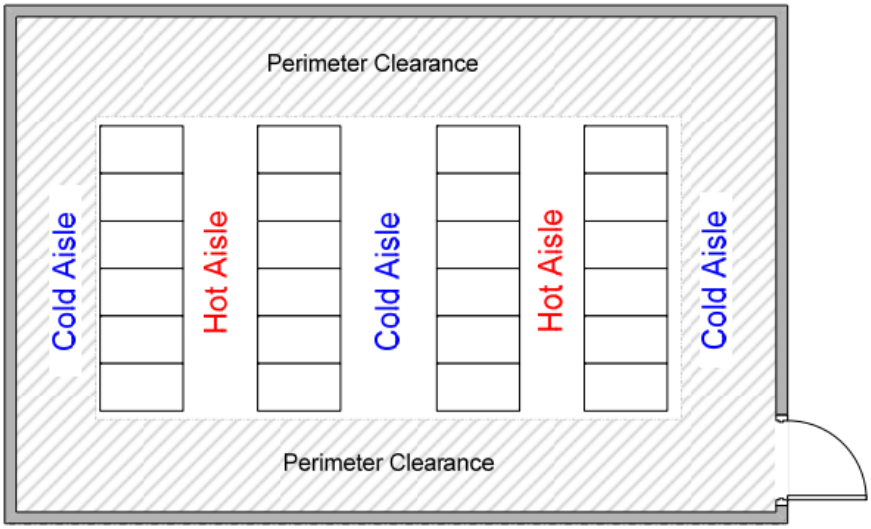


Figure 3.1: A basic double-pitched hot-aisle/cold-aisle layout plan for a data hall determined by Rasmussen and Torell (2015).

Table 3.2: Overview of design aspects to determine the floorplan layout of a data centre. Values come from Rasmussen and Torell (2015) and NEN-EN 50600-2-1 (2021).

Identified design criteria	Value	Unit
Data rack depth	1.2	[m]
Data rack width	0.6	[m]
Data rack height	2.5	[m]
Floor-to-floor clearance	4.5	[m]
Distance between rows	1.2	[m]
Perimeter clearance minimum	1.2	[m]
Perimeter clearance possible hot aisle	1.8	[m]
Minimum row length	5	[-]
Minimum number of rows	2	[-]

3.2. Cross-Section Selection

In the context of automated structural design where the lengths and loads are predefined, the selection of the cross-section becomes a critical task. As the loads of a data centre in the design phase are known and the lengths of each element are set, an initial cross-section calculation can be done according to the Eurocode. The cross-section directly influences the structural performance, including its ability to resist bending, buckling, and shear forces. Three different ways to evaluate cross-section configurations are compared, all considering the same material strength, stiffness, and stability. By effectively determining the optimal cross-section, a design is created that not only meets safety and performance criteria but also minimizes material usage and cost, leading to more sustainable and economical construction practices.

3.2.1. GSA

To generate a robust structural design, structural analysis checks are a necessity as this ensures the feasibility of a structure. One helpful application to apply these checks is the Finite Element software Oasys GSA. Developed within Arup, this program is designed to execute a range of analyses, from static and dynamic to more advanced interaction analyses (Oasys, 2024). Oasys GSA is equipped with multiple design codes, including the Eurocode 3 from 2005 and the Dutch national annex for steel design. In this application specifically, the Steel Designer analysis tool from Grasshopper GSA is utilized. This tool creates an optimum cross-section based on the load, length and location within the structure, as per the Eurocode.

The Steel Designer works on a member-by-member basis. It checks each member for strength and/or serviceability using the specified combination cases. The sections are drawn from the Steel Section Pool that the member refers to. These sections serve as potential candidates for the design. If a member fails a check, the system assigns a fitter section from the pool, thereby creating the optimal design. The steel checks account for combinations of bending, shear, and axial load. Buckling is considered by applying a reduction factor that corresponds to the governing buckling type for that specific member.

Within the scope of this research, only the beam and column elements within the structural design are considered. This creates the need for a floor-loading to load-bearing structure interaction, without additional computing power needed. In Oasys GSA, linear 2D elements can be designated as load panels. Unlike structural elements, these load panels do not contribute to the structure's stiffness or mass. Their primary role is to define loads without material or properties. These loads, defined as face loads on 2D elements, are then distributed to the surrounding beam elements. They are defined by a support type and a reference edge. The support pattern determines the number of edges to which the load is transferred,

while the reference edge dictates the selection of free/loaded edges.

While Oasys GSA offers comprehensive structural analysis and design verification, one notable drawback is its processing speed. The thorough checks it performs, particularly for each individual member under various loading conditions, contribute to a slow response time. This is amplified by the fact that it connects to an external program, adding further delays. In early design stages, when rapid iterations and adjustments are essential, the level of detail provided by GSA may exceed what is necessary, potentially hindering the design workflow by slowing down the process.

3.2.2. Karamba

Karamba3D is a widely used structural analysis and optimization tool that is fully integrated within Grasshopper. It enables engineers and architects to perform structural calculations without leaving the parametric design environment, making it particularly effective for early-stage design explorations. One of Karamba3D's key advantages is its speed. Unlike other external analysis programs, it executes structural analyses directly in Grasshopper, which significantly reduces computation time and allows for rapid iterations (Preisinger, 2013). This makes it ideal for projects where the geometry and loading conditions are still evolving.

Karamba3D's cross-section optimization tool streamlines the process of selecting optimal beam and column profiles. It adjusts the cross-section of elements to meet specific strength and serviceability criteria, ensuring structural efficiency while minimizing material use (Karamba3D, 2024). However, a notable limitation is that the tool can only optimize one cross-section profile type at a time, meaning different types of structural elements, such as beams and columns, must be optimized separately. Despite this constraint, the speed and ease of use make Karamba3D a powerful tool for the initial design stages, where fast feedback is crucial.

3.2.3. Manual Eurocode Checks

To further ensure the structural integrity of the steel hall, cross-section selection can be done manually, based on NEN-EN 1993-1-1:2006 (2020). This method allows for direct control over the calculations, ensuring compliance with design codes while being highly adaptable to specific structural requirements. Cross-section properties, such as axial and bending resistances, are retrieved from standardized profiles from the IPE, HEA and HEB family, as detailed in the Eurocode database EurocodeApplied (2017). These standardized sections provide critical parameters for manual verification.

$$UC = \frac{Resistance}{Force} < 1 \tag{3.1}$$

In this approach, the assumption can be made that the elements of the structure behave as Euler-Bernoulli beams. This assumption simplifies the analysis, as it considers only bending and axial forces, ignoring shear deformations. Given the large slenderness ratios typical of steel structures, this assumption is justified. To calculate the efficiency of a cross-section, Unity Checks need to be done. The general form is seen in equation 3.1. Detailed calculations for the steel-specific design, including checks for buckling and lateral-torsional buckling, are provided in appendix A. When Eurocode calculations are applied, the following checks need to be done:

- Pure bending: Verifies the moment resistance of the section against the applied bending moments.
- Pure axial force: Ensures the axial resistance is sufficient to carry the axial loads.
- Combined axial and bending loads: Assesses the interaction between axial and bending forces using the Eurocode's interaction formulae.

- Buckling resistance: Evaluates both major and minor axis buckling resistances for compression elements, using the reduction factors specified in Eurocode 3.
- Lateral-torsional buckling resistance: Verifies the capacity of beams to resist buckling due to torsion and lateral displacement.
- Deflection check: Ensures that the deflection under applied loads remains within acceptable limits for serviceability.

3.3. Evolutionary optimization

Grasshopper, a visual programming language within Rhino, features a built-in genetic algorithm primarily through its Galapagos component. This algorithm is designed to optimize complex design problems by mimicking the process of natural selection. Users define a fitness function that evaluates the performance of different design iterations based on specified criteria. The genetic algorithm then iteratively evolves the design by selecting, crossing and mutating the best-performing solutions. This process continues until an optimal or satisfactory solution is found. The name "Galapagos" reflects the algorithm's inspiration from evolutionary biology, emphasizing its role in exploring and optimizing design spaces efficiently (Shan, 2014). The Galapagos component uses one of two algorithms to generate random parameter value sets and compare the resulting fitness values to each other. The more variations it tests, the more 'good' solutions Galapagos finds, and it keeps track of the best solutions with the highest fitness in a kind of scoreboard. (Tait, 2024)

Next to the built-in optimizer from Grasshopper, are there additional components which differ in optimization algorithm and thus applicability and computation time. the following other optimization models are considered:

- Goat: This plugin works on the same optimizer as Galapagos, but it adds the possibility to automatically stop the optimization after a certain time period or after the optimization has not improved by a certain threshold. This is a helpful addition in case data needs to be collected automatically, but it does not consider different genetic algorithms than already incorporated in Galapagos. (Rechenraum, 2023)
- Opossum: This plugin includes two of the best-performing, single-objective optimization algorithms in Grasshopper: model-based RBFOpt and evolutionary CMA-ES. It also includes the multi-objective RBFMOpt and NSGA-II algorithms. RBFOpt uses advanced machine learning techniques to find good solutions with a small number of function evaluations, i.e., simulations, while CMA-ES reliably finds near-optimal solutions when many function evaluations are possible. (Wortmann, 2017)
- WallaceiX: This is an evolutionary multi-objective optimization engine that allows users to run evolutionary simulations in Grasshopper 3D. It uses highly detailed analytic tools and various comprehensive selection methods, including algorithmic clustering, to help users better understand their evolutionary runs and make more informed decisions at all stages of their evolutionary simulations. (Makki et al., 2022)

These tools collectively offer varying degrees of control over optimization, each suitable for different stages of the design process. While Galapagos and Goat are suitable for early-stage optimization due to their simplicity, tools like Opossum and WallaceiX provide more advanced capabilities for refining and analysing complex design problems.

3.4. Stock-Constrained Automated Design

Several studies have been conducted regarding stock-constraint automated design, which can be approached in multiple different ways. Bukauskas (2020) set the stage for extending the application of reuse into structural design. They introduced the two new concepts that have been used to optimize designs in an early design stage. First, the concept of "Assignment" is defined in a binary matrix, where values indicate if a component from the design is matched to a stock. The second concept is the "Off-cut Ratio", indicating the needed modification of the stock. The last step is then to approximate the optimal assignment with polynomial-time Heuristics, in order to create the best match within the design.

After these concepts are quantifiable, they can be optimized to create an optimal design based on stock. Brütting et al. (2020) used these definitions to quantify the embodied greenhouse gas emissions related to reuse, which resulted in the following values from their literature review shown in Table 2.2. The automated design was conducted by calculating the moment capacity of the elements in the inventory, linking those to load combinations within a specific design to create the matches using a branch-and-bound global optimization method. This however resulted in a very slow process as the program was required to run again for all different cut-off lengths, as the moment capacity of the elements will change with it.

Van Marcke et al. (2024) proposed a solution to the high computation time by developing a new aggregation engine that combines a generative algorithm with a finite element analysis. Here the design is not optimized but rather the application of the available stock, if separating a longer element into smaller ones would result in a higher percentage of reuse possible. They also looked at the aggregation of trusses, where the design and combination of stock elements are optimized. Haakonsen et al. (2024) then translated this algorithm towards a Grasshopper component in order to use the optimizer already available within the program, creating a visual impression of the process as well. The tool proposes optimal matching in terms of a structure's environmental impact without overloading the user with additional work or excess information. A thorough case study is performed that shows the applicability of the tool in a two-storey public building. While this study focuses on timber, the principles and methodologies could potentially be applied to steel as well.

From a TU Delft master's thesis, Rademaker (2022) provides a method to incorporate reusable elements into the design of a load-bearing structure within a Python model. On the one hand, this study introduces additional steps in the design process, such as checking, testing, and measuring potential reusable elements, and possibly refurbishing them. Additionally, the study presents an optimization problem defined by an objective, variables, and constraints, providing a Python model for stock assignment within object-oriented programming. The results show that the amount of new steel required can be significantly reduced, depending on the constraints and the available stock of reusable elements. However, this could lead to low utilization coefficients and significant changes in the beam configuration. This study offers valuable insights and tools for designing more sustainable load-bearing structures.

These five studies deal with structural design and reuse of materials, all from different perspectives and with different methodologies. They highlight the potential for reusing materials in construction, which can significantly reduce environmental impact. However, combining a differing design size used in the truss topology optimization within Grasshopper on a total building design has not yet been researched. Leaving a good opportunity open to research the automated design of data centres that combines the above-mentioned studies and adds a Data Analysis step in the form of Machine Learning, which will be further discussed in chapter 4, for a complete calculation.

3.5. Summary

This chapter addresses the increasing importance of data centres as global energy consumers and highlights the design challenges associated with their structural systems. Data centres, particularly their main halls housing the data racks, present an ideal case for automated design due to their straightforward load-bearing structures. The chapter outlines critical design criteria derived from Eurocode standards, emphasizing factors such as floor load capacities, column-free aisles, and provisions for future expansions to ensure robust and adaptable facilities.

As a first step for automated design, three cross-section selection methods were reviewed:

1. GSA: A comprehensive tool that ensures compliance with Eurocode 3 by analysing load combinations and selecting optimal cross-sections. While precise, its external processing results in slower response times, making it less ideal for rapid iterations.

2. Karamba3D: Fully integrated into Grasshopper, Karamba3D enables fast early-stage structural optimizations. Despite its inability to optimize multiple section types simultaneously, its speed and usability make it valuable for iterative design processes.

3. Manual Eurocode Checks: These offer direct control and adaptability, leveraging Euler-Bernoulli beam assumptions for simplicity. Unity Checks and detailed verifications ensure compliance, but the method is labour-intensive compared to automated alternatives.

The chapter also explores optimization tools, focusing on evolutionary optimization within Grasshopper, which iteratively optimizes design configurations through natural selection-inspired algorithms. Research done by Haakonsen et al. (2024) showed the possibility of stock-constrained design based on available evolutionary optimizations within Grasshopper, while Rademaker (2022) hinted towards possibilities to apply the optimization within Python.

In conclusion, automated design processes significantly enhance the efficiency and adaptability of data centre structural design. Among the reviewed methods, a combination of Karamba3D and manual Eurocode checks emerges as the most applicable for the initial design stages due to its integration and speed. Combined with evolutionary optimization tools in Grasshopper, it provides a framework well-suited for stock-constrained design, as further applied in Chapter 7.

4

Machine Learning

In the last couple of years, Artificial Intelligence (AI) and Machine Learning (ML) have been increasingly popular in research. In 4 years, the research output containing AI has increased by 58% (Nature, 2023). While it seems like these practices are new, it has been around for quite some time in different forms. Dr.Ir. I. Rocha stated the following in his first lecture of the course Machine Learning and Artificial Intelligence for Engineers: *"Artificial intelligence is the name that popularized Machine Learning, which in their turn popularized statistics. It can all be brought back to the theory of probability."*

In this chapter, the fundamentals of Machine learning are explained and different research done on the application of ML within the design process are explored and compared. By doing so, the third subquestion will be answered:

3. *What machine learning applications that assist in the structural design process have been researched?*

Liao et al. (2024) shows in Figure 4.1 the relation between different types of AI which have been researched to apply within the structural design process. A clear difference between the biologically inspired algorithms as discussed in chapter 3 and ML is shown.

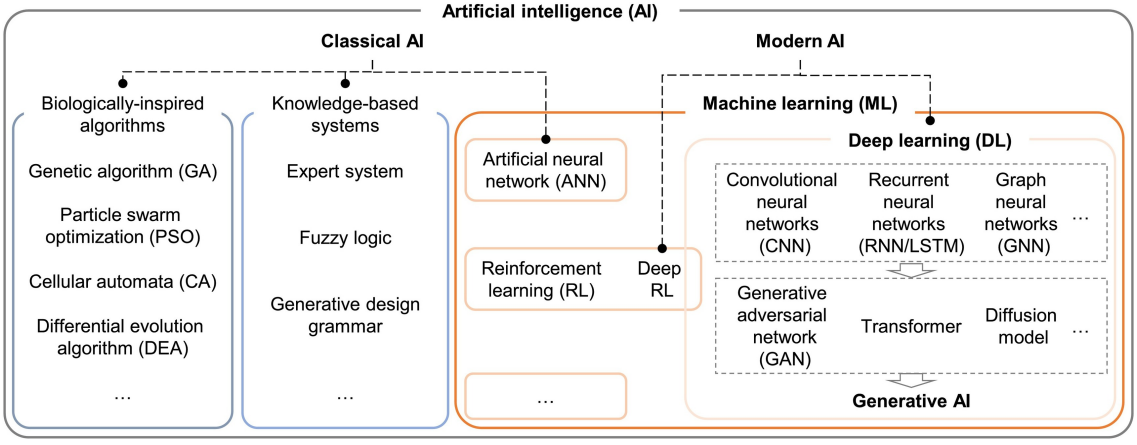


Figure 4.1: Relation of different types of Artificial Intelligence applicable within structural design (Liao et al., 2024)

4.1. Machine Learning Fundamentals

Machine Learning revolves around pattern recognition within data in an automated way. It uses complex algorithms to identify patterns to perform tasks such as classifying data into categories or predicting future outcomes based on previous sequences. An example application of this is recognizing handwritten digits to their value. This is achieved by using a large set of labelled images, known as a training set. The model consists of various activation functions which can be adapted to predict a result which is as close as possible to the actual label. The tuning of these individual parameters is called training the model. Once trained, the model can make predictions on new digit images which were not part of the initial training set. This is an ongoing process, leading to a higher prediction accuracy after more data has come in for the model to 'learn' on (Bishop, 2006).

4.1.1. Types of Training Data

In order to apply real-life data into the activation functions, the original input data is often preprocessed to transform it into a form that is easier for the ML algorithm to work with. This preprocessing stage, sometimes referred to as feature extraction, can involve tasks such as scaling and translating data so that they are all the same size, or computing simple features that are fast to calculate and that preserve useful discriminatory information. A distinction can be made between models which train on labelled data (input to output), unlabelled data (like sequential data for future predictions) or active learning. This leads to the three main categories of machine learning: supervised, unsupervised, and reinforcement learning (Prince, 2023). An example of supervised learning for a classification problem versus unsupervised learning for a clustering problem is shown in Figure 4.2.

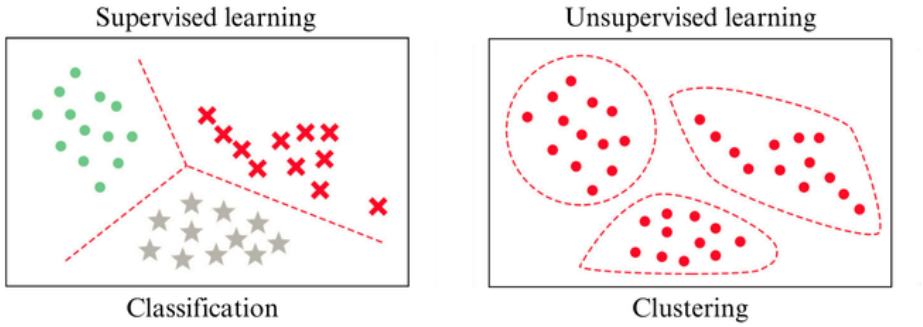


Figure 4.2: Supervised learning is shown with a classification problem where the labels are known beforehand and the boundaries of the classes need to be determined. Unsupervised learning is shown with clustering, where a pattern needs to be identified within unlabelled data. Figure is adapted from Bishop (2006).

Supervised learning models define a mapping from input data to an output prediction. The model is trained on labelled input-output pairs, where the outputs act as a "supervisor" to guide the model in learning the appropriate mapping. This type of machine learning typically requires a training dataset for learning the mapping, a validation dataset for tuning the model's parameters, and a test dataset for evaluating the model's performance. A distinction can be made between a regression problem, where the output is a continuous number, and a classification problem, where the output is a categorical assignment. Some examples are predicting the percentage of reuse possible within a design (regression, prediction of target value) or predicting what energy classification a building will have (classification).

The second type is unsupervised learning, where the goal is to understand the structure of the data rather than learning the output. With this known structure, the clustering of data, the prediction of new sequential data or the generation of text or images can be realized. These models often use latent variables, a smaller number of underlying variables, that can compactly represent the high-dimensional observed data. This concept is applied to weather forecasting and image/text generation.

Lastly, reinforcement learning is the most recent development within this field. It consists of an entity (e.g. a human-like robot or chess-bot) that exists in a world where it can perform actions. These actions change the state of the system and can produce rewards. The goal is for the agent to learn to choose actions that on average lead to high rewards. This involves dealing with the temporal credit assignment problem (associating rewards with the right actions) and the exploration-exploitation trade-off (whether to exploit known good strategies or explore new ones). This concept is subjected to the most resistance within the field of AI as it has the potential to carry harmful biases (Memarian and Doleck, 2024).

4.1.2. Model Validation

Data that is used as a basis of an ML model is often split up into three separate datasets: a training set, a validation set and a test set. These sets are often split for 70%, 20% and 10% respectively of the total, but it depends on the size of the original dataset as there should always be enough training data to get an accurate model (Goodfellow et al., 2016). The training data is used as a basis for the model on which the activation functions are altered to approach the correct result. The validation data is used to tune the hyper-parameters of the model to make sure it is not overfitting on the training data. The quality of the predictions can be monitored by the residuals of the loss function (Prince, 2023).

The trained model can be validated by running the predictions on the unseen test data and comparing the actual result. Within a classification problem, for every label, there are four types of results: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The relationship between these result types is shown in Figure 4.3 and can be used to calculate the Accuracy, Recall, Precision and F1-Score.

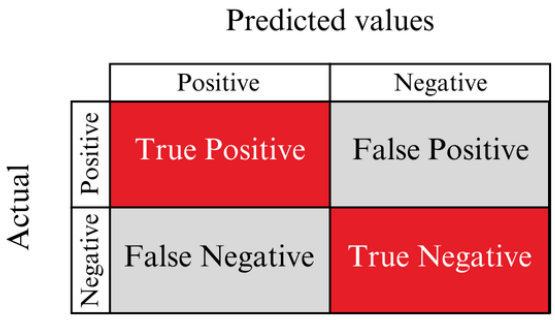


Figure 4.3: Confusion matrix of a binary classification. This visualization shows the relation of model outputs True positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), which are needed to calculate the performance metrics accuracy, precision, recall and the F1 score. Figure is adapted from Jayaswal (2020).

The Accuracy gives a useful overall metric but could be insufficient in uneven spreads of predictions. The Recall is particularly important if the cost of a False Negative is high, whereas the Precision is more important if False Positives are worse. The F1-score creates a balanced summary between the two by combining both the precision, favouring the False Positives, and the recall, which favours the False Negatives. These metrics facilitated iterative improvements, guiding adjustments to both data preparation and model parameters. These values indicate the performance of the predictions and are calculated with the following formula following from Jayaswal (2020):

$$\text{Accuracy} : \frac{TP + TN}{P + N} \tag{4.1}$$

$$\text{Recall} : \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{Precision} : \frac{TP}{TP + FP} \tag{4.3}$$

$$\text{F1} - \text{Score} : \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{4.4}$$

4.2. Recurrent Neural Networks

Within the application of this research, the type of Deep learning that seems the most promising is Recurrent Neural Networks (RNNs). This is explained by Goodfellow et al. (2016) as a specialized type of neural network designed for handling batches of sequential data, such as time series, speech, or text. Or in this case, it is able to learn patterns across different design configurations with multiple inputs. Unlike traditional artificial neural networks, which process data independently, RNNs are built to recognize patterns across sequences by maintaining a form of memory of previous inputs. The differences in network architecture can be found in Figure 4.4. This memory is encoded in the hidden states of the network, which evolve as the network processes each element of the input sequence. RNNs have been successfully applied in many fields, from natural language processing to time series forecasting, making them a powerful tool for tasks involving sequential dependencies.

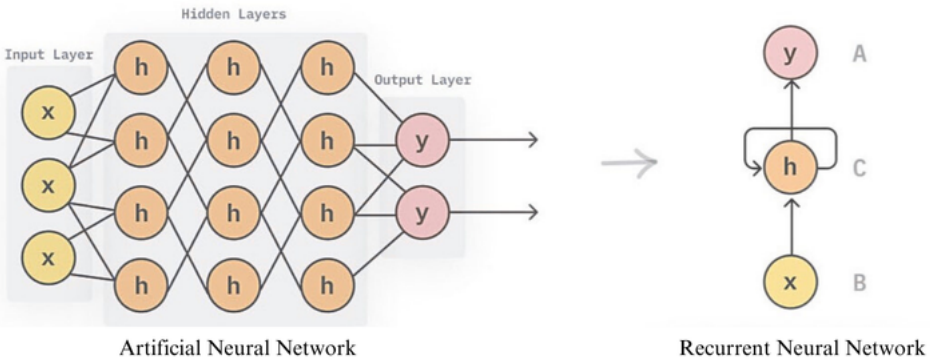


Figure 4.4: Overview of a typical ANN network versus an RNN network made by Baheti (2022). Note that with an RNN, the multiple data inputs are seen as one sequence, creating the possibility of learning based on an entire set of data at once. The RNN also contains a feedback loop which is often done by adding gates. This is further explained in chapter 4.2.2.

4.2.1. RNN Architecture

The core innovation of RNNs is their ability to maintain a hidden state that captures information from previous time steps. This hidden state is passed through each layer of the network along with new input, allowing the RNN to retain a memory of what it has seen before. This property enables RNNs to perform well on problems where the order of inputs matters, such as predicting the next word in a sentence or forecasting future values in a time series. A key characteristic of RNNs is that they use the same set of parameters (shared weights) at every time step, unlike feed-forward networks where different weights are used for each layer. This weight-sharing reduces the complexity of the model and allows RNNs to generalize across sequences of varying lengths.

Despite these advantages, basic RNNs struggle with learning long-term dependencies, as the influence of past inputs diminishes over time. This is known as the vanishing gradient problem, which limits the RNN's ability to remember information from far earlier in the sequence (Goodfellow et al., 2016). To address this issue, more advanced architectures like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) have been developed, which we will discuss in the following sections.

While RNNs are effective for processing sequential data, certain tasks, such as machine translation, require transforming one sequence into another, possibly of different lengths. The Encoder-Decoder architecture was developed for this purpose. In this architecture, the RNN is divided into two parts; an encoder, which processes the input sequence to a context vector in latent space which can be seen as the summary of the sequence. Secondly, there is a decoder which takes this context vector together with previous results to generate a new output. This architecture is particularly effective in tasks like language translation and speech recognition, where the input and output sequences may have different lengths and the goal is to generate a sequence rather than a single output.

Despite the advancements in RNN architectures, like LSTMs and GRUs, which mitigate issues like the vanishing gradient, the problem of exploding gradients can still arise when working with very long sequences or highly non-linear functions. Exploding gradients occur when large derivatives cause the gradients to grow uncontrollably, which can lead to unstable updates during training. To address this, a technique known as gradient clipping is often employed. Gradient clipping prevents these gradients from exceeding a predefined threshold, effectively "clipping" their magnitude to avoid excessively large updates. By scaling down gradients that surpass the threshold, gradient clipping helps maintain stability during training and ensures that the network converges more reliably (Goodfellow et al., 2016). This technique is especially useful in applications where the sequence lengths or the nonlinearity of the network can otherwise lead to numerical instability.

4.2.2. Gated RNN

While basic RNNs excel at handling short-term dependencies, their performance degrades on longer sequences due to the vanishing gradient problem. This is where Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) come in. These are forms of gated RNNs that incorporate mechanisms to control the flow of information through the network, ensuring that important information is retained over long sequences. A visual representation of the differences in components can be seen in Figure 4.5, adapted from Rathor (2018).

- LSTM: LSTMs introduce three gates (input, forget, and output gates) that regulate the flow of information. The input gate controls how much new information should be added to the memory, the forget gate determines what information should be discarded, and the output gate decides how much of the memory should be used to generate the output. This gating mechanism allows LSTMs to remember important information for much longer sequences compared to standard RNNs, making them ideal for tasks that require learning long-term dependencies, such as speech recognition or time series analysis.
- GRU: GRUs simplify the LSTM architecture by combining the forget and input gates into a single gate, reducing the complexity of the model while still retaining the benefits of gating mechanisms. GRUs tend to perform similarly to LSTMs but are computationally more efficient, which makes them a popular choice in scenarios where training speed is a concern.

Both LSTMs and GRUs maintain a more stable gradient flow during training, enabling them to learn better from longer sequences. This makes them highly suitable for structural design applications, where temporal dependencies (such as the sequential arrangement of elements or the incremental progression of a design) can play a critical role.

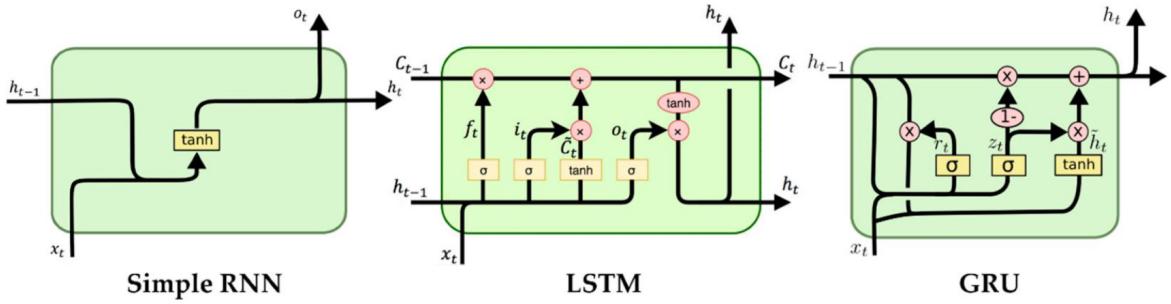


Figure 4.5: Basic architecture overview of a component in a simple RNN, LSTM and a GRU layer. The simple RNN consists of an input and output, passed through a $tanh$ activation function. The LSTM component introduces a forget- and update-gate, where additional information is propagated to the following layer. The GRU combines the two gates and simplifies the output into one stream (Rathor, 2018).

4.3. Application within Structural Design

AI has increasingly been recognized as a promising tool within the structural design process. Liao et al. (2024) created a literature study on the current state of research in this domain, emphasizing that the vast amount of architectural, structural, and empirical data available is often not used to its full potential. Their review identifies generative adversarial networks (GANs) and graph neural networks (GNNs) as the most promising technologies for optimizing component layouts and cross-sectional sizes in building structures. They argue that the intelligence of structural design must evolve through methods that effectively learn from available data, thereby enhancing efficiency and providing diverse design solutions.

4.3.1. Application Examples

The following sub-chapter will explain four other ML concepts that are promising for incorporating donor elements automatically. The approaches aim to simplify the design process by taking part in design decisions at early stages.

Pizarro and Massone (2021) developed a deep neural network (DNN) to predict the length and thickness of reinforced concrete shear walls. Their study used a dataset derived from architectural and structural floor plans of 165 Chilean residential layouts, which featured 30 unique design parameters per shear wall, such as length, height, and surrounding floor area. After normalizing the data, the DNN transformed these input features into outputs specifying the structural dimensions. With a coefficient of determination (R^2) of 0.995, the model demonstrated remarkable accuracy in deriving structural designs directly from architectural inputs. However, a limitation of this approach lies in its reliance on existing floor plans, which may not always be available.

In contrast, Potijk (2024) implemented a simplified artificial neural network (ANN) to estimate building material mass, linking it to both structural and environmental costs. Using a Grasshopper script, the study analysed various building envelopes based on finite element method (FEM) calculations. Since this script was set up for two separate stability systems, the opportunity to compare the two based on the respective structural and environmental costs presented itself. As this ML concept is not considered Deep Learning, it does not capture as many latent variables as the previously mentioned DNN. This turns out to be not necessary for this application as a sufficient comparison could be made, resulting in a much faster model.

Mirra and Pugnale (2023) explored reinforcement learning (RL) for structural design optimization, focusing on the adaptive design of tension structures. They demonstrated how RL agents could effectively

navigate complex design spaces, balancing competing objectives such as structural performance and material efficiency. By applying iterative feedback loops, the RL framework enhanced the adaptability of the design process, making it particularly suited for optimizing structures with highly variable boundary conditions.

Bleker et al. (2024) extended the application of reinforcement learning by integrating deep reinforcement learning (DRL) with graph neural networks (GNNs). Their research tackled the challenge of optimizing large-scale structural systems by leveraging the GNN's ability to represent relationships between structural components. The DRL approach allowed the model to iteratively improve structural configurations, ensuring optimal load distribution and material usage. This combined framework proved especially valuable for designing complex systems with interconnected elements, such as bridges or high-rise structures.

4.3.2. AI Applications for other design aspects

In line with the evolving focus on sustainability in building design, the application of machine learning is extending into LCA calculations. As highlighted by Fenton et al. (2024), early-stage machine learning predictions for embodied GHG emissions can significantly enhance sustainable design by providing a quick insights into the carbon impact of material choices and configurations. By applying such models, designers can integrate environmental considerations earlier in the design process, potentially leading to more informed and lower-impact structural decisions. This approach complements the structural optimization methods discussed above, further broadening the scope of AI applications available to structural engineers and architects (Fenton et al., 2024).

AI is also being applied to improve ecological areas in urban and architectural design. Mirra et al. (2022) explored the use of neural networks for analysing human-made habitat structures to enhance their integration with natural ecosystems. By evaluating parameters such as material composition, geometry, and spatial arrangement, the study developed predictive models that informed designs supporting biodiversity and ecological balance. This represents an interesting new perspective in AI applications, as it is essentially used to assist in creativity.

A big direction AI has demonstrated potential in is data-driven design to again, aid the creative design process. Fuhrimann et al., 2018 investigated data-driven methods to inspire innovative structural forms, inserting AI to generate and refine free-form shapes based on aspects like material usage or structural stability. By acting as a creative collaborator, machine learning tools provided designers with new ideas while ensuring feasibility. This highlights AI's capacity to assist not only in the structural but also the artistic design.

4.3.3. Possibilities for RNN Application

Recurrent Neural Networks (RNNs), especially gated architectures like LSTMs and GRUs, offer significant advantages in the field of structural design. Many design tasks involve sequences such as the step-by-step optimization of building components or the gradual refinement of load-bearing structures. RNNs can model these sequential dependencies effectively, making them suitable for predicting the progression of a structural design or optimizing layouts based on previous design states.

Additionally, in the context of reusing structural elements, where multiple configurations of elements and their interactions over time must be considered, the ability of RNNs to capture long-term dependencies allows for better decision-making. The RNN can be applied to label elements with their respective cross-section size, turning the model into a multi-class classification problem.

4.4. Summary

In summary, the growing integration of Machine Learning into structural design presents an exciting development for optimizing design processes and improving sustainability. From supervised learning models that predict material reuse potential to advanced Recurrent Neural Networks capable of modelling sequential dependencies in design, AI offers powerful tools for modern engineering challenges. In the situation of this research, the application of gated RNNs, such as LSTMs and GRUs, holds promise for tasks that involve complex temporal relationships, such as progressive design optimizations or the incorporation of reused materials. This stems from the fact that RNNs handle temporal sequential data, but also in the application of design configurations, making them a promising candidate for cross-section predictions. As more data becomes available and as AI techniques continue to evolve, the potential for creating more efficient, environmentally friendly designs will only expand, highlighting the necessity for further exploration in this field.

II

Application

5

Methodology

In the following chapter, the overall workflow answering the main research question is explained:

"How can the incorporation of reclaimed steel elements be made more accessible within the design process of datacentres when using Machine Learning applications?"

The methodology that is applied in this research is focused on creating a workflow to incorporate reclaimed steel early on in the design of data centres, by using a combination of machine learning techniques and computational design. The goal is to develop an optimized structural design that minimizes embodied carbon while maximizing the reuse of reclaimed steel elements. Figure 5.1 shows the steps taken to come to the final result, split up into four different groups. The following chapter delves into the initial design set-up, data gathering and the reclaimed steel element databases. The RNN model is explained in chapter 6 and the reclaimed steel integration is further discussed in chapter 7.

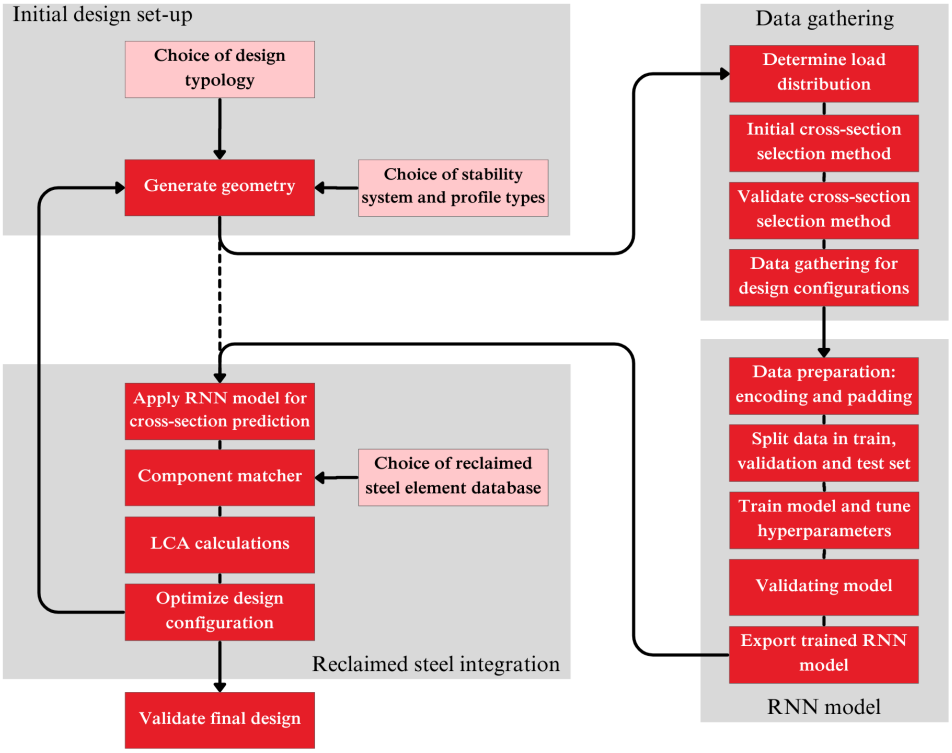


Figure 5.1: Workflow diagram to answer the main research question.

5.1. Initial Design Set-up

The first stage of the workflow is choosing a building typology, which creates a list of design requirements. In this research, only a data hall, a function within a data centre building, is considered. This leads to specific design requirements as the floorplan layout influences the efficiency of the necessary cooling system. The number of rows and the number of server racks per row together with the location of the columns in the floorplan determine the initial design. Further design requirements are discussed in chapter 3.1 and Table 3.2. By standardizing the data centre layout, the methodology ensures that the generated designs adhere to industry specifications and are directly applicable to practical scenarios.

The range that is considered for the number of rows is set from 2 to 7 rows, resulting in possible widths of 7.2 to 19.2 meters. The range that is considered for the number of racks per row is 11 to 29, resulting in an overall floorplan length of 9 to 19.8 meters. After the basic floorplan layout is set up, the location of the columns can be determined in the form of the number of rows and the data rack the column is located. This is based on the fact that columns should not be in the aisle between the racks (Rasmussen and Torell, 2015). The columns can be placed in any row, but the minimum row length of 5 racks needs to be followed. On the other hand, the length should not exceed the threshold for maximum cross-sections, which is equal to a maximum of 10. This results in a total of $6 \cdot 19 = 114$ different floorplan sizes being considered. When combining this with the different grid spaces possible, a total of 40824 different floorplan configurations can be generated. Three examples of the generated floorplans can be found in Figure 5.2. Note that these floorplans show the size boundaries, but all combinations of floorplan width and height in between are considered as well.

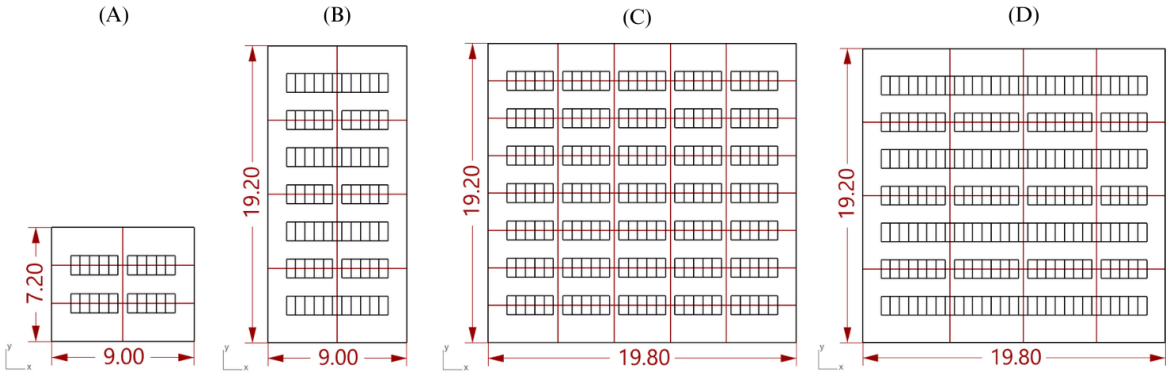


Figure 5.2: Overview of some floorplan layout options with grid sizes, all values show the lengths in meters. (A) shows the smallest floorplan with the specified ranges and the smallest grid space. Floorplan (B) indicates that all combinations of lengths and widths are considered, not just square configurations. (C) shows the biggest floorplan with still the smallest grid space. (D) shows again the biggest floorplan but now with the biggest grid spaces considered.

The next step necessary to generate an initial design is to choose the profile types and stability system. The choice of stability system impacts the moment distribution and the buckling length of the columns, creating different section profile sizes for each element. The two stability systems considered are a braced frame and a moment-fixed frame. A braced frame structure consists of simply supported beams with hinged connections to the columns. The stability is created from diagonal braces that only support axial forces, the system has no moment in the corners and a buckling length equal to the column height. A moment-fixed frame creates stability by fixing element rotations in the connections, creating lower moments at midspan, higher at the corners and a buckling length of twice the column height. Next to these stability systems, two profile-type combinations are considered within this research. One configuration has IPE beams and HEB columns, and one configuration has all elements of type HEA. These design choice options are visualised in Figure 5.3 and result in 4 combinations, which are abbreviated to IPEm, IPEb, HEAm and HEAb.

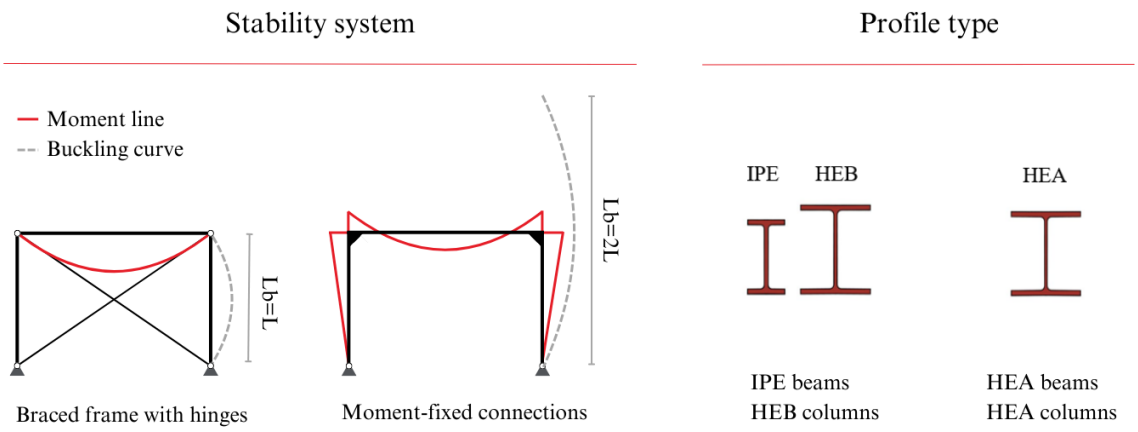


Figure 5.3: Visualisation of the two stability systems considered within the research. As well as the two combinations of section profile types applied. These combinations result in four different design choice configurations abbreviated by IPEm, IPEb, HEAm and HEAb.

After the choice of stability system is made and the profile types for the beams and columns are determined, the initial design can be generated. Within this research, a design of two floors is chosen to better visualise the load transfer of the server racks, assumed to be uniform over the floor surface, to the columns. Since these loads are relatively high, a two-way spanning floor system is chosen. This results in lower, triangular-distributed loads on the beams which ultimately leads to lower maximum moments (Dolan and Hamilton, 2019). The resulting initial design can be seen in Figure 5.4. The next step is to optimize the cross-sections to match the load pattern of this specific building type. These checks are done with ULS loading calculations as shown in appendix A, and tested following Eurocode calculations as further described in chapter 5.2.

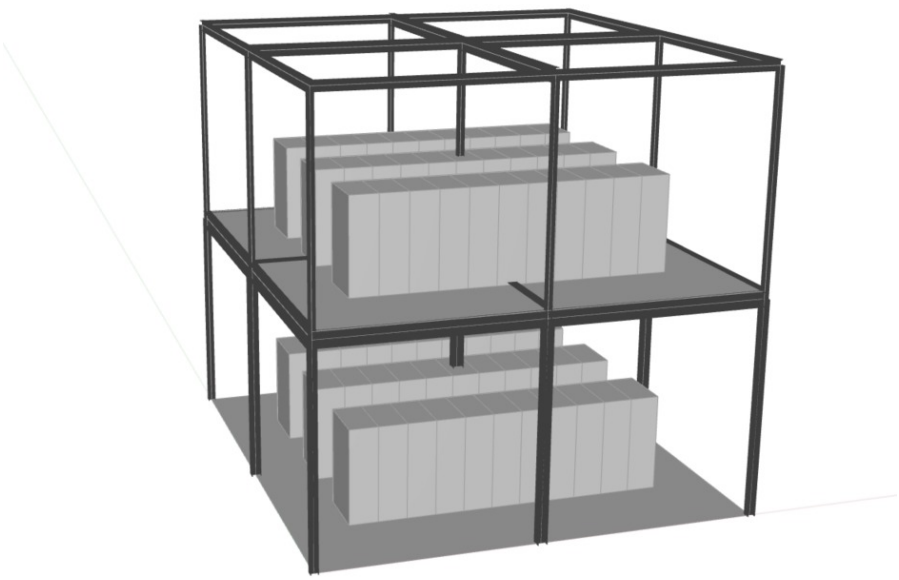


Figure 5.4: Render of the initial design set-up for a floor plan design consisting of three rows and eleven data racks per row

5.2. Data Gathering

This subchapter outlines the cross-section optimization process for the design of a data centre structure, combining design tool Karamba3D together with Eurocode-based unity checks from appendix A, applied by using the Python script from appendix C. The primary aim of the machine learning model was to develop a flexible, automated tool that selects optimal cross-sections for structural elements (beams and columns) based on moment, axial, and buckling resistances, tailored to each design configuration. In order to realize this model, data needs to be gathered which can act as a basis. The section begins with an overview of the Eurocode checks and proceeds to explain the dataset preparation, followed by the generation of cross-section data and design configurations. The Eurocode provides a comprehensive framework for structural design, incorporating various load scenarios and structural checks. These checks are implemented within a Python script as seen in appendix C, where the Eurocode-based resistance values are calculated and compared against applied forces and moments. These results are then validated with GSA as a sanity check. To ensure structural safety and serviceability, the following checks (detailed in appendix A) were incorporated into the Python script:

1. **Bending Resistance**: This evaluates whether the section can withstand applied moments without yielding. The script uses moment resistance values from Eurocode to compare with the calculated moments in each element. [1993-1-1:2006 (2020) 6.2.5]

2. **Axial Force Resistance**: Checks whether the section can resist applied axial forces. This is essential for columns and beams experiencing significant axial loading. [1993-1-1:2006 (2020) 6.2.4]

3. **Combination of Axial and Bending**: For elements subjected to both axial forces and bending moments, the Eurocode provides interaction equations to verify the combined effects. [1993-1-1:2006 (2020) 6.2.9]

4. **(Lateral Torsional) Buckling Resistance**: Buckling occurs due to compressive forces, and lateral torsional buckling is particularly relevant for beams under bending. The Python script calculates buckling resistances based on element length, section properties, and boundary conditions. [1993-1-1:2006 (2020) 6.3.1 and 6.3.2]

5. **Deflection**: The serviceability limit for deflection is checked to ensure that the deformations do not exceed allowable values. [1990:2021 (2021) Table A.1.10]

One key challenge in automating cross-section selection was the varying lengths of the dataset for different grid spaces. For example, a structure with a 2x3 grid configuration would have fewer elements compared to one with a 4x6 configuration. To handle this variability, the element IDs were normalized, allowing the script to consistently address elements across different configurations. This step ensured that the selection process was adaptable to any grid size or structure geometry.

The next step involved generating datasets using a Grasshopper script. This script calculated cross-sections for various configurations of a data centre floorplan. For each configuration, the following data was collected:

- Element ID: A unique identifier for each element in the structure, normalized to be between 0 and 1. This value can be used in the Machine learning model to identify separate design configurations.

- Element type: Distinctions were made between different types of elements based on their orientation, location and structural function. The beams are identified with the labels 'B1x', 'B1y', 'B2x' and 'B2y', which separate the beams on the first floor (1) and on the roof (2) together with the span direction. The columns are identified with the labels 'C1e', 'C1m', 'C2e' and 'C2m',

which separate the columns at the edge of the structure to the middle columns together with the level they are located.

- Element length: The lengths of the element have the biggest impact on the load distribution, as longer spans carry a higher percentage of the floor-surface loads.
- Cross-section Calculation: Cross-sections were determined for each element based on four scenarios, some configurations of length combined with the high loads resulted in a non-applicable (N.A) element as there is no profile which could sustain the configuration, the spans and loads were too high for possible cross-sections. The design configurations and their respective percentage of N.A elements are as follows:

    1. IPEm: Moment-tight connections, IPE beams and HEB columns. 0.5% N.A
    2. HEAm: Moment-tight connections, HEA cross-sections. 0.02% N.A
    3. IPEb: Hinges in a braced structure, IPE beams and HEB columns. 5% N.A
    4. HEAb: Hinges in a braced structure, HEA cross-sections. 0.05% N.A

The resulting data is a 3D array which includes the three input variables that separate the different elements together with the four different cross-sections linked to the four design choice combinations. This method represents an efficient and systematic approach to selecting cross-sections for data centre structures using a combination of Karamba3D and Eurocode standards. Table 5.1 shows the dataset sizes which are generated together with their respective generation time. Note that for this research, 12% of all possible floorplan configurations are used to train the RNN model of chapter 6.

Table 5.1: Dataset sizes that are generated to be used as a basis for the machine learning model. The input is given in the number of rows and the number of racks per row. The sizes are the number of designs it has generated times the length of the element index sequence. Note that this length is all adapted to be the same length as the shorter configurations are padded with the value minus one to still be the same length.

Dataset input [row x racks]	Size	Generation time [hours]
2x11 + 3x13 + 4x15 + 4x21	205x294x7	1
3x20 + 5x16	171x294x7	0.5
2x29 + 7x11	339x294x7	1.5
5x23	338x294x7	1.5
6x26	1134x294x7	4.5
7x29	2686x294x7	6
Total design configurations	4873	

5.3. Reclaimed Steel Databases

This sub-chapter focuses on the different databases of reclaimed steel elements that were used for incorporating reused steel in the design process of data centres. Three different types of databases were explored: open-source databases, a database from realized structures, and a custom-generated dataset based on common lengths and Eurocode standards. Each dataset had unique characteristics, offering a range of element sizes, lengths, and cross-sections, all of which were evaluated to determine their suitability for integration into the design process. Figures illustrating the spread of cross-sections and element lengths for each dataset are provided, with further details on the generated dataset can be found in appendix B.

The first dataset consists of a collection of open-source resources, which compile available reclaimed steel elements from various projects and demolition companies. These databases were sourced from the platform Opalis.eu (2024), which is a collection of small-scale storage locations that house reclaimed steel elements. The elements within these databases vary widely in terms of cross-section profiles, lengths, and conditions. The open-source nature of this database makes it highly accessible, offering a broad range of options that can be quickly matched with new designs. The downside of this data is that they tend to have a high degree of variability in terms of element availability and quality. Some databases may include elements that are outdated or damaged, which can limit their applicability in structural design. For the sake of this research, it is assumed that the quality of the available steel elements is sufficient and that differences in transportation distance are not an added environmental load. The following three sources are used to generate the database: GBW (2024), Snellen (2024) and van Liempd (2024). An overview of the content of this combined database can be seen in Figure 5.5.

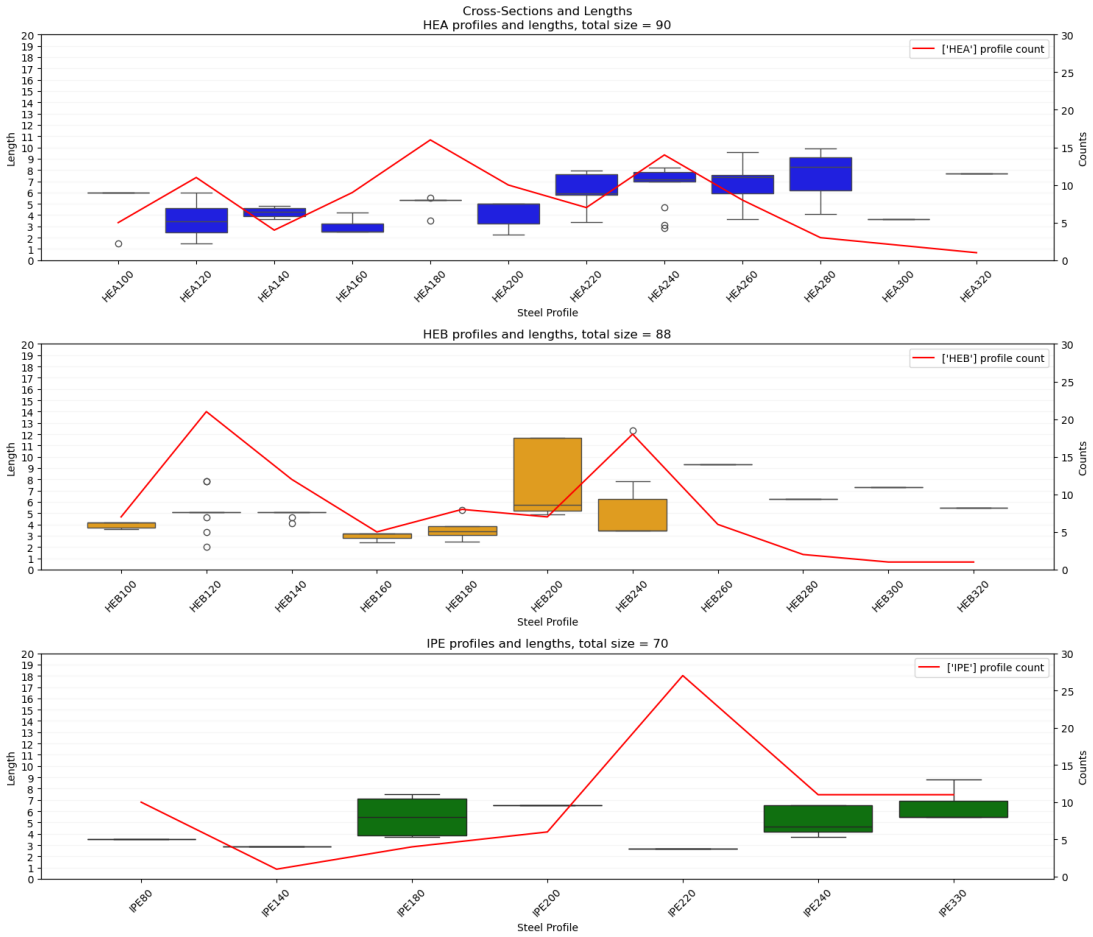


Figure 5.5: Overview of available steel elements combined from three sources. The box plots show the spread of the lengths per cross-section in the data and the red line shows the number of elements present in the database per cross-section.

The second database is based on the idea of utilizing realized structures as input for reclaimed steel that are either up for decommission or expected to be decommissioned in the future. The key idea behind this database is that steel elements from these structures can be reused in new designs after deconstruction. By cataloguing steel elements from existing buildings, this database aims to provide a realistic stock of reusable steel for upcoming construction projects. The advantage of this method is that the structural integrity of the elements is known and designed to current engineering standards, making them suitable for reuse. A challenge with this approach is that retrieving all steel from a design can be expensive when

it is not designed for deconstruction. For this research, the database consists of two separate designs realized by Arup, one data centre consisting of over 80% steel and one residential high-rise building which only housed small amounts of steel elements. Since the confidentiality requirements of the data centre design are high, the added elements from the residential building act as a diffuser, as to not be able to identify what element is from which. The residential building was finalized in 2023, whereas the data centre has just started construction. From these designs, the steel elements are retrieved with details on their cross-section profiles, length and current structural usage. Figure 5.6 shows an overview of the amount and lengths per cross-section present in the database. Note that the y-axis on element counts is not consistent as the number of elements per profile type differs a lot.

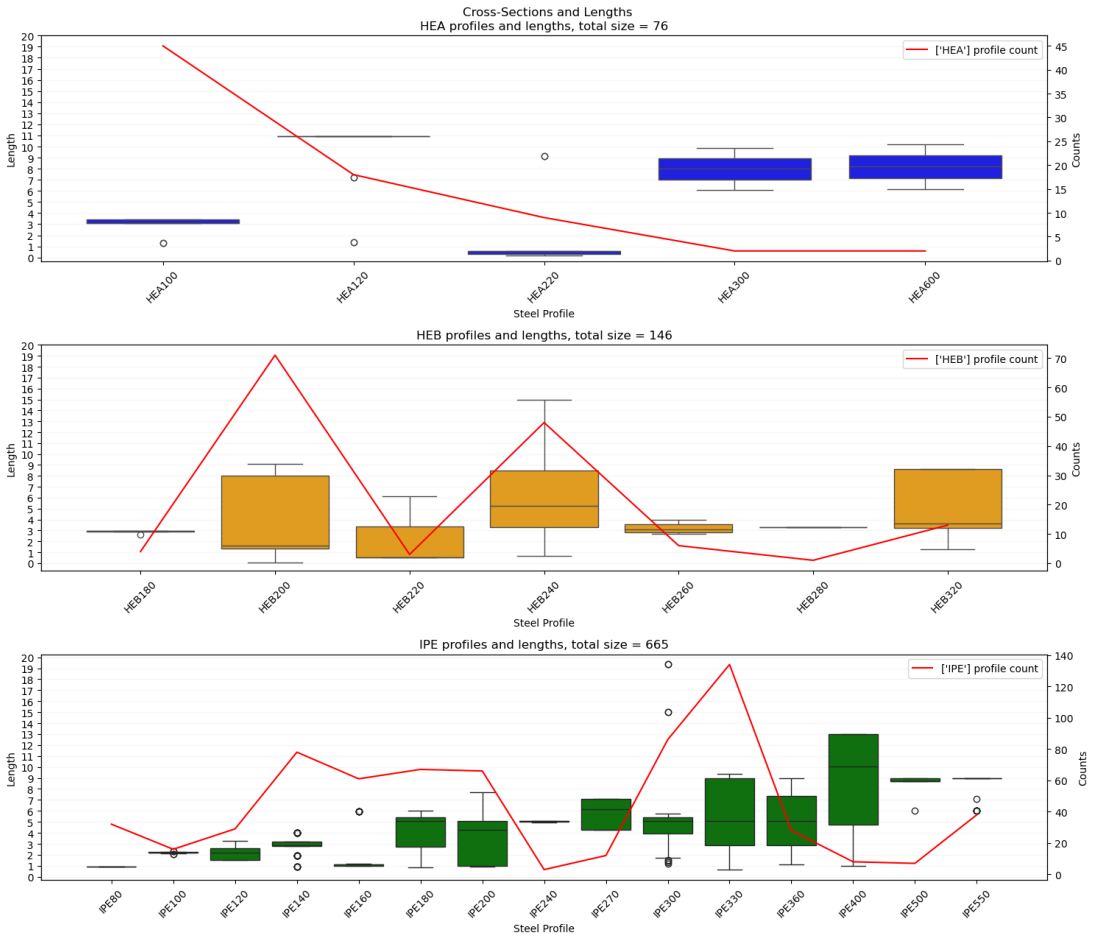


Figure 5.6: Overview of available steel elements combined from two Arup designs. The box plots show the spread of the lengths per cross-section in the data and the red line shows the number of elements present in the database per cross-section. Note that the axis showing the counts is not consistent as the differences are relatively big.

The third database was generated using a Python script (which can be found in appendix B) based on common steel profiles and lengths specified by Eurocode standards. This database serves as a theoretical stock of steel elements, including standard lengths and cross-sections that are frequently used in structural design. By generating this database, the research aimed to create a baseline against which the availability and suitability of reclaimed steel could be evaluated. This database is predictable and based on standard design practices. It allows for detailed analysis and testing of design configurations, particularly for scenarios where the availability of reclaimed steel elements is unknown. It is important to keep in mind that the database is artificially generated, creating a basis for a standardized design, but it does not represent actual availability. The overview of the data can be seen in Figure 5.7.

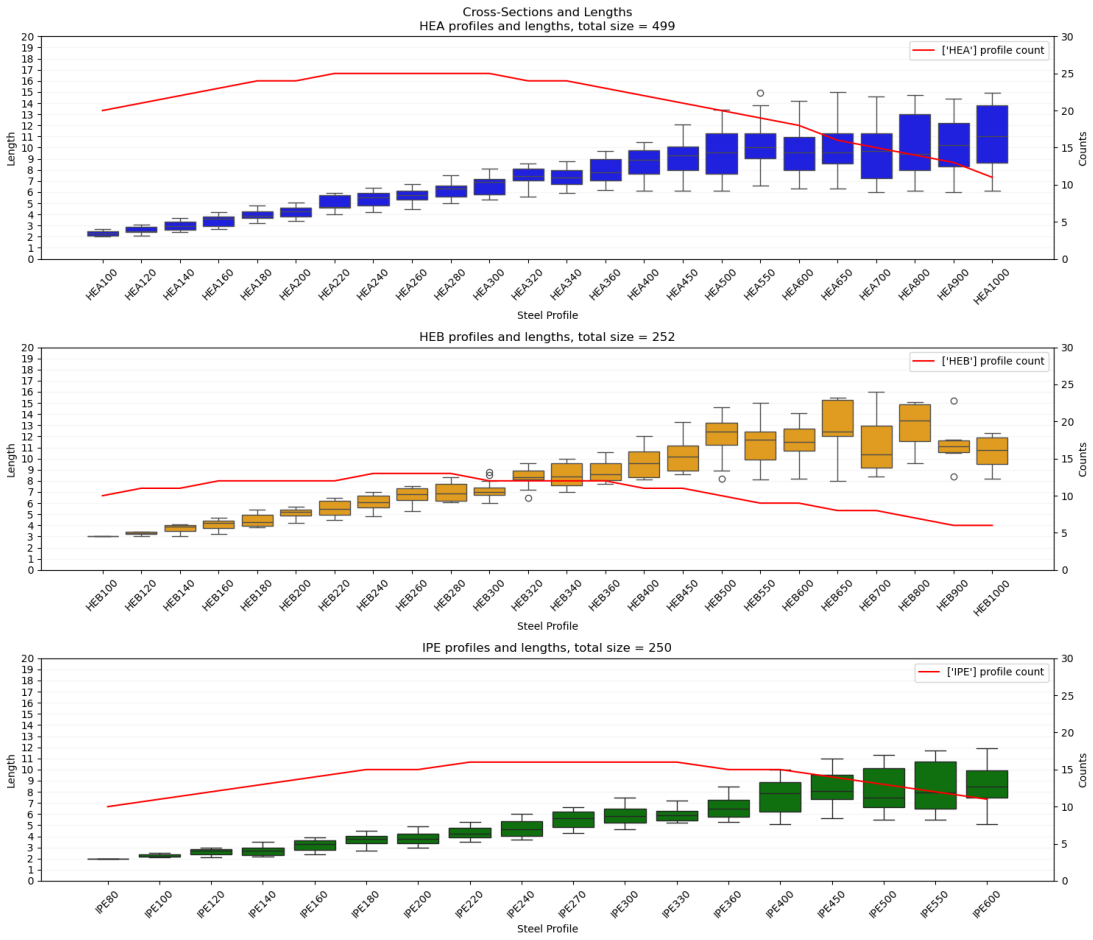


Figure 5.7: Overview of the generated available steel database, based on Eurocode design rules-of-thumb, the code that generates the database can be found in appendix B. The box plots show the spread of the lengths per cross-section in the data and the red line shows the number of elements present in the database per cross-section.

5.4. Summary

This chapter highlights the approach taken to answer the main research question. The approach integrates machine learning and computational design to incorporate reclaimed steel into the early design stages of data centres, balancing low carbon emissions with structural efficiency.

The first stage defines the initial design set-up of data centres, such as layout and structural geometry, based on industry standards for practical relevance. The design model, a two-storey structure, accommodates heavy floor loads typical in data centres, with load scenarios tested according to Eurocode standards. The next stage focuses on gathering data which is used as a basis for the machine learning model, where the element characteristics are combined with the optimized cross-section for four design choice combinations. The data is used to set up a machine learning model that can predict the cross-sections for those four design choice combinations and is further discussed in chapter 6. The predicted cross-sections are then tested on three realistic inputs for reclaimed steel elements databases: An open-source database, a database consisting of existing designs and a generated database based on steel design standards from the Eurocode. In the final optimization phase, the Grasshopper model seeks a configuration with minimal embodied carbon, adjusting parameters like grid spacing and column positioning to align the machine learning predictions with available steel stock. This automated approach to integrating reclaimed steel provides a foundation for sustainable data centre designs.

6

Machine Learning Model

In this chapter, an answer to the fourth research sub-question is presented:

4. *How can machine learning techniques be applied to effectively incorporate reclaimed steel within the design process?*

This research presents a Recurrent Neural Network model that predicts the optimized cross-sections for structural elements in a data centre design. The goal of this model is to automate the selection of steel profiles based on a variety of input parameters, including grid configuration, element types and lengths. By doing so, a proof of concept is shown to streamline the design process, ultimately used to integrate sustainable construction principles, as further explained in chapter 7.

The model employs a supervised learning approach, using a neural network architecture that was fine-tuned to handle variable input lengths and diverse element types. The following sections outline the data preparation, network architecture, training process and post-processing steps required to achieve accurate cross-section predictions. The code which generates this model can be found in appendix D.

6.1. Data Preparation

Effective data preparation is essential for achieving reliable results in machine learning, especially in applications with varying input lengths and different element types. The process involved four main steps: normalization, padding, one-hot encoding and dataset splitting. a visual representation of the one-hot encoding process can be seen in Figure 6.1.

Figure 6.1: Visual indication of the process of one-hot encoding labelled data. The list of profiles in the training data is transposed to have a possibility per possible profile. Since the output is known in the training data, the value for the label which is present is set to one. In this way, the model can assign prediction values per label when it is run on unseen data. The label with the highest possibility is chosen for the prediction.

Neural networks perform optimally when input data is normalized. This process scales features to a range of 0 to 1, ensuring that all input features contribute equally to the learning process. Normalization helps in accelerating convergence of the gradient descent algorithm and prevents the model from being biased towards features with larger magnitudes (Prince, 2023). This normalization is applied to the `element_ID` data, which indicates the location of the element within the design. This normalization also helps identify what sequence represents one design. Another normalization step includes one-hot encoding of string data types. Since the `Element_type` and `Profile` data are in strings format, the data needs to be transformed to a numeric result indicating the different labels available. After training is done, the results would need to be decoded again to generate the output back to the cross-section profile.

The next step is to pad the sequences to be the same length. When the spaces between columns enlarge, fewer columns will fit in the floor plan. This creates a shorter list length necessary to indicate the grid spaces. To accommodate this difference, the lists are padded with zeroes until it has the maximum list length. Note that the location of the zeroes does not change the end values as it generates the same floor plan layout. After loading in the data as one 2-dimensional data frame, the data is transformed into a 3-dimensional tensor with every design configuration as a new layer and padded to be the same length. The total dataset consists of 5125 design configurations. Figure 6.2 shows a representation of the 3D tensor.

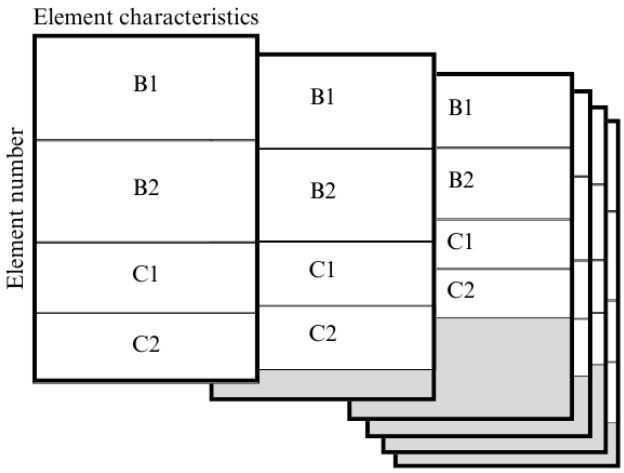


Figure 6.2: Representation of the dataset used as input for the RNN. The squares indicate different design configurations with at the top the four element characteristics: the ID, element type, length and profile. The side represents the length of each configuration by the number of elements. The grey squares indicate the zero-padding done to make configurations with fewer elements of the same length. In the square, a division has been made, indicating the beams on the first floor (B1), the beams on the roof (B2) and the columns on the ground floor (C1) and on the first floor (C2).

Lastly, the dataset was split into training (80%) validation (15%), and test sets (5%) to enable effective training, hyper-parameter tuning and unbiased model evaluation. This split helps to monitor over-fitting and allows the model's performance to be objectively assessed on unseen data. a relatively high percentage of training data is chosen compared to what is discussed in chapter 4, as the dataset would otherwise be too small to use as a basis for the model.

6.2. RNN Architecture

As discussed in chapter 4, the Recurrent Neural Network architecture was chosen and adjusted based on the specific characteristics of this structural design problem. The following sections detail the network architecture, compilation and training of the model. The graphical representation of the specific architecture used within this research can be seen in Figure 6.3. This architecture was specifically chosen to balance the need for sequential context, computational efficiency, and predictive accuracy, making it well-suited for the prediction of cross-sections for an entire design at the same time.

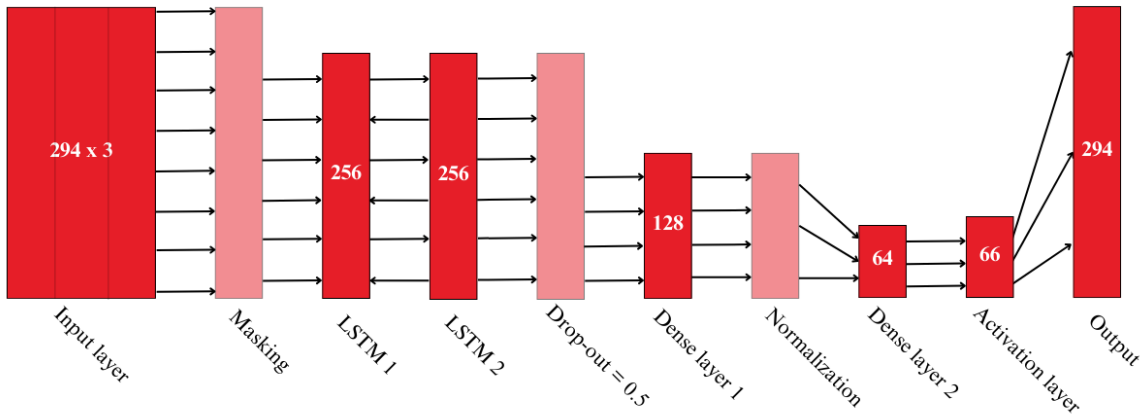


Figure 6.3: Graphical overview of the RNN architecture used within this research. This input of the model is the padded list of elements with the element characteristics. The first input layer of the model has the same size as the input, so `294x3`. Then a masking layer is applied so the model is alerted to what the padded values are. Then two consecutive bi-directional LSTM layers are applied to find the hidden correlation of the values. A collection of duplicates is removed in the drop-out layer before two sense neural network layers are applied with a ReLu activation function to map the data onto the correct length. The activation layer consists of a SoftMax function and is the same length as there are options of profiles to choose from, which are IPE, HEA or HEB profiles. This is then translated to the output layer, which is the padded list of element profiles.

The RNN model architecture used for this application consists of a series of specialized layers to process sequential data, particularly to handle dependencies across variable-length sequences. First, an input layer receives preprocessed sequences of elements, including IDs, one-hot encoded element types, and associated lengths. The model starts with a Masking layer that ignores padded values (-1), ensuring that non-informative elements do not impact model training or prediction. The Bidirectional LSTM (Long Short-Term Memory) layers form the core of the model, with two bidirectional layers each consisting of 128 units to effectively capture both past and future contextual relationships within each sequence. The bi-directionality of the LSTMs helps by accounting for dependencies in both temporal directions, enhancing prediction accuracy in structural configurations where sequential relationships might not strictly follow a single direction (Goodfellow et al., 2016).

A Dropout layer is included to reduce over-fitting by randomly setting a fraction of the LSTM outputs to zero during training, helping the model generalize better to new data. Following the LSTMs, a Dense layer with 64 units and a ReLU activation is applied, providing additional capacity to capture complex relationships within the data after the sequential features are extracted. The ReLU (Rectified Linear Unit) activation function is commonly used in hidden layers due to its simplicity and effectiveness in mitigating the vanishing gradient problem (Nair and Hinton, 2010). Finally, a SoftMax output layer with a size equal to the number of possible element profiles predicts the probability distribution for each profile type, enabling the model to classify each element effectively. The Adam optimizer with gradient clipping is used to stabilize training, particularly when dealing with the long sequences and high parameter count characteristic of RNNs, ensuring convergence and minimizing the risk of exploding gradients. The Adam optimizer is a popular choice due to its adaptive learning rate and efficient

handling of sparse gradients. It combines the advantages of two other extensions of stochastic gradient descent, namely AdaGrad and RMSProp (Kingma and Ba, 2017).

With the specific model architecture set-up, a fitting loss function needs to be chosen which will be used to monitor the quality of predictions during training. Within this model type, an adapted loss function is necessary to encapsulate the loss over the different classes. Ankalaki (2022) proposed the use of the Categorical Cross-Entropy (CCE) loss function which is based on the Softmax function, which is present in the output layer of the model architecture. The Softmax and CCE functions are given below.

$$\text{Softmax} : f(X_i) = \frac{e^{X_i}}{\sum_{i=1}^{n} e^{X_c}} \tag{6.1}$$

$$\text{CCE} : -\sum_{i=1}^{n} y_i * \log(f(X_i)) \tag{6.2}$$

X_i : Predictions
X_c : Groundtruth
n : Classes

The training setup for the model utilizes a carefully tuned configuration to optimize performance and prevent over-fitting. The model is trained for up to 150 epochs with a batch size of 16, selected based on the dataset size to maintain efficient memory usage and stable gradient updates. To train the model, batch sizes and epochs need to be set. Batch size determines the number of samples that will be propagated through the network at once. Smaller batch sizes can lead to more stable updates, while larger batch sizes can speed up training. Epochs are the amount of time the model is run, but keep in mind that the model can over-fit on noise present in the data when it is run for too long. A learning rate reduction callback (ReduceLROnPlateau, Keras (2024)) monitors the validation loss, dynamically adjusting the learning rate by a factor of 0.5 if improvements stagnate for 10 consecutive epochs, with a minimum learning rate threshold of 1e-6 to ensure the model can continue refining its predictions while preventing excessively small updates. This approach optimizes convergence by gradually reducing learning rates during plateaus, which promotes higher accuracy and better generalization across both training and validation data. To choose when to stop the training, early stopping can be used to halt training when the validation loss starts increasing again, ensuring that the model does not over-fit on the training data (Prechelt, 2012).

The last step is to evaluate the model. This is already partly done during the monitoring of over-fitting, as the validation dataset is used there to determine the hyper-parameters. The final model is then evaluated on the test set to get an unbiased estimate of its performance, as it shows the model's performance on unseen data.

6.3. Output

The output of the model consists of predicted cross-section profiles for each structural element. The model outputs a list of probability distributions across possible cross-sections for each element, allowing for the selection of the profile with the highest predicted probability. This probabilistic approach enables flexibility in cases where multiple profiles show similar suitability. After obtaining the model predictions, the profiles were checked for compliance with Eurocode standards to ensure structural integrity. The model can be validated by running predictions for the previously unseen test dataset. When plotting the predicted output against the actual profiles, a clear diagonal line should be present, indicating that the predictions coincide with the true labels. The accuracy of the model can be quantified by

calculating the performance metrics Accuracy, Recall, Precision and the F1-score, as discussed in chapter 4. Since this model is used to indicate the applicability of reclaimed steel, there is no bias towards false negatives or false positives. This means that the F1-score, which shows a balance between the two, is the most favourable measure to indicate the effectiveness of the model.
This process is repeated for the four different design choice combinations as mentioned before: IPEm, HEAm, IPEb and HEAb. This resulted in four separately trained models to compare the design choices for each application, resulting in an early insight into the effects these design decisions have on the total embodied carbon of the structure.

6.4. Summary

This chapter detailed the development and implementation of a machine learning model aimed at predicting optimal cross-sections for structural elements in data centre design, aiming to streamline the incorporation of reclaimed steel in a secondary model. Leveraging a carefully structured RNN architecture, the model effectively processes variable input lengths and multiple element types, providing predictions for each structural component based on unique grid configurations, element types, and lengths.

The methodology involved data preparation, including normalization, one-hot encoding and padding to handle varying sequence lengths. Key design choices in the model architecture (such as bidirectional LSTM layers, dropout regularization, and adaptive learning rate adjustments) helped the model capture sequential dependencies and generalize well across diverse data configurations. The performances of the models are further discussed in chapter 8. A probabilistic output approach allowed for flexible cross-section selection while ensuring compliance with structural standards.

This machine learning model not only streamlines cross-section selection but also integrates with the broader design process, creating the groundwork for future integration of the secondary optimization step within the model.

7

Optimization Model

The following chapter discusses the fifth research sub-question:

5. *What are the key optimization variables for incorporating reclaimed steel into a design?*

Figure 7.1 shows the approach to answer this question, it contains a shorter version of the workflow given in Figure 5.1 as it excludes the steps for data gathering and training of the ML model. The cross-section prediction can be done directly through the pre-trained RNN model to quickly set up an initial design of a data centre. The next step is to match the components to a given database of reclaimed steel, after which the LCA calculations can be performed to indicate the effects of reclaimed steel integration. Due to the swift nature of the generation, an optimization algorithm can be used to generate the design configuration which contains the most reclaimed steel. While still ensuring a valid design that adheres to the design standards of data centres. This optimization algorithm is executed entirely within Grasshopper.

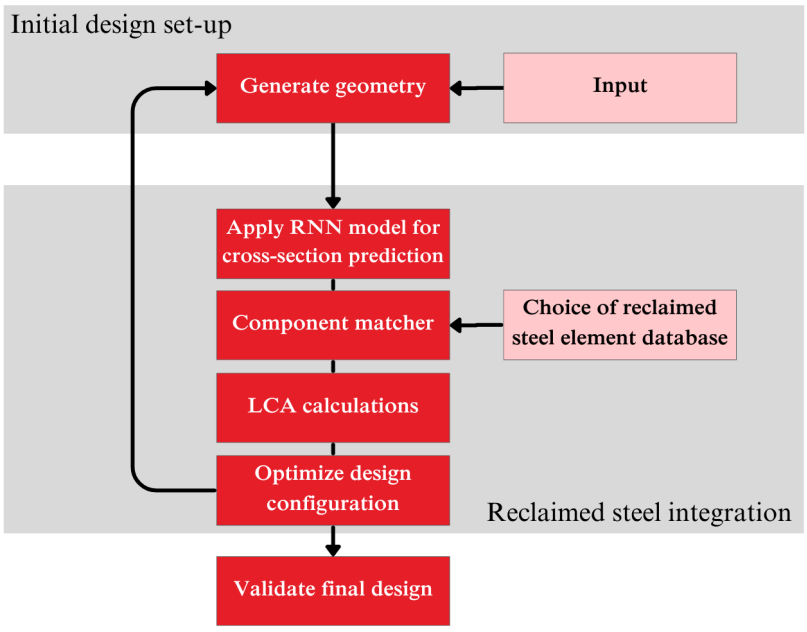


Figure 7.1: Schematic of the optimization model setup in Grasshopper and the relation to the machine learning RNN model as mentioned in chapter 6.

Grasshopper is a visual programming language and environment that runs with the Rhinoceros 3D computer-aided design (CAD) application and is widely used for parametric design. It allows designers

to create complex forms and structures through a graphical interface without writing traditional code. Next to that, it is particularly powerful for optimization tasks, as it can handle multiple variables and constraints to find the best possible solution as stated in chapter 3.3. In this chapter, the optimization model aimed to design an optimal data centre developed within this research is explained. The model incorporates reclaimed steel to minimize the embodied carbon of the structure, using background information as explained in part one of this research.

7.1. Input for RNN Model

As discussed in chapter 5.2, in this research there are four options for design choices: HEA profiles on all elements with a stability system of moment fixed connections or hinged connections with a braced frame structure. The other design option is IPE beams and HEB columns, again with the same options for stability systems. These inputs are defined in the Grasshopper model with the use of value lists, as can be seen in Figure 7.2. The second input of the model is the floorplan size, how these configurations lead to the final floor plan design has been described in chapter 5.1. In the figure, the gene pools are shown which are used to generate the grid-spaces based on the location of the columns in the floorplan. As mentioned in chapter 3.1, columns can only be located at the location of a data rack, which is how the grid spaces are defined. The total number of rows, the number of racks per row and the lists of integers indicating the column locations are entered into a Python component. In this component are the data racks visualized and the actual floorplan size and grid spaces calculated. The code can be found in Appendix E.

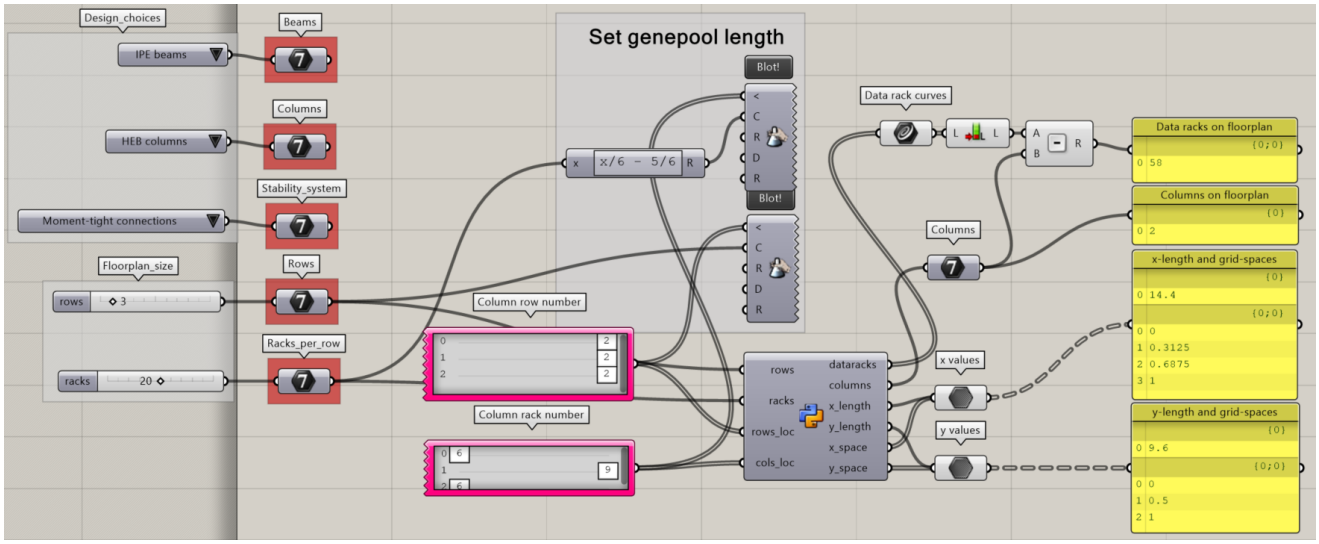


Figure 7.2: Overview of input options for the Grasshopper model, consisting of three design choices and 2 options for floorplan sizes. The size of the gene pools, which determine the grid spaces, is influenced by the floorplan size.

When the design choices are set and the distances between the columns are determined, they can be transformed into a grid, as discussed earlier in chapter 5.1. In this research, the floor height and number of floors are not considered variable inputs as these values are determined by the building type. After all the elements are created by elevating line elements to their destined locations, the data can be converted to match the RNN input. This is done by determining the vector of the beams, giving them either an x or a y direction handle, and normalizing the IDs to be between zero and one. These values can then act as the input of one of the 4 generated RNN models, depending on the initial design choices, to predict the cross-section per element. The code from the Python components in the Grasshopper script can be found in appendix E.

7.2. Component Matcher

After the List of elements present in the design is created, a comparison can be made to the list of readily available reclaimed steel elements from chapter 2.4. To make this comparison, a Grasshopper component developed by Lut et al. (2022) can be used. The component, called MagPie follows the workflow as shown in Figure 7.3, where it matches the design elements to a stock of reclaimed elements based on a perfect match in cross-section and length, or closest fit. The latter would result in either an over-dimensioned cross-section or the need to cut off the excess length.

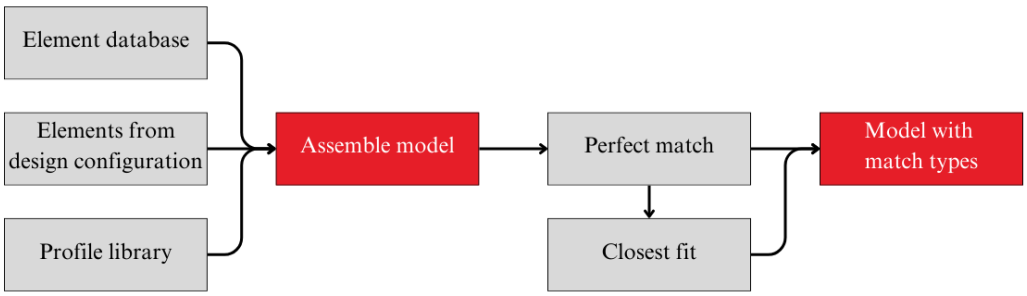


Figure 7.3: Workflow of the component matcher (Lut et al., 2022)

After the component matcher, all elements are divided into three groups: The new steel needed for the design, the reused steel and the added cut-off waste which is created when indirect reuse is applied. All three options come with their own LCA calculations as per chapter 2, resulting in a total embodied carbon score for the design.

There are two types of indirect reuse possible, either over-dimensioning of the beam or an element length which is too long and needs to be cut off for the design. The type of indirect reuse that is applied depends on the cut-off waste length and the section size. The component compares the added volume from both options and chooses the least amount. Table 7.1 shows a calculation done for different profiles and the same two indirect reuse options applied; choosing one profile higher or choosing an element with one meter of cut-off length. In the case of IPE100, the second option creates less added volume. Whereas with IPE500, the first option generates significantly less volume.

Table 7.1: Comparison of the two different options for indirect reuse for two different section profiles applied in the same length.

	Volume [mm^3]	Volume increase
IPE100, L=4	4.13e6	-
1 Profile higher (IPE120)	5.28e6	28%
1 Meter cut-off length	5.26e6	25%
IPE500, L=4	46.2e6	-
1 Profile higher (IPE550)	53.8e6	16%
1 Meter cut-off length	57.8e6	25%

7.3. LCA Calculations

To evaluate the environmental impact of the design, a Life Cycle Assessment (LCA) was conducted comparing the embodied carbon for a new structure versus a structure incorporating reused steel elements. The Grasshopper model shown in Figure 7.4 was used to calculate the total embodied carbon for two design scenarios: One containing only new steel and one design utilizing reclaimed steel. The model produced several key outputs, including the total embodied carbon for the new structure [$kgCO_2e$], the structure with reused elements [$kgCO_2e$], and the percentage reduction in embodied carbon due to reuse [%].

Table 7.2: Embodied carbon of four different types of applications of steel. When indirect reuse is applied, the weight of the used part is multiplied by 0.3423, while the weight of the cut-off part is multiplied by 0.129. Those values combined serve as the total embodied carbon for indirect reuse.

Steel application type	Equivalent embodied carbon per kilogram [$kgCO_2e/kg$]
New steel	1.0494
Direct reuse of steel	0.3294
Indirect reuse of steel	0.3423
Indirect reuse - cut-off waste	0.129

The model also calculated additional metrics such as the carbon score, representing the embodied carbon per square meter of the structure [$kgCO_2e/m^2$], and the embodied carbon per data rack present in the design [$kgCO_2e$/rack]. These metrics allow for a complete understanding of the environmental savings achieved through reuse. The LCA calculations are based on specific carbon factors for different types of steel usage, as outlined in chapter 2. These values, as shown in Table 7.2, are incorporated into the Grasshopper model to calculate the overall carbon impact of both the new and reused steel structures.

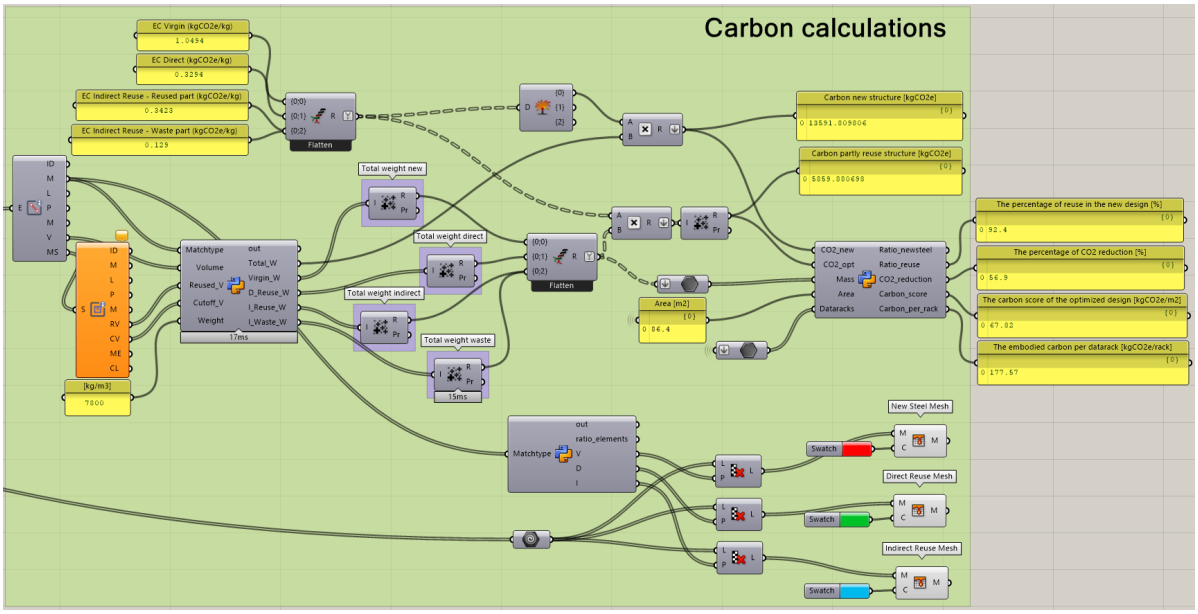


Figure 7.4: Overview of the LCA calculation in Grasshopper based on the weight of material per application type. The contents of the Python components shown can be found in appendix E.

7.4. Optimizer

The final step in the Grasshopper model workflow focuses on optimizing the structural configuration to achieve minimal embodied carbon. This phase relies on the Opossum optimization plug-in, which offers a powerful combination of machine learning-based RBFOpt and evolutionary CMA-ES algorithms, making it well-suited for tackling complex and multi-variable structural design tasks, as discussed in chapter 3.3. Through iterative adjustments of the grid spaces, the optimizer narrows down the floor plan configuration that maximizes the application of reclaimed steel.

The optimization model evaluates several critical metrics:

- Total Embodied Carbon: The primary target for minimization, as it reflects the environmental impact of the material choices made within the structure. This includes both direct and indirect reuse of reclaimed steel, providing a comprehensive carbon calculation for each design iteration. Related to this metric is the carbon score calculated as well, which indicates the embodied carbon per square meter of the building and it is a generic term for embodied carbon comparisons. (Hrivnak, 2023)
- Percentage of Total Reuse: This metric indicates the proportion of reclaimed steel incorporated into the design. High reuse rates indicate an efficient allocation of existing resources, reducing the need for new steel and its associated carbon footprint.
- Percentage of CO_2 reduction: Next to the total embodied carbon of the design incorporating reuse, the embodied carbon of a new design is calculated. This difference shows the impact reuse can have on the embodied carbon of a design.
- Embodied carbon per data rack: This metric is set up to also incorporate a design requirement, as data centres often require the least amount of columns in the floorplan and a design which houses the most data racks possible, as discussed in chapter 3.

These metrics combined determine the score for each configuration, with carbon per data rack acting as a single, unified metric that balances environmental impact with design practicality. By optimizing for this carbon-per-rack score, the model inherently considers both structural integrity and carbon efficiency, guiding the design towards sustainable, viable configurations.

Using an iterative process, the Opossum optimizer modifies the grid spaces for which the embodied carbon per data rack is recalculated each time. This adaptive feedback loop allows the optimizer to adjust its approach as it evaluates configurations, ultimately selecting the grid spaces that yield the lowest carbon impact while meeting functional requirements. The RBFOpt was chosen for this research as it is a single objective optimization, the optimization will terminate if the variable has not been improved in the last 40 iterations. The result is an optimized structural configuration that maximizes reclaimed steel usage, reduces embodied carbon, and balances the practical needs of data centre layout and structural integrity. This iterative optimization approach provides crucial insights, offering designers a clear path to more sustainable building practices.

7.5. Summary

In this chapter, we have detailed the Grasshopper optimization model developed to integrate reclaimed steel in data centre structures. The model combines parametric inputs, RNN cross-section predictions, and LCA calculations with an advanced optimization framework, centred around reducing embodied carbon through the reuse of reclaimed steel. Each component of the model plays a distinct role, contributing to an integrated solution that balances structural, environmental, and economic considerations. The optimization phase, powered by the Opossum optimizer, is the core of this model, transforming raw input configurations into valid design solutions. By iteratively testing grid spaces and component choices, the optimizer identifies a configuration that meets functional requirements while achieving the lowest possible carbon footprint. The metric 'carbon per data rack' provides a single objective for optimization, merging structural integrity with environmental goals. This process supports data-driven design, enabling engineers to make informed decisions that prioritize sustainability alongside performance.

Overall, this model demonstrates a replicable, adaptable approach to sustainable structural design, showcasing how reclaimed materials can be incorporated to achieve reductions in embodied carbon. The resulting designs of this workflow are presented and validated in chapter 8. The developed method establishes a foundation for future work, potentially applying similar optimization processes to other building types or expanding the dataset of reclaimed elements to maximize reuse.

III

Research Outcome

8

Results

This chapter addresses the results from the workflow as described in chapter 5, 6 and 7, examining the automatic integration of reclaimed steel into data centre design to answer the main research question.

"How can the incorporation of reclaimed steel elements be made more accessible within the design process of datacentres when using Machine Learning applications?"

The chapter is structured to discuss the results of the RNN models as a cross-section prediction and the applications of the optimization model, which aims to maximize the reuse of steel elements in load-bearing structures. The research's core objective was to develop predictive models to facilitate efficient cross-section selection from available reclaimed steel stock. By training distinct Recurrent Neural Network (RNN) models on four design configurations, the models were enabled to predict appropriate cross-sections based on parameters such as element length, location, and type. These cross-section predictions then feed into a separate optimization model developed in Grasshopper. The chapter concludes with a validation of the final model by using Oasys GSA.

8.1. RNN Model

The RNN models as explained in chapter 6 were set up as a proof of concept to incorporate machine learning into the design process. This exploration is linked to sub-question 4 within this research:

4. *How can machine learning techniques be applied to incorporate reclaimed steel within the design process effectively?*

With the development of these models, the first step into the incorporation of reclaimed steel with machine learning has been made, as the RNN is set up to predict the cross-sections of the design. To facilitate the selection of optimal steel cross-sections, four separate RNN models were trained to predict the profiles for different design choice combinations. The four combinations are visualised and explained in Figure 5.3 and abbreviated to IPEm, IPEb, HEAm and HEAb throughout this chapter. Each model's performance was evaluated based on training and validation loss, illustrated in Figure 8.1 and Table 8.1. These are then further analysed by running the prediction on the unseen test data and evaluating the confusion matrices.

Figure 8.1 illustrates the training and validation loss curves, with all models demonstrating effective convergence as the loss values decrease over time (the epochs). This indicates that the models were able to generalize well from the training data, maintaining a stable performance on unseen validation data. The IPEm and IPEb models showed steady convergence after approximately 100 epochs, after which they were stopped early to prevent overfitting on the training data.

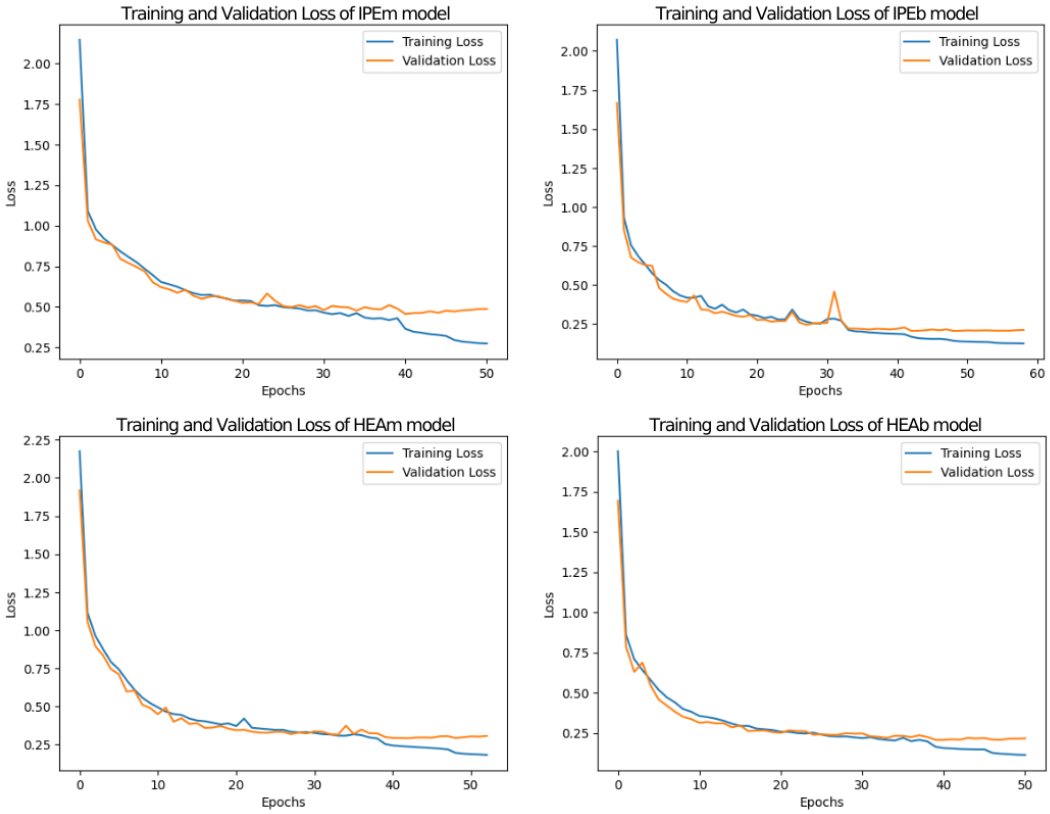


Figure 8.1: Result of the cross-entropy loss function over time (epochs) for the training and validation dataset. This is shown for each of the 4 models.

Whereas Figure 8.1 shows the training over time, Table 8.1 shows the last recorded residual of the loss function before the validation loss starts to increase from overfitting on the training data as explained in chapter 4.1.2. The table shows that IPEm results in the highest residual loss-function value, indicating that this model will have the lowest accuracy. The IPEb model on the other hand approaches the lowest residual loss values and training continued for the highest amount of epochs, indicating more useful interconnections between the data could be found. These extremes are an interesting result when looking at the percentage of non-valid elements as discussed in chapter 5.2. Where the IPEb and IPEm configurations consist of 10 to 100 times more N.A values than the HEA data. This in combination with the model accuracy could indicate that the IPEb configuration created a better understanding of the N.A values whereas the IPEm did not contain enough data to learn that pattern. The HEA configurations contain so few N.A values due to the high beam dimensions that are possible (as IPE goes to 600mm in height and HEA up to 1000mm), creating a stable accuracy even when the model does not predict the existence of N.A values.

Table 8.1: Residuals of the loss function for training and validation data together with their respective accuracy. This is explained in chapter 4.1.2

	Residual training loss	Total training accuracy	Residual validation loss	Total validation accuracy
IPEm	0.3095	0.8846	0.4562	0.8264
IPEb	0.1164	0.9641	0.2056	0.9343
HEAm	0.1924	0.9319	0.2921	0.8862
HEAb	0.1266	0.9580	0.2074	0.9297

To further analyse and validate the RNN models, predictions were generated for the previously unseen test data, which was determined to be 5% of the initial dataset as discussed in chapter 6. Since the model contains a multi-class classification problem, only calculating the accuracy might not be enough as imbalances across labels are not shown, as previously discussed in chapter 4. Following this, the models were evaluated using both the accuracy and F1-score, which can be seen in Table 8.2. Across all datasets, the models achieved high performance, with F1-scores ranging from 0.8202 (IPEm) to 0.9350 (IPEb). The highest performance was observed for datasets IPEb and HEAb, where F1-scores were 0.9350 and 0.9335, respectively, indicating the model's strong ability to generalize. Even for the lower-performing dataset (IPEm), the model achieved an F1-score of 0.8202, demonstrating consistent effectiveness across varied conditions.

These clear differences in accuracy for the two stability systems can be explained by looking at the force distribution between the two. In the case of a braced structure, the elements are connected with hinges, which entails that no moments are transferred between the two. This results in the fact that each element is only subjected to the moments from forces directly on the element itself, together with shear and normal forces transferred through the system. Creating a straightforward calculation. However, for moment-fixed connections, the elements can be seen as continuous. Creating an interplay between the elements throughout the system as the heavy loads are transferred via moments. Since this is a more complex calculation, the hidden interconnections between the data are more difficult to pick up for the RNN model. This could be prevented by deepening only the RNN models trained for moment-fixed connections by adding extra input features or LSTM layers in the model.

Table 8.2: Accuracy and weighted F1-Score of the four RNN models when applied to the unseen test dataset. The score is weighted based on the number of instances each label is present in the output data.

Model	Accuracy	Weighted F1-Score
IPEm	0.8215	0.8202
IPEb	0.9349	0.9350
HEAm	0.8825	0.8822
HEAb	0.9337	0.9335

To visualise the label-specific behaviour of the models, a confusion matrix can be set up, indicating the accuracy per label. The confusion matrix is often shown as a heatmap where the predicted value resulting from the trained model is plotted against the actual value. This can only be applied for supervised learning as the result needs to be known, which is the case for the remaining test dataset. A diagonal line in the figure indicates a correct prediction, as it aligns with the actual value. Figure 8.3 and 8.2 both show the clear diagonal line from the elements, with some over or under-estimation of the profiles of only 1 profile size. Note that the scale of the colour map is logarithmic, indicating that most predictions are in fact on the diagonal line.

Figure 8.2 shows the Predictions versus the true labels for the IPE configurations. The figure shows a light-grey line that separates the IPE beams from the HEB columns, as that is how this design configuration is set up in chapter 5. These separation lines show that there are no predictions that have a false profile type (IPE versus HEB). It does however result in a lower number of elements per label, which could explain the higher level of confusion shown for high HEB profiles. Figure 8.3 on the other hand shows a clear diagonal line throughout as in this design configuration, one profile type is present.

From Figure 8.2, it can be concluded that the lower accuracy rate from the moment-fixed stability system models is caused by less accurate predictions of the column profiles, as the highest level of confusion is located at the HEB profiles. This further validates the conclusions stated based on the test accuracies of

Table 8.2 that moment-fixed stability systems could benefit from additional design features as input. As mentioned in chapter 5.2, the current models operate with three input features: Element ID, element type and element length. Together they indicate the location of the element within the model. An additional design feature that could improve the moment-fixed predictions is the number of beams connected to the columns, as two beams indicate a corner column, three indicate an edge column and four a middle column. This affects the total direction of the transferred moment onto the column creating highly different values for the column's buckling resistance, following from the calculations shown in Appendix A.

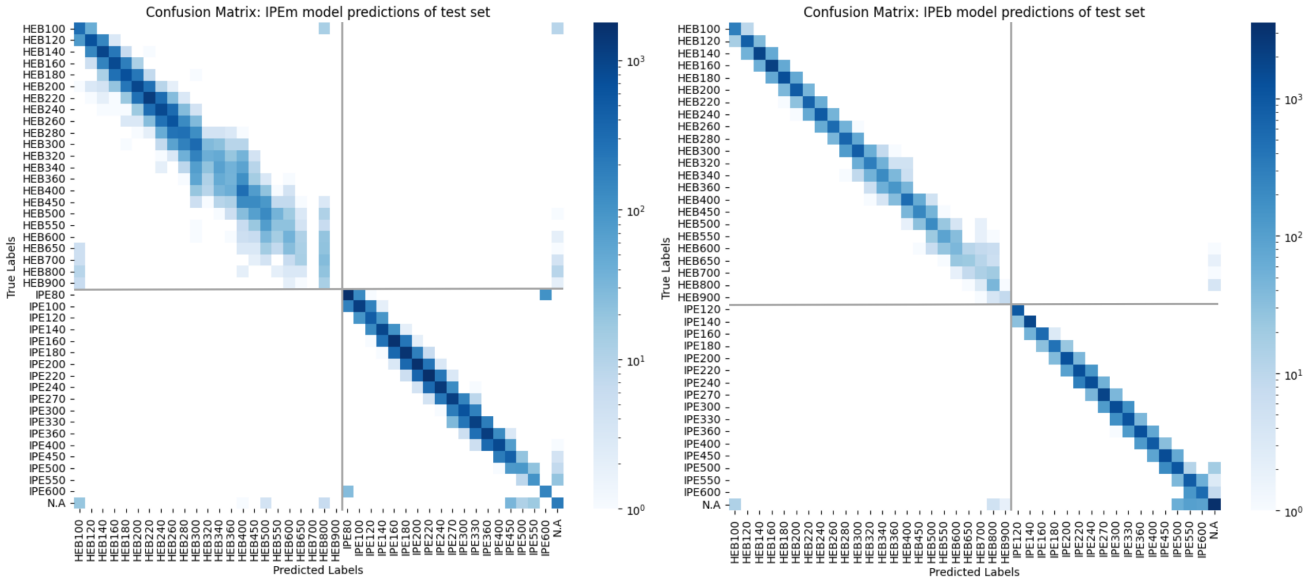


Figure 8.2: Confusion matrix of the predicted values versus the true values for the unseen test dataset for the IPEm and IPEb model. The grey lines indicate the separation between the IPE profiles chosen for the beams and the HEB profiles for the columns. Note that the gradient scale is set to logarithmic to show the spread of predictions more clearly.

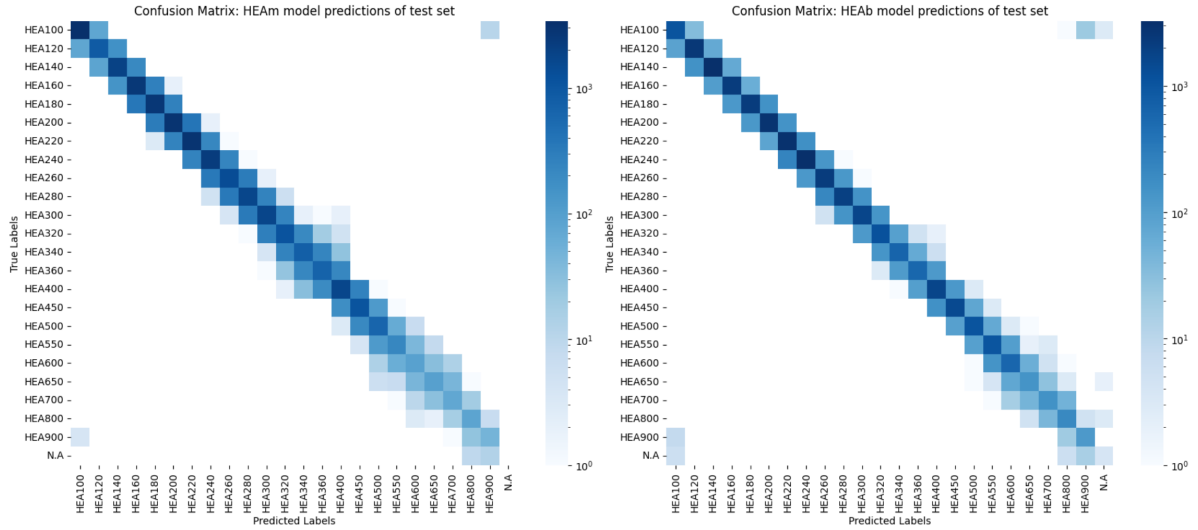


Figure 8.3: Confusion matrix of the predicted values versus the true values for the unseen test dataset for the HEAm and HEAb model. Note that the gradient scale is set to logarithmic to show the spread of predictions more clearly.

8.2. Grasshopper Model

The following sub-chapter shows the results linking to the last sub-question:

5. *What are the key optimization variables for incorporating reclaimed steel into a design?*

After training and validating the prediction models, they can be loaded into the Grasshopper optimization model to start the second part of the automated design process. To show the effects of the cross-section selection process and generate a comprehensive overview, a single design configuration was selected. The configuration creates interesting results when it makes predictions for unseen data and visualises the effect of the model on a relatively large design. To adhere to these aspects, a floorplan of 5 rows with 20 racks per row is chosen. This configuration was not present in the RNN training data from Table 5.1 and results in a large floorplan equal to a size of 14.4 by 14.4 meters.

The figures below illustrate the outcomes of the optimization model in Grasshopper, designed to integrate reclaimed steel into a load-bearing structure with minimal embodied carbon. For each of the three datasets of reclaimed steel elements from chapter 5.3, four design configurations are presented, reflecting variations in structural choices: IPE/HEB beams and moment-fixed (IPEm) vs. braced (IPEb) systems, as well as HEA beams with moment-fixed (HEAm) and braced (HEAb) systems. The colour coding highlights the source of materials, where red indicates newly sourced steel, green represents directly reused elements, and blue denotes indirect reuse. Key optimization variables across these configurations include grid spaces, the total embodied carbon, and the proportion of reused materials in the accompanying tables. These LCA calculation results are based on the calculations explained in chapter 7.3.

First, the Open-Source dataset from Figure 5.5 was applied to the model, creating a real-life example of reclaimed steel integration into the design. Figure 8.4 shows the optimized structures and their reclaimed steel applications, there it can be seen that direct reuse (green elements) is not as often applied. It also shows that reuse is mostly applied in the roof structure or columns, which stems from the fact that the Open-Source dataset consists of relatively low section profiles whereas the high floor loads in the data centre require more resistance. From Table 8.3 it can be seen that for grid spaces in y-direction, relatively small spans are applied as the grid is divided into 4 or even 5 sections. This indicates that the added steel and decrease in possible data racks in the floorplan do not outweigh the CO_2 reduction possible when the spans are smaller.

Next, the model was run based on the Exiting Designs dataset as shown in Figure 5.6. This application is based on a hypothetical situation where a decommissioned structure is disassembled for reuse. This application is a common practice due to the lowered transportation cost and a better indication of the previously applied load when the elements are from one structure, as discussed in chapter 2. Figure 8.5 shows the optimized structure for the same four design choice combinations as discussed for the Open Source dataset. It can be seen that the Existing Design dataset consists mostly of IPE beams, as these configurations incorporate the most reused elements. For the HEA configurations, only the elements under lower loading conditions like the roof structure incorporate some reuse, since the few HEA elements have a relatively small cross-section.

Lastly, the optimization model was applied based on the generated dataset from Figure 5.7 based on Eurocode calculations (Appendix B). This dataset is the largest and has the most even spread across cross-sections, which can be seen from the high percentage of reuse from Table 8.5. The application of this dataset shows the generalized availability of steel elements when the basic design codes are applied. Figure 8.6 shows almost no red elements, meaning that the optimal design coincides with the generated dataset relatively well.

Table 8.3: Results from the Grasshopper optimizer when applied with the Open-Source dataset. The configuration ID indicates the design choices, whether the design consists of IPE beams and HEB columns or all HEA, and the applied stability system, either moment-fixed connections or bracings.

Open-Source dataset				
Configuration ID	IPEm	HEAm	IPEb	HEAb
Grid - x	[6, 9]	[6, 9]	[8, 7]	[7, 7]
Grid - y	[1, 2, 2]	[1, 2, 2]	[1, 1, 2, 1]	[1, 2, 2]
Total carbon - original [kgCO_2e]	17169.3	25060.4	29631.7	34399.3
Total carbon - improved [kgCO_2e]	12224.9	19565	22968.7	28817.3
Total reuse [%]	62.9	48.2	56.3	36.8
CO_2 reduction [%]	28.8	21.9	22.5	16.2
Carbon Score [kgCO_2e/m^2]	58.96	94.35	110.77	138.97
Carbon per data rack [kgCO_2e/rack]	130.05	208.14	249.66	306.57

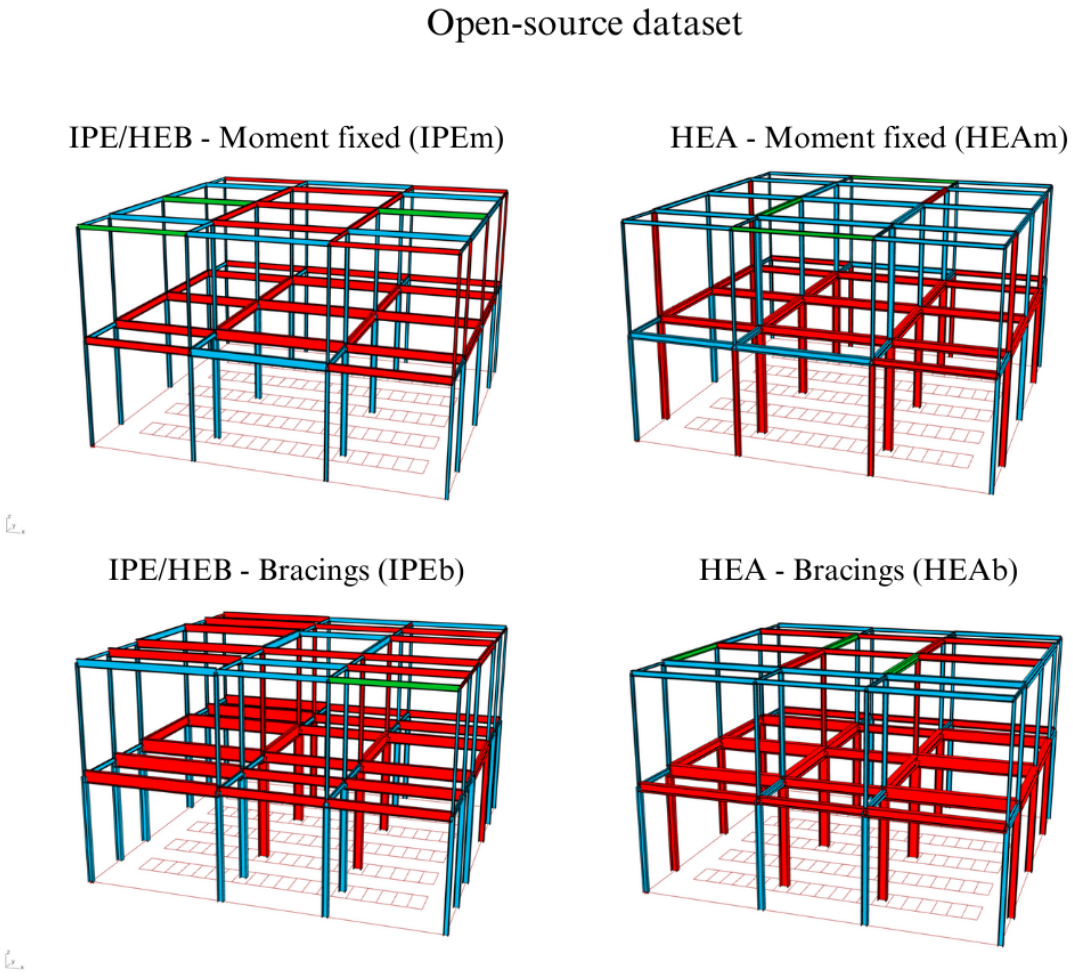


Figure 8.4: Optimized design considering the four different design choice combinations as discussed in chapter 5 when the Open-Source database from Figure 5.5 is applied. The red colour indicates the new steel, green is direct reuse and blue is indirect reuse. The structure is optimized to generate the least amount of embodied carbon per data rack.

Table 8.4: Results from the Grasshopper optimizer when applied with the Existing Designs dataset. The configuration ID indicates the design choices, whether the design consists of IPE beams and HEB columns or all HEA, and the applied stability system, either moment-fixed connections or bracings.

Existing Designs dataset				
Configuration ID	IPEm	HEAm	IPEb	HEAb
Grid - x	[6, 9]	[6, 9]	[8, 7]	[7, 8]
Grid - y	[2, 2]	[1, 2, 2]	[2, 2]	[2, 2]
Total carbon - original [$kgCO_2e$]	19137.9	25060.4	29242.8	33772.8
Total carbon - improved [$kgCO_2e$]	9249.2	24373.2	18885.9	33133.1
Total reuse [%]	97.4	10.7	77.8	7
CO_2 reduction [%]	51.7	2.7	35.4	1.9
Carbon Score [$kgCO_2e/m^2$]	44.6	117.54	91.08	159.79
Carbon per data rack [$kgCO_2e$/rack]	96.35	259.29	200.91	345.14

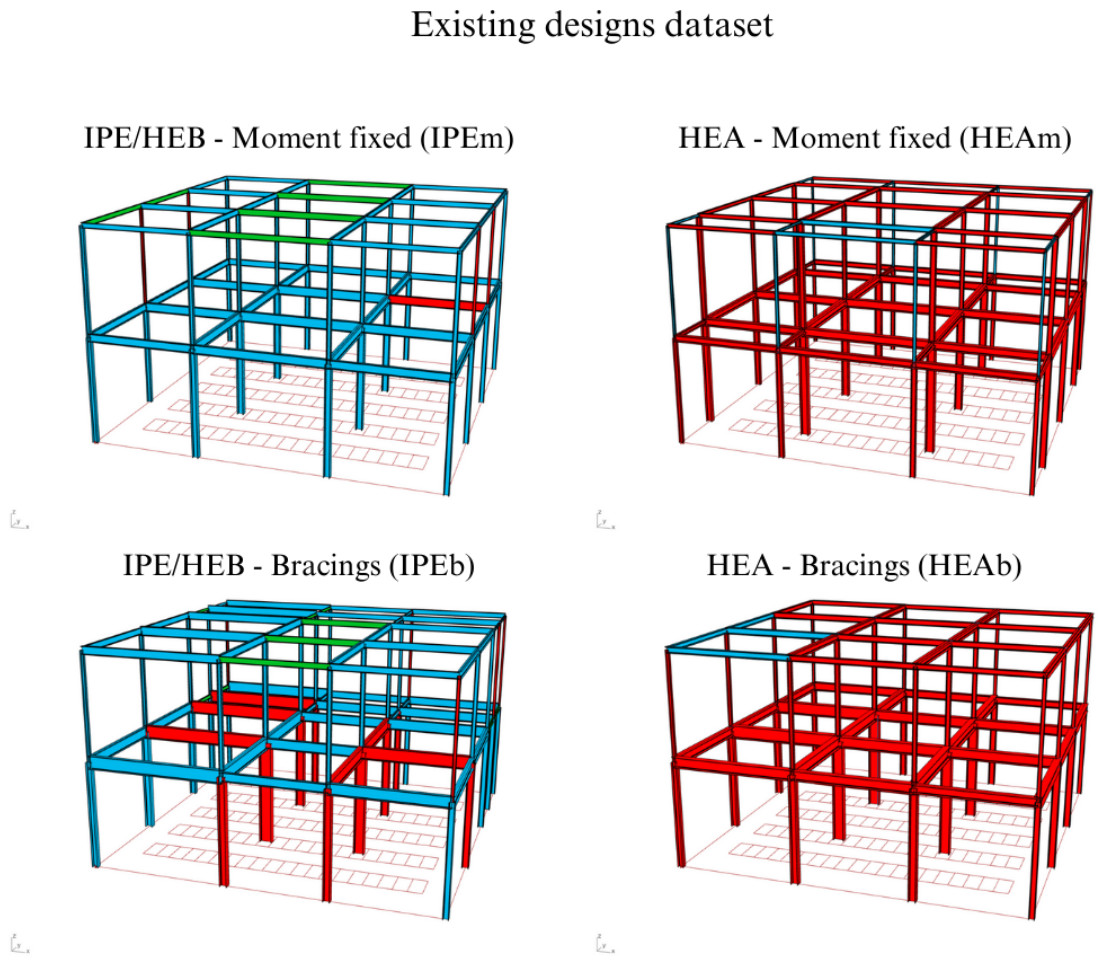


Figure 8.5: Optimized design considering the four different design choice combinations as discussed in chapter 5 when the Existing-Designs database from Figure 5.6 is applied. The red colour indicates the new steel, green is direct reuse and blue is indirect reuse. The structure is optimized to generate the least amount of embodied carbon per data rack.

Table 8.5: Results from the Grasshopper optimizer when applied with the generated dataset. The configuration ID indicates the design choices, whether the design consists of IPE beams and HEB columns or all HEA, and the applied stability system, either moment-fixed connections or bracings.

Generated dataset				
Configuration ID	IPEm	HEAm	IPEb	HEAb
Grid - x	[7, 9]	[7, 9]	[8, 8]	[7, 8]
Grid - y	[2, 2]	[2, 2]	[2, 2]	[2, 2]
Total carbon - original [kgCO_2e]	19481.6	24947.8	32276.3	33772.8
Total carbon - improved [kgCO_2e]	7756.8	9358.1	13715	13389.2
Total reuse [%]	98.4	96.8	96.6	98.4
CO_2 reduction [%]	60.2	62.5	57.5	60.4
Carbon Score [kgCO_2e/m^2]	37.41	45.13	66.14	64.57
Carbon per data rack [kgCO_2e/rack]	80.8	97.48	142.86	139.47

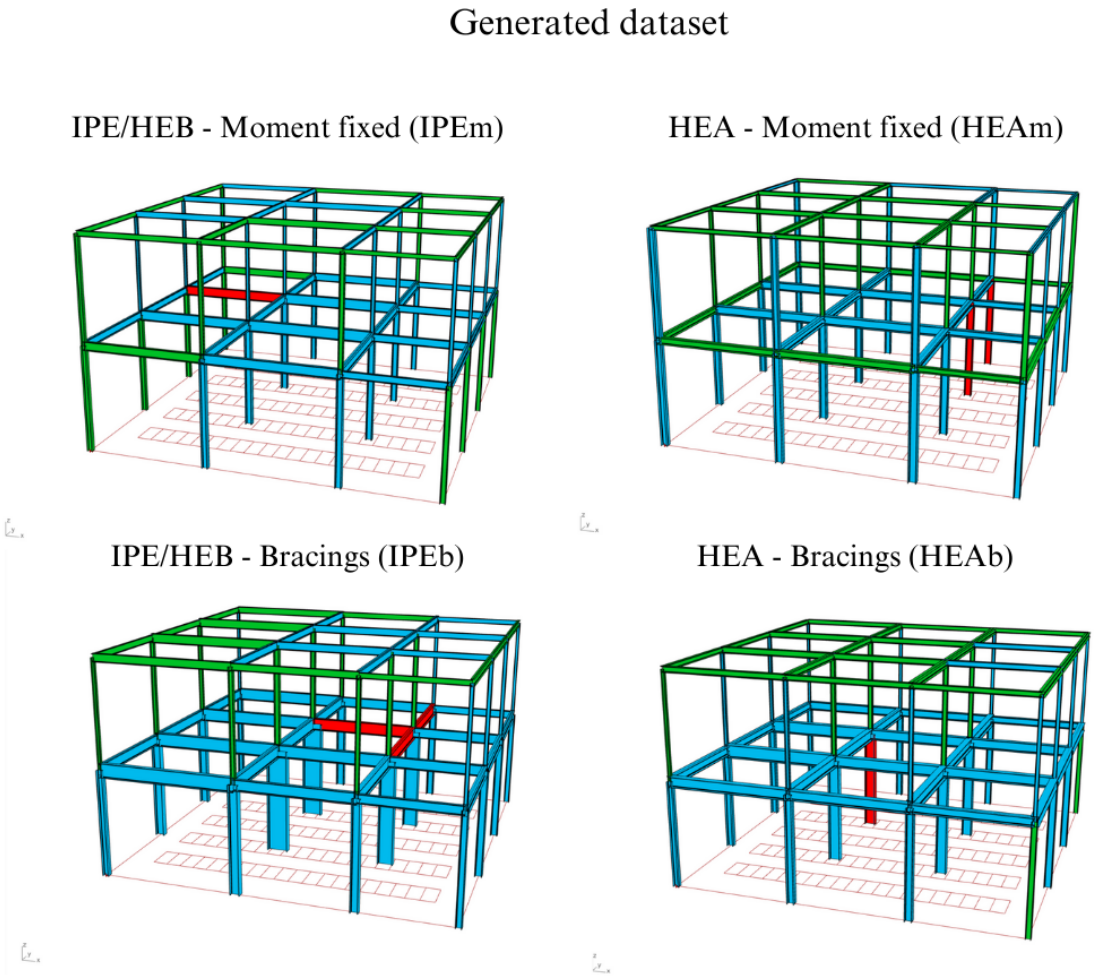


Figure 8.6: Optimized design considering the four different design choice combinations as discussed in chapter 5, when the Generated database from Figure 5.7 is applied. The red colour indicates the new steel, green is direct reuse and blue is indirect reuse. The structure is optimized to generate the least amount of embodied carbon per data rack.

8.3. Model validation

The last step of the workflow presented at the beginning of Chapter 5 is the validation of the final optimized structure. This step uses the FEM software Oasys GSA to calculate force distributions, deflections and overall steel utilization, ensuring the structural integrity and efficiency of the design. For this validation, the optimized design based on the generated database is applied to both the IPEm and IPEb configurations. These configurations are selected for specific reasons: the IPEm configuration provides a test case for the less accurate predictions of the RNN, while the IPEb configuration allows a comparison of the force distribution between the two stability systems considered.

The GSA model was set up by importing the optimized model into the software. The model was optimized to include the most reclaimed steel from the Generated dataset and it includes the predicted profiles from the trained RNN model. To resemble the design from the Grasshopper optimization, the same load combinations are applied as stated in Appendix A. The design was subjected to a static analysis that calculates the load distribution with the use of a finite element analysis. These loads are then tested according to Eurocode 3, which results in the steel utilization per element. The analysis incorporated both linear elastic behaviour and global stability effects, which are particularly relevant given the mix of braced and moment-resisting stability systems in the designs.

The results of the validation as shown in Figure 8.7 and 8.8 highlight the robustness of the optimization process and the reliability of the structural predictions made by the model. The moment distributions highlight key differences between the two stability systems. For the braced structure (IPEm), moments are highest at the midspan of beams, reflecting the bending behaviour expected in simply supported spans. In contrast, the moment-fixed system (IPEb) shows peak moments at the connections, consistent with the rotational restraint provided by the fixed joints. These patterns align with the expected structural behaviour, indicating that the stability system types were accurately generated.

The steel utilization ratios provide further insight into the design's efficiency. All beams exhibit utilization ratios between 0.2 and 0.8, ensuring an acceptable balance between material usage and structural performance. The spread in unity checks stems from a combination of factors:

- Reclaimed Steel Application: Over-dimensioning of elements is permitted when it enables a higher percentage of reclaimed steel to be used. This strategy results in some elements being underutilized (closer to 0.2), as their capacity exceeds the applied loads due to the limited availability of smaller reclaimed profiles.
- Interconnection Effects: The interconnected nature of the structural system causes load redistribution. These variations in load paths lead to varying failure mechanisms which can be amplified in critical elements when even one profile lower is applied. Elements near connections often experience higher forces, leading to utilization ratios closer to 0.8, while elements further from load-critical regions exhibit lower utilization.

The analysis underscores the success of the proposed methodology in combining machine learning predictions with structural optimization and validation tools. The moment diagrams validate that the stability system designs align with expected behaviours for the applied load cases. Moreover, the well-balanced utilization ratios indicate that the design efficiently meets both performance and sustainability objectives. The observed spread in utilization ratios, although seemingly random, reflects the constraints and strategies inherent to the design. Specifically, the integration of reclaimed steel prioritizes environmental impact reduction over strict optimization of individual member utilizations. This trade-off, evident in the results, offers a promising pathway toward reducing embodied carbon through the reuse of steel, combining advanced computational methods with practical engineering requirements.

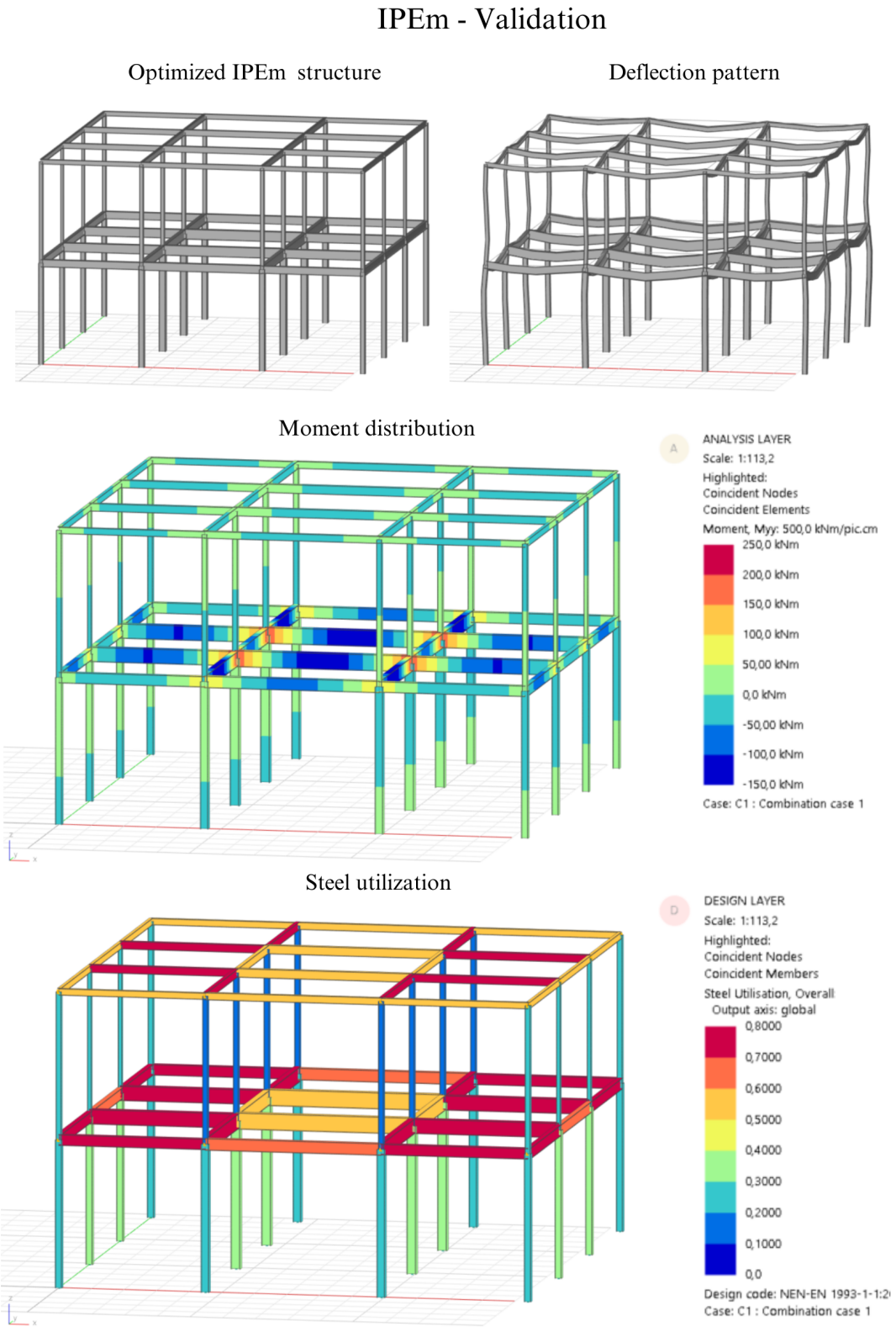


Figure 8.7: IPEm validation by placing the optimized design with 5 rows and 20 racks per row based on the Generated dataset

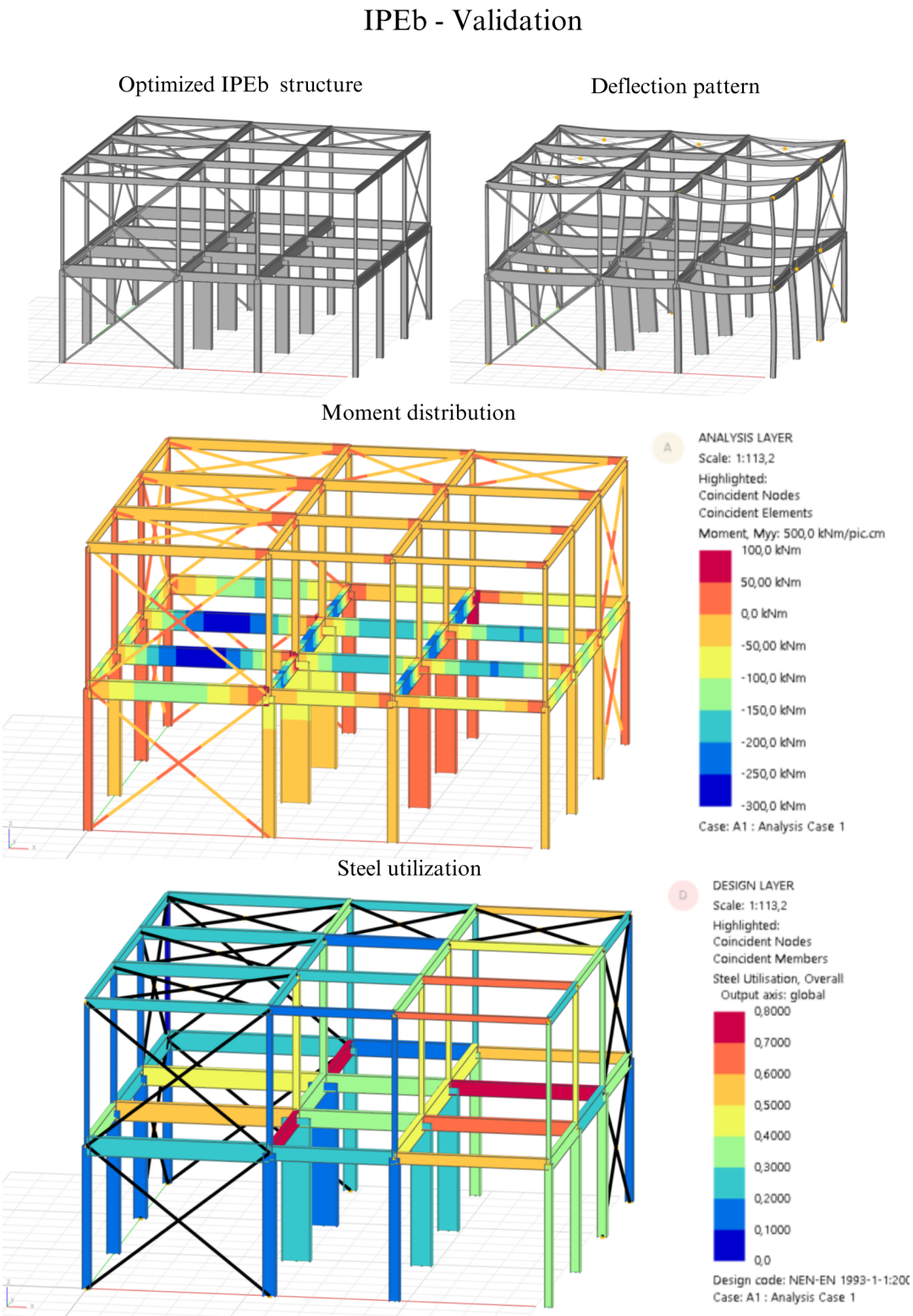


Figure 8.8: IPEb validation by placing the optimized design with 5 rows and 20 racks per row based on the Generated dataset

8.4. Summary

The results chapter concludes by addressing the thesis's main research question through a thorough evaluation of both the RNN models and the Grasshopper optimization model. The chapter demonstrates that RNN models can effectively predict cross-sections from available reclaimed steel stock, which can then inform an optimization model in Grasshopper to maximize steel reuse in a load-bearing structure for data centres. The following key findings are addressed:

1. Performance of RNN Models:
    - The models regarding the braced structure demonstrated the best performance, achieving a weighted F1-score for IPEb of 0.9350 and for HEAb of 0.9335.
    - Lower accuracy was observed for the moment-fixed structures, with scores for HEAm of 0.8822 and the lowest for IPEm of 0.8202.
    - Confusion matrices indicated that the less accurate predictions lie within the column calculations
2. Stability System Insights:
    - Braced systems (IPEb, HEAb) outperformed moment-fixed systems (IPEm, HEAm) due to their more simple force distribution.
    - Enhancing moment-fixed systems' accuracy could involve adding input features, such as the number of connected beams, which influence moment transfer and buckling resistance.
3. Optimized Reclaimed Steel Integration:
    - Designs based on the Generated Data set achieved the highest reuse percentages (>90%) with minimal reliance on new steel and a CO_2 reduction up to 62.5%.
    - The Open-Source dataset showed limited direct reuse, mainly in roof structures and columns. With varying percentages of CO_2 reduction between 16.2 and 28.8%.
    - The Existing Designs dataset demonstrated the highest variation in applied reuse with higher reuse for configurations involving IPE beams and HEB columns (35.4 and 51.7%) in contrast with the HEA configurations (1.9 and 2.7%).
4. Design Validation:
    - Validations showed the correct application of the different stability systems considered and a steel utilization between 0.2 and 0.8 for all elements.

Overall, the findings emphasize that the selection of reclaimed steel profiles is influenced by both the dataset characteristics and the structural configuration. While the optimization model successfully reduced embodied carbon across all scenarios, the greatest impacts were observed when datasets aligned closely with the design requirements. These insights underscore the importance of tailoring reclaimed stock to anticipated design needs and refining predictive models for more complex structural systems.

9

Discussion

This chapter analyses the broader implications of the research findings presented in the previous chapters, focusing on the integration of reclaimed steel into data centre design through the application of machine learning and optimization models. By interpreting the results of the RNN models and the Grasshopper optimization model, this discussion aims to review the significance of these findings within the field of sustainable construction. Additionally, it addresses the limitations encountered during the study and proposes directions for future research to enhance the applicability and effectiveness of the proposed methodologies. Through this analysis, the chapter seeks to highlight the potential of advanced computational techniques in promoting environmentally responsible design practices.

9.1. Research Implications

This research provides a significant proof of concept for the application of machine learning in the automated design of data centre structures, achieving improvements in both structural efficiency and environmental performance. The Recurrent Neural Network models developed for this study demonstrated high predictive accuracy for braced structures, achieving weighted F1-scores of 0.9350 (IPEb) and 0.9335 (HEAb). While moment-fixed systems resulted in lower accuracies, with scores of 0.8822 (HEAm) and 0.8202 (IPEm), the model effectively captured critical patterns within the complex configurations, providing a foundation for further development. The lower accuracy for moment-fixed systems suggests that additional input features, such as beam-to-column connections, could improve the model's ability to account for more complex structural behaviours.

The integration of these RNN predictions into a Grasshopper workflow demonstrated how reuse-focused optimization can align reclaimed steel stock with design needs. The Generated Dataset achieved reuse rates exceeding 90%, reducing CO_2 emissions by up to 62.5%. In contrast, the Existing Designs dataset showed reuse percentages of 35.4% for IPE beams and 51.7% for HEB columns, while HEA configurations reached much lower rates (1.9% and 2.7%). These findings confirm that alignment between available stock and design requirements is crucial for maximizing reuse potential.

Validation of the generated structure further confirmed its applicability to practical design scenarios. Stability systems were applied correctly and all structural elements achieved utilization levels between 0.2 and 0.8, demonstrating a balanced design approach that meets both structural and environmental targets. By combining the automated cross-section selection, component matcher and LCA calculations, this model positions itself as a valuable approach for automated decision-making, enabling environmentally conscious design within tight project timelines.

The established workflow in this study shows the potential of integrating ML and Grasshopper for automated design processes, presenting a workflow that includes data collection, ML-driven predictions and design optimization within a user-friendly interface. Grasshopper's parametric environment paired with the predictive capabilities of an ML model demonstrates how powerful tools can work together to not only optimize for reclaimed materials but also streamline the design process. The interaction between Grasshopper's geometric flexibility and ML's adaptability highlights a pathway for further innovations in building design, enabling a workflow that is both technically robust and accessible to a broad spectrum of design professionals.

9.2. Research Limitations

A key limitation of the proposed workflow lies in the uncertainties associated with the predictions for cross-section selection. The RNN model, while powerful, is ultimately data-dependent and includes prediction uncertainties, especially given the complexity and variability inherent in reclaimed steel sections. These uncertainties pose a risk, as incorrect predictions could result in structural designs that fail to meet necessary safety and performance criteria. Therefore, this workflow should be viewed as a preliminary indication of reuse potential rather than a final design configuration. Practitioners are advised to use these results as a guide, conducting further verification steps and cross-referencing with traditional structural analysis methods to ensure the safety and reliability of the final design.

The model's precision is also impacted by non-valid element values (N.A values) within the dataset. The elements that could not be designed under the given load and length combination, are set to N.A. One could argue that if an element within the design is not applicable, that means that the entire design configuration for that specific combination of design choices is insufficient and should be neglected.

Another limitation is the specificity of this research to data centre design, particularly looking at the data halls within those design types. While this focused scope allows for a better insight into the application, it restricts the workflow's direct applicability to other types of structures. The methods, datasets, and parameters were curated with data centres in mind; applying the model to other building types would require replicating the entire process, including data gathering, ML training, and validation. Although the workflow could theoretically be adapted to other applications, these structural contexts may introduce new variables and demands that the current model is not optimized to handle.

9.3. Future Research

The following five potentials for future research have been identified:

Incorporating prediction uncertainty in Grasshopper:
A valuable extension of this workflow would be the ability to incorporate the prediction uncertainty for each optimized model directly within Grasshopper. Since the reuse potential currently reflects a percentage of reused elements in the design, it would be beneficial to also display the model's prediction accuracy for cross-section selections. By integrating this uncertainty score, designers would gain a clearer understanding of the reliability of reuse recommendations, adding a new layer of transparency to the results. This feature could allow users to weigh reuse potential more effectively, particularly in critical applications further into the design process.

Enhancing the RNN model:
Further improvement of the ML model could be achieved by restructuring the N.A values within the RNN model. By creating a first entry that classifies if a model is correct, the result will have an initial indication of N.A elements are present within the design configuration. Additionally, introducing additional design features focussed on the moment-fixed connections would improve the accuracy of

the models trained for that stability system. An example of such an added feature could be linking the element types of the columns to the number of beams connected to it instead of only saving if it is an edge or a middle column. By updating the model's focus on these element-specific characteristics, future iterations could result in more refined predictions, especially for unique or challenging structural components.

Expanding the dataset:
The development of a larger, more diverse dataset could substantially enhance the performance of the ML model. Generating additional configurations of valid designs could improve the model's ability to generalize across various scenarios, particularly for those less represented in the current dataset. Expanding data volume would likely lead to more accurate predictions and greater resilience in model performance across a broader range of design constraints.

Combining the separate workflows into one ML model:
An intriguing direction for future research is the integration of optimization within the RNN model itself, streamlining the process by combining design configuration and reclaimed steel databases as inputs. This approach would allow the model to output both the reuse potential percentage and associated uncertainty, creating a complete model capable of evaluating reuse potential without a separate optimization step in Grasshopper. This integrated model could simplify the workflow and potentially increase the efficiency and accessibility of the design process by consolidating prediction and optimization into one single framework.

Application to new design aspects and building types:
Expanding this workflow to incorporate additional design aspects or to address different structural types would broaden its applicability and reveal new insights into the versatility of ML-driven, reuse-focused design. Testing the model on various building types or applying it to different aspects of structural design, such as floor systems or alternative material applications, could lead to innovations in other areas of sustainable design. Comparative studies on these different applications would also provide valuable information on the adaptability and scalability of this workflow, showcasing the potential to apply similar methods in a wider range of architectural and engineering contexts.

10

Conclusion

In this final chapter, the core findings from each sub-question are summarized to provide a comprehensive answer to the main research question:

"How can the incorporation of reclaimed steel elements be made more accessible within the design process of datacentres when using Machine Learning applications?"

This summary highlights the impact of steel reuse, the availability and structural properties of reclaimed elements, the potential design methodologies for integrating stock-constrained elements and the role of machine learning (ML) in optimizing configurations that leverage reclaimed materials. This chapter is structured based on the research sub-questions.

1. What effect has the integration of reclaimed steel on the design approach and sustainability calculations of data centres?

The application of reclaimed steel has demonstrated significant environmental advantages, particularly in terms of reducing carbon emissions throughout the material's life cycle. According to the LCA model outlined by Broniewicz and Dec (2022), reused steel reduces emissions across key life cycle stages, such as the extraction and manufacturing phases (A1-A3) and the end-of-life stage (C1-C4), where steel reuse minimizes the need for new material production. Module D of the LCA framework, which covers potential avoided environmental impacts, plays a critical role in quantifying the benefits of steel reuse. Studies such as those by Buzatu et al. (2023) further highlight the emissions reduction potential of reused steel, with estimates ranging from 29-35% lower emissions than new steel. Tools like Steel-IT developed by Aardoom (2023), make it easier to visualize these benefits, aiding decision-makers in realizing substantial environmental gains while ensuring that reclaimed steel remains cost-effective. This emphasis on life cycle impacts demonstrates that reclaimed steel in building design not only contributes to immediate reductions in greenhouse gas emissions but also supports a broader, long-term environmental strategy.

2. What stock-constrained automated design processes have been researched and which apply best within this application?

To address the second research question on stock-constrained automated design processes, five key studies offer insights into the integration of stock availability in structural design. Bukauskas (2020) introduced foundational concepts for reuse in structural design, defining "Assignment" in a binary matrix to indicate stock matches, and "Cut-off Ratio" to show required modifications. Building on this, Brütting et al. (2020) linked moment capacities of stock elements to structural load cases using a branch-and-bound method. However, this approach proved slow due to the recalculation needed for each cut-off

length. To address computation time, Van Marcke et al. (2024) employed a generative algorithm paired with finite element analysis, focusing on maximizing stock application by optimizing truss structures and element aggregation. Haakonsen et al. (2024) adapted this algorithm to Grasshopper, improving user accessibility by visualizing stock assignment for structural design in a two-storey timber building. Lastly, Rademaker (2022) presented a Python-based model for reusable stock assignment, achieving significant reductions in new steel usage with custom optimization constraints, though yielding lower utilization coefficients.

These studies highlight diverse methodologies for stock-constrained design, each with unique strategies for minimizing environmental impact through reuse. However, the full-scale application to a building-level design, specifically a data centre, was determined as a research gap.

3. What machine learning applications that assist in the structural design process have been researched?

Machine learning has shown significant potential in advancing the structural design process by using data more effectively and enabling automated decision-making. Research by Liao et al. (2024) emphasizes the transformative role of models like generative adversarial networks (GANs) and graph neural networks (GNNs), which excel in optimizing component layouts and cross-sectional sizes based on diverse design data. Practical applications include Pizarro and Massone (2021)'s deep neural network (DNN), which predicts the dimensions of reinforced concrete shear walls with remarkable accuracy, and Potijk (2024)'s artificial neural network (ANN), which simplifies comparisons of structural and environmental costs for different stability systems. Additionally, Fenton et al. (2024) highlight the growing integration of sustainability metrics, using machine learning to predict embodied greenhouse gas emissions early in the design process. Further possibilities lie in recurrent neural networks (RNNs), which are particularly suited for sequential design tasks and the reuse of structural elements, offering a promising topic for both efficiency and sustainability in structural engineering.

4. How can machine learning techniques be applied to effectively incorporate reclaimed steel within the design process?

To answer the fourth research question, an RNN model was set up as a proof of concept to show the application of machine learning on cross-section selection. Data processing techniques such as normalization, one-hot encoding, and padding for variable sequence lengths optimize the model's adaptability, while bidirectional LSTM layers capture dependencies in between the design configurations. The final trained model showed accuracies ranging from 82 to 92% with accurate predictions as seen on the confusion matrices for HEA and IPE profiles, but HEB profile predictions showed less promising results.

5. What are the key optimization variables for incorporating reclaimed steel into a design?

The final sub-question combines the design aspects determined by the Grasshopper optimization model and building-type specific design requirements. The developed optimization model initiates with grid-spacings determining the column placement in the floorplan. With the help of the RNN for cross-section selection predictions, the model can be automatically generated, for which a comparative LCA calculation was done resulting in the total embodied carbon based on the percentage of reuse applied. Since data halls from data centres perform significantly better when fewer columns are located in the floorplan, a customized metric is set up which combines the embodied carbon with the data centre design requirements. To conclude, the final metric to minimize with the optimization algorithm is the embodied carbon per data rack present in the floorplan.

In answering the main research question, this research has successfully demonstrated a practical application of machine learning to streamline the incorporation of reclaimed steel within data centre design. The combination of an RNN model for cross-section prediction and a customized optimization model in Grasshopper effectively reduces the manual input and expertise typically required to match reclaimed materials to specific structural demands. This model-driven approach not only simplifies the selection and integration of reclaimed steel, enhancing accessibility within the design process but also aligns with industry goals of minimizing embodied carbon. While this framework shows strong potential, further refinement could enhance its effectiveness. Overall, the research provides a significant step toward automated, resource-conscious structural design, offering a scalable solution for sustainable data centre development.

Bibliography

15804+A2, N. (2019). Sustainability of construction works - environmental product declarations - core rules for the product category of construction products. https://connect.nen.nl/Standard/Detail/3621354?compId=10037&collectionId=0

1990:2021, N. (2021). NEN connect - eurocode: Grondslagen van het constructief ontwerp. Retrieved December 5, 2024, from https://connect.nen.nl/Standard/Detail/3649766?compId=10037&collectionId=0

1993-1-1:2006, N. (2020). NEN connect - eurocode 3: Ontwerp en berekening van staalconstructies - deel 1-1: Algemene regels en regels voor gebouwen. Retrieved October 23, 2024, from https://connect.nen.nl/Standard/PopUpHtml?RNR=44904&search=&Native=1&token=f6bcd4f2-3418-4e33-b95f-7172711960de

50600-2-1, N. (2021). Information technology - data centre facilities and infrastructures - part 2-1: Building construction. https://connect.nen.nl/Standard/Detail/3645086?compId=10037&collectionId=0

Aardoom, K. (2023). The business-case and environmental impact score for the preliminary structural design of a reclaimed steel low-rise office building: Developing a tool that generates and evaluates reclaimed steel design alternatives. Retrieved March 11, 2024, from https://repository.tudelft.nl/islandora/object/uuid%3A74a879f1-ff0d-4db3-a990-38ddd0fe32c8

Alaux, N., Marton, C., Steinmann, J., Maierhofer, D., Mastrucci, A., Petrou, D., Potrč Obrecht, T., Ramon, D., Le Den, X., Allacker, K., Passer, A., & Röck, M. (2024). Whole-life greenhouse gas emission reduction and removal strategies for buildings: Impacts and diffusion potentials across EU member states. *Journal of Environmental Management*, *370*, 122915. https://doi.org/10.1016/j.jenvman.2024.122915

Ankalaki, S. (2022). A customized 1d-CNN approach for sensor-based human activity recognition. *International Journal of Advanced Technology and Engineering Exploration*, *9*, 216–231. https://doi.org/10.19101/IJATEE.2021.874828

Baheti, P. (2022, June 29). *The complete guide to recurrent neural networks*. Retrieved October 24, 2024, from https://www.v7labs.com/blog/recurrent-neural-networks-guide,%20https://www.v7labs.com/blog/recurrent-neural-networks-guide

Bakker, H., & de Kleijn, J. (2014). *Management of engineering projects | people are key*. NAP Process Industry Network.

Beschikbaarheid en process hergebruik constructiestaal. interview by tessel van oers. (2024, June 17).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer. https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

Bleker, L., Tam, K.-M. M., & D'Acunto, P. (2024). A graph-based grammar for structural design using deep reinforcement learning. Retrieved December 6, 2024, from https://mediatum.ub.tum.de/1754110

Broer, R., Simjanovic, J., Toth, Z., Caldas, L. R., Belizário, F., & John, V. (2022). IMPLEMENTING THE PARIS AGREEMENT AND REDUCING GREENHOUSE GAS EMISSIONS THROUGHOUT THE LIFE CYCLE OF BUILDINGS: https://www.bpie.eu/wp-content/uploads/2022/01/SPIPA-LCA-2022FINAL.pdf

Broniewicz, E., & Dec, K. (2022). Environmental impact of demolishing a steel structure design for disassembly [Number: 19 Publisher: Multidisciplinary Digital Publishing Institute]. *Energies*, *15*(19), 7358. https://doi.org/10.3390/en15197358

Brütting, J., Senatore, G., Schevenels, M., & Fivet, C. (2020). Optimum design of frame structures from a stock of reclaimed elements [Publisher: Frontiers]. *Frontiers in Built Environment*, *6*. https://doi.org/10.3389/fbuil.2020.00057

Bukauskas, A. (2020). Inventory-constrained structural design. *University of Bath*. Retrieved June 26, 2024, from https://researchportal.bath.ac.uk/en/studentTheses/inventory-constrained-structural-design

Buzatu, R., Ungureanu, V., & Hradil, P. (2023). Environmental and economic impact of steel industrial buildings made of reclaimed elements [Num Pages: 9]. In *Life-cycle of structures and infrastructure systems*. CRC Press.

den Hollander, J. (2024). Wat is er aan de hand met module d? *Bouwen met Staal*, *57*(299), 20–23. https://vakbladbouwenmetstaal.nl/archief/wat-is-er-aan-de-hand-met-module-d

Dolan, C. W., & Hamilton, H. R. ((2019). Two-way slabs. In C. W. Dolan & H. R. (Hamilton (Eds.), *Prestressed concrete: Building, design, and construction* (pp. 301–329). Springer International Publishing. https://doi.org/10.1007/978-3-319-97882-6_11

EurocodeApplied. (2017). *Table of properties for IPE,HEA,HEB,HEM,UB,UC,UBP profiles - eurocode 3* [EurocodeApplied.com]. Retrieved October 23, 2024, from https://eurocodeapplied.com/design/en1993/ipe-hea-heb-hem-design-properties

EuropeanCommission. (2020, March 4). *Proposal for a regulation of the european parliament and the council, establishing the framework for achieving climate neutrality and amending regulation (EU) 2018/1999 (european climate law)*. CLIMA. Brussels. Retrieved June 26, 2024, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020PC0080&qid=1719394579190

Fenton, S. K., Munteanu, A., De Rycke, K., & De Laet, L. (2024). Embodied greenhouse gas emissions of buildings—machine learning approach for early stage prediction. *Building and Environment*, *257*, 111523. https://doi.org/10.1016/j.buildenv.2024.111523

Fuhrimann, L., Moosavi, V., Ohlbrock, P. O., & Dacunto, P. (2018, September 23). Data-driven design: Exploring new structural forms using machine learning and graphic statics. https://doi.org/10.48550/arXiv.1809.08660

GBW. (2024). *Stalen profielen* [Gebruikte Bouwmaterialen Weert]. Retrieved October 23, 2024, from https://gebruiktebouwmaterialenweert.nl/product/stalen-profielen/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/

Gordon, M., Batallé, A., De Wolf, C., Sollazzo, A., Dubor, A., & Wang, T. (2023). Automating building element detection for deconstruction planning and material reuse: A case study. *Automation in Construction*, *146*, 104697. https://doi.org/10.1016/j.autcon.2022.104697

Haakonsen, S. M., Tomczak, A., Izumi, B., & Luczkowski, M. (2024). Automation of circular design: A timber building case study [Publisher: SAGE Publications]. *International Journal of Architectural Computing*, 14780771241234447. https://doi.org/10.1177/14780771241234447

Hrivnak, J. (2023, January 19). *How to calculate embodied carbon for the 2030 climate challenge*. Retrieved October 26, 2024, from https://www.ribaj.com/intelligence/how-to-calculate-embodied-carbon-for-riba-2030-climate-challenge

inc., T. (2023, October 19). *(6) recycled steel in construction: Strength, sustainability, and savings | LinkedIn*. Retrieved March 19, 2024, from https://www.linkedin.com/pulse/recycled-steel-construction-strength-sustainability-sfkfe/

Jayaswal, V. (2020, September 15). *Performance metrics: Confusion matrix, precision, recall, and f1 score* [Medium]. Retrieved October 31, 2024, from https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262

Jones, N. (2018). How to stop data centres from gobbling up the world's electricity [Bandiera_abtest: a Cg_type: News Feature Publisher: Nature Publishing Group Subject_term: Energy, Engineering, Research data, Computer science]. *Nature*, *561*(7722), 163–166. https://doi.org/10.1038/d41586-018-06610-y

Karamba3D. (2024, October 8). *3.6.8: Optimize cross section | karamba3d v3*. Retrieved October 23, 2024, from https://manual.karamba3d.com/3-in-depth-component-reference/3.5-algorithms/3.5.8-optimize-cross-section

Kavoura, F., & Veljkovic, M. (2023). Design strategies for reusable structural components in the built environment: IALCCE 2023 (F. Biondini & D. M. Frangopol, Eds.) [Publisher: Taylor & Francis]. *Life-Cycle of Structures and Infrastructure Systems - Proceedings of the 8th International Symposium on Life-Cycle Civil Engineering, IALCCE 2023*, 799–806. https://doi.org/10.1201/9781003323020-97

Keranis, A., Mantas, D., & van Oers, T. (2023, November). *Definition and allocation of departmental tasks towards a circular design and construction workflow*. Joint Interdisciplinary Project (JIP).

Keras, T. (2024). *Keras documentation: ReduceLROnPlateau*. Retrieved October 25, 2024, from https://keras.io/api/callbacks/reduce_lr_on_plateau/

Kingma, D. P., & Ba, J. (2017, January 29). Adam: A method for stochastic optimization. Retrieved September 12, 2024, from http://arxiv.org/abs/1412.6980

Kraaijenbrink, R., Levels-Vermeer, J., & van Nunen, D. (2022a). DFR: Zwaar constructiestaal, design for reuse, overspanning tot 25m, 80% hergebruik. *SNS / Bouwen met Staal*. https://www.mrpi.nl/epd-files/epd/1.1.00288.2022_MRPI-EPD_Zwaar%20constructiestaal-%2016Procent%20hergebruik%20EoL_FINAL.pdf

Kraaijenbrink, R., Levels-Vermeer, J., & van Nunen, D. (2022b). Hergebruik: Zwaar constructiestaal uit 90% hergebruik, 16% hergebruik einde leven. *SNS / Bouwen met Staal*. https://www.mrpi.nl/epd-files/epd/1.1.00291.2022_MRPI-EPD_Zwaar%20constructiestaal%20uit%2090Procent%20hergebruik-%2016Procent%20hergebruik%20EoL_FINAL.pdf

Kraaijenbrink, R., Levels-Vermeer, J., & van Nunen, D. (2022c). Zwaar constructiestaal, 16% hergebruik einde leven. *SNS / Bouwen met Staal*. https://www.mrpi.nl/epd-files/epd/1.1.00288.2022_MRPI-EPD_Zwaar%20constructiestaal-%2016Procent%20hergebruik%20EoL_FINAL.pdf

Le Den, X., Steinmann, J., Kockat, J., Zsolt, T., Röck, M., Allacker, K., Kovacs, A., BPIE, Directorate-General for Environment (European Commission), KU Leuven, & Ramboll. (2023). *Supporting the development of a roadmap for the reduction of whole life carbon of buildings: Final report*. Publications Office of the European Union. Retrieved October 23, 2024, from https://data.europa.eu/doi/10.2779/634412

Liao, W., Lu, X., Fei, Y., Gu, Y., & Huang, Y. (2024). Generative AI design for building structures. *Automation in Construction*, *157*, 105187. https://doi.org/10.1016/j.autcon.2023.105187

Lut, S., Titulaer, R., Reale, V., & Slangen, S. (2022, March 23). *Arup-group/magpie: Match reclaimed steel elements with a design.* [GitHub]. Retrieved June 24, 2024, from https://github.com/arup-group/magpie

Makki, M., Showkatbakhsh, M., & Song, Y. (2022). *Evolutionary engine for grasshopper3d* [Wallacei]. Retrieved July 2, 2024, from https://www.wallacei.com

Memarian, B., & Doleck, T. (2024). A scoping review of reinforcement learning in education. *Computers and Education Open*, *6*, 100175. https://doi.org/10.1016/j.caeo.2024.100175

Mirra, G., Holland, A., Roudavski, S., Wijnands, J. S., & Pugnale, A. (2022). An artificial intelligence agent that synthesises visual abstractions of natural forms to support the design of human-made

habitat structures [Publisher: Frontiers]. *Frontiers in Ecology and Evolution*, *10*. https://doi.org/10.3389/fevo.2022.806453

Mirra, G., & Pugnale, A. (2023). Enhancing interactivity in structural optimisation through reinforcement learning: An application on shell structures. *Proceedings of IASS Annual Symposia*, *2023*(8), 1–12.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines, 807–814. https://doi.org/10.5555/3104322.3104425

Nature. (2023). *The impact of AI research | nature research intelligence*. Retrieved July 4, 2024, from https://www.nature.com/research-intelligence/ai-research-report

NTA 8713, N. (2023). Reuse of structural steelwork. Retrieved March 11, 2024, from https://connect.nen.nl/Standard/Detail/3687533?compId=10037&collectionId=0

Oasys, G. (2024). *GSA theory | oasys GSA documentation* [Docs oasys software]. Retrieved June 24, 2024, from https://docs.oasys-software.com/structural/gsa/references-theory/

Opalis.eu. (2024). *Dealers list | opalis*. Retrieved October 23, 2024, from https://opalis.eu/en/dealers/list?f%5B0%5D=materials%3A549

Pizarro, P. N., & Massone, L. M. (2021). Structural design of reinforced concrete buildings based on deep neural networks. *Engineering Structures*, *241*, 112377. https://doi.org/10.1016/j.engstruct.2021.112377

Potijk, E. (2024). Predicting the best stability system for high-rise steel buildings by an artificial neural network. Retrieved June 24, 2024, from https://repository.tudelft.nl/islandora/object/uuid%3Aeeb75e45-a7a2-4418-857b-72ab3f388e66

Prechelt, L. (2012). Early stopping — but when? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 53–67). Springer. https://doi.org/10.1007/978-3-642-35289-8_5

Preisinger, C. (2013). Linking structure and parametric geometry. *Architectural Design*, *83*. https://doi.org/10.1002/ad.1564

Prince, S. J. (2023). *Understanding deep learning*. The MIT Press. Retrieved June 24, 2024, from http://udlbook.com

Rademaker, G. (2022). Balancing design and circularity: Optimizing the reuse of steel elements in the design of frame structures. Retrieved June 26, 2024, from https://repository.tudelft.nl/islandora/object/uuid%3A7ccb8b38-6a93-4bac-aec2-dc2745ea1f8a

Rakhshan, K., Morel, J.-C., Alaka, H., & Charef, R. (2020). Components reuse in the building sector – a systematic review. *Waste Management & Research*, *38*, 347–370. https://doi.org/10.1177/0734242X20910463

Rasmussen, N., & Torell, W. (2015). Data center projects: Establishing a floor plan. *White Paper 144*, *V2*. https://www.mercurymagazines.com/pdf/APC3.pdf

Rathor, S. (2018, June 2). *Simple RNN vs GRU vs LSTM :- difference lies in more flexible control* [Medium]. Retrieved October 26, 2024, from https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57

Rechenraum, G. (2023). *Rechenraum GmbH - goat* [Goat]. Retrieved September 10, 2024, from https://www.rechenraum.com/en/goat.html

Shan, R. (2014, July 27). *Integrating genetic algorithm with rhinoceros and grasshopper in whole building energy simulation*. University of Michigan. https://www.researchgate.net/publication/267980610_Integrating_Genetic_Algorithm_with_Rhinoceros_and_Grasshopper_in_Whole_Building_Energy_Simulation

Snellen. (2024). *Tweedehands ijzerwaren - Tweedehandsmaterialen - Online bestellen* [Tweedehandsmaterialen]. Retrieved October 23, 2024, from https://www.tweedehandsmaterialen.nl/product-categorie/ijzerwaren/

Tait, T. (2024, April 12). *Galapagos in grasshopper: Step-by-step guide to the evolutionary solver* [Hopific]. Retrieved July 2, 2024, from https://hopific.com/galapagos-grasshopper-tutorial/

UNEP. (2022, November 9). *2022 global status report for buildings and construction: Towards a zero emission, efficient and resilient buildings and construction sector* (No. 2022). United Nations Environment Programme. Nairobi. Retrieved December 3, 2024, from https://globalabc.org/resources/publications/2022-global-status-report-buildings-and-construction

Van Marcke, A., Laghi, V., & Carstensen, J. V. (2024). Automated planar truss design with reclaimed partially disassembled steel truss components. *Journal of Building Engineering*, *84*, 108458. https://doi.org/10.1016/j.jobe.2024.108458

Van Noorden, R., & Perkel, J. M. (2023). AI and science: What 1,600 researchers think [Bandiera_abtest: a Cg_type: News Feature Publisher: Nature Publishing Group Subject_term: Machine learning, Mathematics and computing, Technology, Computer science]. *Nature*, *621*(7980), 672–675. https://doi.org/10.1038/d41586-023-02980-0

van Liempd, A. (2024). *Gebruikte stalen constructiebalken | gebruiktebouwmaterialen.com*. Retrieved October 23, 2024, from https://gebruiktebouwmaterialen.com/assortiment/ijzerwaren/constructiebalken.html

van Maastrigt, J. (2019). Quantifying life cycle environmental benefits of circular steel building designs: Development of an environmental assessment tool for reuse of steel members in building designs for the netherlands. Retrieved April 9, 2024, from https://repository.tudelft.nl/islandora/object/uuid%3Acf8f1434-ce13-41cd-a1c6-25270c46af68

Vergoossen, R. P. H., Eck, G. J. v., & Jilissen, D. H. J. M. (2023). Re-use of existing load-bearing structural components in new design [Num Pages: 7]. In *Life-cycle of structures and infrastructure systems*. CRC Press.

Voorraad constructiestaal. interview by tessel van oers. (2024, May 9).

Wortmann, T. (2017). *Opossum | institute for computational design and construction | university of stuttgart* [Uni-stuttgard]. Retrieved July 2, 2024, from https://www.icd.uni-stuttgart.de/research/research-tools/opossum/

A

Structural Loads and Steel Resistance

Table A.1: Participation factors from Eurocode

Load	ψ_0	ψ_1	ψ_2
Cat E2: industrial	1.0	0.9	0.8
Wind	0	0.2	0
Snow	0	0.2	0

Table A.2: ULS load factors for consequence class CC2

ULS	Imposed load	Governing live load	Other live loads
ULS 1	$1.35 \cdot G$	-	$1.5 \cdot \psi_{0,i} Q_{k,i}$
ULS 2	$1.2 \cdot G$	$1.5 \cdot Q_{k,1}$	$1.5 \cdot \psi_{0,i} Q_{k,i}$

Within the calculations, only plastic design is considered as the assumed cross-section class is 1 or 2. The buckling curves are determined by EurocodeApplied (2017).

Bending resistance [1993-1-1:2006 (2020) 6.2.5]

$$\frac{M_{Ed}}{M_{c,Rd}} \leq 1.0$$

$$M_{c,Rd} = M_{pl,Rd} = W_{pl} f_y$$

Compression [1993-1-1:2006 (2020) 6.2.4]

$$\frac{N_{Ed}}{N_{c,Rd}} \leq 1.0$$

$$N_{c,Rd} = A f_y$$

Combination Axial and bending [1993-1-1:2006 (2020) 6.2.9]

$$\frac{M_{Ed}}{M_{N,y,Rd}} \leq 1.0$$
$$M_{N,y,Rd} = M_{pl,y,Rd}\frac{1-n}{1-0.5a}$$
$$n = \frac{N_{Ed}}{N_{c,Rd}}$$
$$a = \frac{A-2bt_f}{A}$$

Buckling [1993-1-1:2006 (2020) 6.3.1]

$$\frac{N_{Ed}}{N_{b,Rd}} \leq 1.0$$
$$N_{b,Rd} = \chi A F_y$$
$$\chi = \frac{1}{\phi + \sqrt{\phi^2 - \bar{\lambda}^2}}$$
$$\phi = 0.5[1 + \alpha(\bar{\lambda} - 0.2) + \bar{\lambda}^2]$$
$$\bar{\lambda} = \sqrt{\frac{Af_y}{N_{cr}}}$$
$$N_{cr} = \frac{\pi^2 EI}{L_b^2}$$
$$\text{Bracing} : L_b = L \vee \text{Moment} : L_b = 2L$$

Lateral Torsional Buckling [1993-1-1:2006 (2020) 6.3.2]

$$\frac{M_{Ed}}{M_{b,Rd}} \leq 1.0$$
$$M_{b,Rd} = \chi_{LT} W_{pl,y} f_y$$
$$\chi_{LT} = \frac{1}{\phi_{LT} + \sqrt{\phi_{LT}^2 - \bar{\lambda}_{LT}^2}}$$
$$\phi_{LT} = 0.5[1 + \alpha_{LT}(\bar{\lambda_{LT}} - 0.2) + \bar{\lambda}_{LT}^2]$$
$$\bar{\lambda}_{LT} = \sqrt{\frac{Af_y}{M_{cr}}}$$
$$M_{cr} = \frac{C}{L}\sqrt{EI_z GI_t}$$

Deflection [1990:2021 (2021) Table A.1.10]

$$w_{max} \leq \frac{L}{250}$$

B

Reclaimed Element Database

Dataset_gen

October 27, 2024

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from matplotlib.ticker import FixedLocator
```

Generating the reuse dataset

First all existing cross-sections are loaded in from the website https://eurocodeapplied.com/design/en1993/ipe-hea-heb-hem-design-properties. Then the minimum and maximum length are set using the eurocode and adapted using the ranges optained by interviewing Vic Obdam and Bouwen met Staal. first the number of sections with a certain profile is picked following a gaussian distribution. The accompanying lengths are then picked randomly from a uniform distribution in the given interval. This notebook ends with some visualisations of the dataset.

[ ]:
```
data = pd.read_csv('CS_info_total.csv', skiprows=[0, 1, 3, 4], usecols=[0, 1])
data.rename(columns={'Unnamed: 0': 'Steel Profile', 'h': 'height'}, inplace=True)
data.head()
```

[ ]:
```
def Quick_reference(df, QRmin, QRmax, Min, maxMin, Max):
    df['QRlength_min'] = df['height'] * QRmin / 1000
    df['QRlength_max'] = df['height'] * QRmax / 1000
    df['QRlength_min'] = df['QRlength_min'].apply(lambda x: Min if x < Min else␣
 ↪x)
    df['QRlength_min'] = df['QRlength_min'].apply(lambda x: maxMin if x > maxMin␣
 ↪else x)
    df['QRlength_max'] = df['QRlength_max'].apply(lambda x: Max if x > Max else␣
 ↪x)
    return df

def Gaussian_counts(df, counts, std, mean_loc):
    num_rows = len(df)
    mean = num_rows * mean_loc  # Mean value (somewhere around HEA300, HEB300␣
 ↪and IPE240)
    std_dev = num_rows / std  # Standard deviation
    x = np.arange(num_rows)
```

```
    gaussian_values = np.exp(-0.5 * ((x - mean) / std_dev) ** 2)

    gaussian_values = gaussian_values / gaussian_values.sum() * counts
    gaussian_values = np.round(gaussian_values).astype(int)

    df['Gaussian_count'] = gaussian_values
    return df

def Extend_dataset(df, extended_df):
    for index, row in df.iterrows():
        # Convert Gaussian_count to an integer
        count = int(row['Gaussian_count'])

        # Generate random lengths within the specified range
        lengths = np.random.uniform(row['QRlength_min'], row['QRlength_max'],␣
 ↪count)

        # Round the lengths to one decimal point
        lengths = np.round(lengths, 1)

        # Create a DataFrame for the current profile with the generated lengths
        temp_df = pd.DataFrame({
            'Steel Profile': [index] * count,
            'height': [row['height']] * count,
            'Length': lengths
        })

        # Append the temporary DataFrame to the extended DataFrame
        extended_df = pd.concat([extended_df, temp_df], ignore_index=True)

    # Reset the index to create a new simple numeric index
    extended_df.reset_index(drop=True, inplace=True)
    return extended_df
```

[ ]:
```
HEA = data[data['Steel Profile'].str.startswith('HEA')].set_index('Steel␣
 ↪Profile')
HEB = data[data['Steel Profile'].str.startswith('HEB')].set_index('Steel␣
 ↪Profile')
IPE = data[data['Steel Profile'].str.startswith('IPE')].set_index('Steel␣
 ↪Profile')

# Initial design rules-of-thump from the Eurocode and bounding boxes for each␣
 ↪type
Quick_reference(HEA, 18, 28, 2, 6, 15)
Quick_reference(HEB, 20, 30, 3, 8, 16)
Quick_reference(IPE, 15, 25, 2, 5, 12)
```

```
# Mean and standard deviation per profile type
Gaussian_counts(HEA, 500, 2, 1/3)
Gaussian_counts(HEB, 250, 2, 1/3)
Gaussian_counts(IPE, 250, 2, 1/2)

combined_df = pd.concat([HEA.reset_index(), HEB.reset_index(), IPE.
 ↪reset_index()], axis=1)
combined_df = combined_df.reset_index(drop=True)
display(combined_df)
```

[ ]:
```
# Initialize an empty DataFrame to store the extended data
Dataset_gen = pd.DataFrame()

Dataset_gen = Extend_dataset(HEA, Dataset_gen)
Dataset_gen = Extend_dataset(HEB, Dataset_gen)
Dataset_gen = Extend_dataset(IPE, Dataset_gen)

Dataset_gen['ID'] = [f'database_element_{i+1}' for i in range(len(Dataset_gen))]
Dataset_gen['Material Type'] = 'S235'
# Dataset_gen.to_csv('Gen_dataBASE_small.csv', index=False, header=True)
display(Dataset_gen)
```

Visualisations

[ ]:
```
data2 = Dataset_gen
IPE = data2[data2['Steel Profile'].str.startswith('IPE')]
HEA = data2[data2['Steel Profile'].str.startswith('HEA')]
HEB = data2[data2['Steel Profile'].str.startswith('HEB')]
```

[ ]:
```
# Data and titles
datasets = [HEA, HEB, IPE]
titles = ['HEA', 'HEB', 'IPE']
colors = ['blue', 'orange', 'green']

plt.figure(figsize=(15, 13))
plt.suptitle('Cross-Sections and Lengths')

for i, (df, title, color) in enumerate(zip(datasets, titles, colors), start=1):
    ax1 = plt.subplot(3, 1, i)
    plt.title(f'{title} profiles and lengths, total size = {len(df)}')
    sns.boxplot(data=df, x='Steel Profile', y='Length', color=color, ax=ax1)
    ax2 = ax1.twinx()
    ax2.plot(df['Steel Profile'].value_counts(sort=False), color='r',␣
 ↪label=f'{title.split()} profile count')
    ax2.set_yticks(np.arange(0, 35, 5))
    ax2.set_ylabel('Counts')
    ax2.legend()
```

```
    ax1.set_yticks(np.arange(0, 21))
    ax1.xaxis.set_major_locator(FixedLocator(ax1.get_xticks()))
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=45)
    ax1.grid(True, axis='y', color='whitesmoke')

plt.tight_layout()
# plt.savefig('Gen_dataset.png', bbox_inches='tight')
plt.show()
```

C

Cross-Section Selection Validation

Cross-section selection verification

Tessel van Oers - 4953479

December 6, 2024

The following document shows my Python code for automatic cross-section selection based on Eurocode 0 and 3. The document has all the code highlighted in grey, which also contains the formulas used for the calculations. The moment and deflection calculation is done with a Python implementation of the Matrix-method, where the stiffness matrix is used to generate nodal loads from the elements.

The document starts with loading the data found online on the cross-sections which are considered. Next, the Matrix-method is described in three classes. Then functions are set up to calculate flexural- and lateral torsional buckling, which are used within the unity check calculations. These functions are first verified with a simply supported beam, then in a fixed 2-dimensional frame. Lastly, the code is implemented in Grasshopper. The resulting 3-dimensional structure is placed in GSA for a final steel utilisation verification.

Contents

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1 Loading cross-section data

All data is from the website: https://eurocodeapplied.com/design/en1993/ipe-hea-heb-hem-design-properties , Where steel grade S235 is chosen.

[2]:
```
data = pd.read_csv('CS_info_total.csv', skiprows=[0, 1, 3, 4], index_col=0,
                   usecols=[0, 1, 2, 4, 6, 8, 11, 15, 19, 21, 23, 27, 29, 30,␣
 ↪31])
data.rename(columns={'Unnamed: 30': 'buck_cy', 'Unnamed: 31':'buck_cz', 'IT':
 ↪'It',
                     'Npl,Rd':'Npl_rd', 'Mpl,Rd,y':'Mpl_rdy', 'Mpl,Rd,z':
 ↪'Mpl_rdz'},
            inplace=True)

# Add columns for EIy, EIz and EA
E = 210000                                  # n / mm2
data['EIy'] = data['Iy'] * E * 10**(-3)     # kNm2
data['EIz'] = data['Iz'] * E * 10**(-3)     # kNm2
data['EA'] = data['A'] * E * 10**(-3)       # kN

print('Note that height h[mm], width b[mm], self weight m[kg/m], A[mm2], ')
print('Iy and Iz [10^6 mm4] are not in units of kN and m, all other values are.')

# Split dataset into different cross-section families
IPE = data[data.index.str.startswith('IPE')]
HEA = data[data.index.str.startswith('HEA')]
HEB = data[data.index.str.startswith('HEB')]
data.head()
```

```
Note that height h[mm], width b[mm], self weight m[kg/m], A[mm2],
Iy and Iz [10^6 mm4] are not in units of kN and m, all other values are.
```

[2]:

	h	b	tf	m	A	Iy	Iz	It	Iw	Npl_rd	\
IPE80	80	46	5.2	6.0	764	0.8014	0.08489	6.727	115.1	179.62	
IPE100	100	55	5.7	8.1	1032	1.7100	0.15920	11.530	342.1	242.60	
IPE120	120	64	6.3	10.4	1321	3.1800	0.27670	16.890	872.0	310.44	
IPE140	140	73	6.9	12.9	1643	5.4100	0.44920	24.010	1951.0	386.01	
IPE160	160	82	7.4	15.8	2009	8.6900	0.68310	35.300	3889.0	472.15	

	Mpl_rdy	Mpl_rdz	buck_cy	buck_cz	EIy	EIz	EA
IPE80	5.46	1.37	a	b	168.294	17.8269	160440.0
IPE100	9.26	2.15	a	b	359.100	33.4320	216720.0
IPE120	14.27	3.19	a	b	667.800	58.1070	277410.0

```
IPE140    20.76     4.52       a       b  1136.100   94.3320  345030.0
IPE160    29.11     6.13       a       b  1824.900  143.4510  421890.0
```

2 Matrix-method in Python

To program the matrix-method in Python, three classes are set up to encompass the structure. This method makes use of local axes and combines them into two-dimensional 'global'-axes. Since the design is orthogonal, it can be transformed to three-dimensional by adding the 2D frames together orthogonally. The first class sets the nodes of the system, this is the location where loads from different two-dimensional systems can be combined. The elements are added, after which the stiffness matrix is defined to calculate the deflections and moments. Lastly, the constraints of the system are added.

2.1 Node class

Three degrees of freedom. Possibility to add loads on the node itself.

[3]:
```
class Node:
    ndof = 0
    nn   = 0

    def clear():
        Node.ndof = 0
        Node.nn = 0

    def __init__ (self, x, y):
        self.x     = x
        self.y     = y
        self.p     = np.zeros(3)

        self.dofs  = [Node.ndof, Node.ndof+1, Node.ndof+2]

        Node.ndof += 3
        Node.nn   += 1

    def add_load (self, p):
        self.p += p
```

2.2 Element class

Euler-Bernoulli beam element, accounting for only axial forces and bending. Here the section properties and loads are applied, resulting in the calculated deflection and bending moment.

[4]:
```
class Element:
    ne = 0

    def clear():
        Element.ne = 0
```

```
    def __init__ (self, nodes):
        self.nodes = nodes

        self.L = np.sqrt((nodes[1].x - nodes[0].x)**2.0
                         + (nodes[1].y - nodes[0].y)**2.0)

        dx = nodes[1].x - nodes[0].x
        dy = nodes[1].y - nodes[0].y

        self.cos = dx / self.L
        self.sin = dy / self.L

        R = np.zeros ((6,6))

        R[0,0] = R[1,1] = R[3,3] = R[4,4] = self.cos
        R[0,1] = R[3,4] = -self.sin
        R[1,0] = R[4,3] =  self.sin
        R[2,2] = R[5,5] = 1.0

        self.R  = R
        self.Rt = np.transpose(R)

        self.q = np.zeros(2)

        Element.ne += 1

    def set_section (self, props):
        if 'EA' in props:
            self.EA = props['EA']
        else:
            self.EA = 1.e20
        if 'EI' in props:
            self.EI = props['EI']
        else:
            self.EI = 1.e20

    def global_dofs  (self):
        return np.hstack ((self.nodes[0].dofs, self.nodes[1].dofs))

    def stiffness ( self ):
        k = np.zeros ((6, 6))

        EA = self.EA
        EI = self.EI
        L = self.L
```

```
        # Extension contribution

        k[0,0] = k[3,3] = EA/L
        k[3,0] = k[0,3] = -EA/L

        # Bending contribution

        k[1,1] = k[4,4] =  12.0 * EI / L / L / L
        k[1,4] = k[4,1] = -12.0 * EI / L / L / L
        k[1,2] = k[2,1] = k[1,5] = k[5,1] = -6.0 * EI / L / L
        k[2,4] = k[4,2] = k[4,5] = k[5,4] = 6.0 * EI / L / L
        k[2,2] = k[5,5] = 4.0 * EI / L
        k[2,5] = k[5,2] = 2.0 * EI / L

        return np.matmul ( np.matmul ( self.Rt, k ), self.R )

    def add_distributed_load ( self, q ):

        l = self.L

        self.q = np.array( q )

        el = [ 0.5*q[0]*l, 0.5*q[1]*l, -1.0/12.0*q[1]*l*l,
               0.5*q[0]*l, 0.5*q[1]*l, 1.0/12.0*q[1]*l*l ]

        eg = np.matmul ( self.Rt, np.array ( el ) )

        self.nodes[0].add_load ( eg[0:3] )
        self.nodes[1].add_load ( eg[3:6] )

    def bending_moments ( self, u_global, num_points=2 ):

        l = self.L
        q = self.q[1]
        EI= self.EI

        xi = np.linspace ( 0.0, l, num_points )
        M  = np.zeros(num_points)

        ul = np.matmul ( self.R, u_global )

        M = ( -l**5.0*q + 6.0 * l**4.0*q*xi
              - 6.0*q*xi*xi*l**3.0 - 48.0*(ul[2] + ul[5]/2.0)*EI*l**2.0
              + 72.0*EI*((ul[2]+ul[5])*xi+ul[1]-ul[4])*l -
              144.0*xi*EI*(ul[1]-ul[4]) ) / 12.0 / l**3.0
        w = ( l**5*q*xi**2 - 2*l**4*q*xi**3 +
              (q*xi**4 - 24*EI*ul[2]*xi + 24*EI*ul[1])*l**3
```

```
            + 48*xi**2*(ul[2] + ul[5]/2)*EI*l**2 -
            24*((ul[2] + ul[5])*xi + 3*ul[1] - 3*ul[4])*xi**2*EI*l
            + 48*xi**3*EI*(ul[1] - ul[4]))/EI/l**3/24

        return w, M
```

2.3 Constrainer class

Constrain degrees of freedom in nodes and calculate support reactions.

[5]:
```
class Constrainer:
    def __init__ (self):
        self.cons_dofs = []
        self.cons_vals = []

    def fix_dof (self, node, dof, value=0):
        self.cons_dofs.append(node.dofs[dof])
        self.cons_vals.append(value)

    def fix_node (self, node):
        for dof in node.dofs:
            self.fix_dof (node, dof)

    def full_disp (self,u_free):
        u_full = np.zeros(len(self.free_dofs) + len(self.cons_dofs))

        u_full[self.free_dofs] = u_free
        u_full[self.cons_dofs] = self.cons_vals

        return u_full

    def constrain (self, k, f):
        self.free_dofs = [i for i in range(len(f)) if i not in self.cons_dofs]

        Kff = k[np.ix_(self.free_dofs,self.free_dofs)]
        Kfc = k[np.ix_(self.free_dofs,self.cons_dofs)]
        Ff = f[self.free_dofs]

        return Kff, Ff - np.matmul(Kfc,self.cons_vals)

    def support_reactions (self,k,u_free,f):
        Kcf = k[np.ix_(self.cons_dofs,self.free_dofs)]
        Kcc = k[np.ix_(self.cons_dofs,self.cons_dofs)]

        return np.matmul(Kcf,u_free) + np.matmul(Kcc,self.cons_vals) - f[self.
↪cons_dofs]
```

3 Function set-up

A separate function is set up for the flexural buckling and the lateral torsional buckling. Then a function is set up to automatically load in and calculate a single simply-supported beam.

3.1 Calculate buckling resistance

For a set load case and length, the flexural buckling of the columns and the lateral torsional buckling of the beams can be computed. Since the length of all columns is set to 4.5m, the only variable in those calculations is the type of constraints as it influences the buckling length. the constraint needs to be specified to be either 'simple' or 'fixed'.

[6]:
```
def buckling_res(CS, EI, constrain):
    EI = data.loc[CS, EI]
    Npl_rd = data.loc[CS, 'Npl_rd']
    # All columns have length = 4.5m,
    # simply supported on one end and partly constrained
    if constrain == 'simple':
        L = 4.5
    elif constrain == 'fixed':
        L = 4.5 * 2

    if data.loc[CS, 'buck_cz'] == 'a':
        a = 0.21
    elif data.loc[CS, 'buck_cz'] == 'b':
        a = 0.34
    elif data.loc[CS, 'buck_cz'] == 'c':
        a = 0.49

    Fe = np.pi**2 * EI / (L**2)
    slender = np.sqrt(Npl_rd / Fe)
    phi = 0.5 * (1 + a * (slender - 0.2) + slender**2)
    reduction = 1 / (phi + np.sqrt(phi**2 - slender**2))
    Nb_rd = reduction * Npl_rd
    return Nb_rd

def LTB(CS, L, constrain):
    # Lkip is assumed to be l as it accounts for no lateral restraints
    l = L
    # C values depend on moment line caused by the constraints
    if constrain == 'simple':
        C1 = 1.13
        C2 = 0.45
    elif constrain == 'fixed':
        C1 = 2.57
        C2 = 1.55

    EIw = 210 * data.loc[CS, 'Iw'] * 10**(-6)
```

```
    GIt = 81 * data.loc[CS, 'It'] * 10**(-3)

    S = np.sqrt(EIw / GIt)
    C = np.pi * C1 * (np.sqrt(1 + (np.pi**2 * S**2 * (C2**2 + 1)/
                                   (l**2))) + np.pi * C2 * S / l)

    if data.loc[CS, 'h'] / data.loc[CS, 'b'] <= 2:
        a = 0.34
    elif data.loc[CS, 'h'] / data.loc[CS, 'b'] > 2:
        a = 0.49

    M_cr = (C / l) * np.sqrt(data.loc[CS, 'EIz'] * GIt)
    slender_lt = np.sqrt(data.loc[CS, 'Mpl_rdy'] / M_cr)
    phi_lt = 0.5 * (1 + a * (slender_lt - 0.4) + 0.75 * slender_lt**2)
    reduction_lt = 1 / (phi_lt + np.sqrt(phi_lt**2 - 0.75 * slender_lt**2))
    reduction_lt = min(1, reduction_lt, 1 / (slender_lt**2))
    Mb_rdy = reduction_lt * data.loc[CS, 'Mpl_rdy']
    return Mb_rdy
```

3.2 Calculate Bending moment and deflection

Based on the cross-section chosen and the applied line load on the set length. The following equation sets up a simply supported beam and subsequently calculates the moments and deflections in the element. This is used to verify the method.

[7]:
```
def M_w_calculation_simple(CS, q, L):
    section = {}
    section['EI'] = data.loc[CS, 'EIy']
    section['EA'] = data.loc[CS, 'EA']

    Node.clear()
    Element.clear()

    node1 = Node(0, 0)
    node2 = Node(L, 0)

    elem = Element([node1, node2])
    elem.set_section(section)

    con = Constrainer()

    # Simply supported beam, u and w are constrained for both ends, but phi is␣
 ↪not
    con.fix_dof(node1, 0)
    con.fix_dof(node1, 1)
    con.fix_dof(node2, 0)
    con.fix_dof(node2, 1)
```

```
    # Add distributed load on the structure,
    # array form with [local x-direction, local z-direction]
    elem.add_distributed_load(q)

    global_k = elem.stiffness()
    global_f = np.zeros(6)

    global_f[0:3] = node1.p
    global_f[3:6] = node2.p

    Kc, Fc = con.constrain(global_k, global_f)
    u_free = np.matmul(np.linalg.inv(Kc), Fc)

    u_elem = con.full_disp(u_free)[elem.global_dofs()]

    w, moments = elem.bending_moments(u_elem, 101)
    w = w * 1000    # In mm
    return w, moments
```

3.3 Unity checks

For the unity checks, there has been made a distinction between the unity check for the beams and for the columns. Additionally, two separate equations are set up for a simply supported beam, which is calculated with the M_w_calculation_simple equation and one for moment fixed connections.

[8]:
```
def UC_beam_simple(CS, q, L):
    q_tot = q + data.loc[CS, 'm'] * 9.81 / 1000
    w, moments = M_w_calculation_simple(CS, [0, q_tot], L)
    w_max = max(w)
    M_ed = max(moments)

    Mpl_rd = data.loc[CS, 'Mpl_rdy']

    UC_m = M_ed / Mpl_rd
    UC_ltb = M_ed / LTB(CS, L, 'simple')
    UC_w = w_max / (L * 1000 / 250)
    return UC_m, UC_ltb, UC_w

def UC_beam(CS, elem_no, elements, constrainers):
    section = {}
    section['EI'] = data.loc[CS, 'EIy']
    section['EA'] = data.loc[CS, 'EA']

    elements[elem_no].set_section(section)
    Kc, Fc = con.constrain ( global_k, global_f )
    u_free = np.matmul ( np.linalg.inv(Kc), Fc )
```

```
    u_full = con.full_disp(u_free)
    u_elem = u_full[elements[elem_no].global_dofs()]
    m_ed = max(abs(elements[elem_no].bending_moments(u_elem, 100)[1]))
    w_max = max(abs(elements[elem_no].bending_moments(u_elem, 100)[0]))

    L = elements[elem_no].L

    UC_m = m_ed / data.loc[CS, 'Mpl_rdy']
    UC_ltb = m_ed / LTB(CS, L, constrainers[elem_no])
    UC_w = w_max / (L * 1000 / 250)

    return UC_m, UC_ltb, UC_w
```

[9]:
```
def UC_col(CS, elem_no, nodes, elements, constrainers, indices):
    section = {}
    section['EI'] = data.loc[CS, 'EIy']
    section['EA'] = data.loc[CS, 'EA']
    elements[elem_no].set_section(section)

    fz = global_f.reshape((9, 3))[:, 1].reshape((3, 3))
    N1 = fz[indices[elem_no][0], indices[elem_no][2]]
    N2 = fz[indices[elem_no][1], indices[elem_no][2]]
    N_ed = N1 + N2
    m_ed = m_max[elem_no]
    w_ed = w_max[elem_no]

    A = data.loc[CS, 'A']
    n = N_ed / data.loc[CS, 'Npl_rd']
    a = min((A - 2 * data.loc[CS, 'b'] * data.loc[CS, 'tf']) / A, 0.5)
    Nb_rdy = buckling_res(CS, 'EIy', constrainers[elem_no])
    Nb_rdz = buckling_res(CS, 'EIz', constrainers[elem_no])

    UC_n = n
    UC_by = N_ed / Nb_rdy
    UC_bz = N_ed / Nb_rdz
    UC_mn = m_ed / ((1 - n) / (1 - 0.5 * a) * data.loc[CS, 'Mpl_rdz'])
    UC_mn_int = n**2 + m_ed / data.loc[CS, 'Mpl_rdz']
    UC_mm = (m_ed / data.loc[CS, 'Mpl_rdy'])**2 + (m_ed / data.loc[CS,␣
↪'Mpl_rdz'])**5
    UC_w = w_ed / (4.5 * 1000 / 250)

    return UC_n, UC_by, UC_bz, UC_mn, UC_mn, UC_mm, UC_w
```

4 Verify implementations - Beam

These calculations are applied to a single simply supported beam and then checked against forget-me-nots and a FEM analysis in GSA.

[10]:
```
L = 6.3
q = [0, 40]
CS = 'IPE450'
```

[11]:
```
w, moments = M_w_calculation_simple(CS, q, L)
```

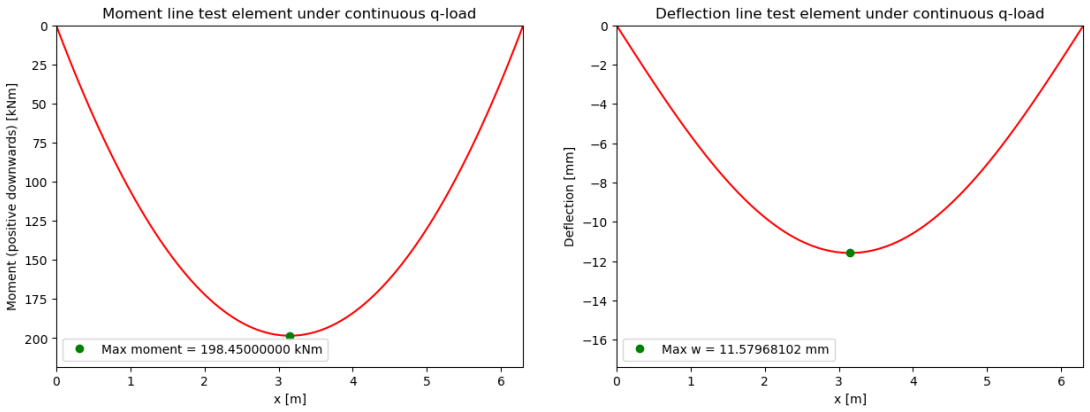
Plot the moment and deflection line of the single simply-supported beam element. The moment and deflection at mid-span can be compared to the following forget-me-nots:

$$M_{max} = \frac{1}{8} q L^2$$

$$w_{max} = \frac{5}{384} \frac{q L^4}{EI}$$

[12]:
```
print(f'Moment at midspan = {max(moments)} kNm')
print(f'Moment from forget-me-not = {q[1] * L**2 / 8} kNm')
print(f'Deflection at midspan = {max(w)} mm')
print(f"Deflection from forget-me-not = {5 * q[1] * L**4 / (384 * data.loc[CS,␣
 ↪'EIy']) * 1000} mm")

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.title('Moment line test element under continuous q-load')
plt.xlabel('x [m]')
plt.ylabel('Moment (positive downwards) [kNm]')
plt.plot(np.linspace(0,L,101),moments, 'r')
plt.ylim(0, moments.max() + moments.max()/10)
plt.xlim(0, L)
plt.plot(L/2, moments.max(), 'go', label=f'Max moment = {moments.max():.8f} kNm')
plt.gca().invert_yaxis()
plt.legend()

plt.subplot(1, 2, 2)
plt.title('Deflection line test element under continuous q-load')
plt.xlabel('x [m]')
plt.ylabel('Deflection [mm]')
plt.plot(np.linspace(0,L,101),-w, 'r')
plt.ylim(-w.max() - w.max()/2, 0)
plt.xlim(0, L)
plt.plot(L/2, -w[50], 'go', label=f'Max w = {w.max():.8f} mm')
plt.legend();
```

```
Moment at midspan = 198.45 kNm
Moment from forget-me-not = 198.45 kNm
Deflection at midspan = 11.57968101659751 mm
Deflection from forget-me-not = 11.57968101659751 mm
```



4.1 CS selection

For the single simply-supported beam in the verification example.

[13]:
```
def find_optimal_CS(data, q=40, L=6.3, threshold=0.9):
    for CS in data.index:
        UC = max(UC_beam_simple(CS, q, L))
        if UC < threshold:
            return CS
    return None  # If no cross-section meets the threshold

optimal_IPE = find_optimal_CS(IPE)
if optimal_IPE:
    print(f"The optimal cross-section is: {optimal_IPE} ")
    print(f"with UC = {UC_beam_simple(optimal_IPE, 40, 6.3)}")
else:
    print("No cross-section meets the threshold.")

optimal_HEA = find_optimal_CS(HEA)
if optimal_HEA:
    print(f"The optimal cross-section is: {optimal_HEA} ")
    print(f"with UC = {UC_beam_simple(optimal_HEA, 40, 6.3)}")
else:
    print("No cross-section meets the threshold.")
```

```
The optimal cross-section is: IPE450
with UC = (0.5056680869423886, 0.7614919581125341, 0.6237173809174011)
The optimal cross-section is: HEA340
with UC = (0.4680830648714528, 0.5028495197085249, 0.7649687424326245)
```

5 Verify implementation - Frame

Show cross-section selection and deflection calculation of a 2d frame structure. This is just to show how the cross-section selection is set up. The gravity loading which results in the correct link between beams and columns is added in Grasshopper. The moment- and deflection lines are then compared to the result from MatrixFrame.

[14]:
```
Node.clear()
Element.clear()

coordinates = [(0, 0), (4.5, 0), (9, 0), (0, 4.5), (4.5, 4.5),
               (9, 4.5), (0, 9), (4.5, 9), (9, 9)]

# Create nodes using a for-loop and assign them to variables
nodes = []
for i, (x, y) in enumerate(coordinates, start=1):
    globals()[f'node{i}'] = Node(x, y)
    nodes.append(globals()[f'node{i}'])

# List of node pairs for elements
col_pairs = [
    (node1, node4),
    (node2, node5),
    (node3, node6),
    (node4, node7),
    (node5, node8),
    (node6, node9)
]
beam_pairs = [
    (node4, node5),
    (node5, node6),
    (node7, node8),
    (node8, node9)
]

elements = []
# Create elements using a for-loop
for i, (node_a, node_b) in enumerate(col_pairs, start=1):
    globals()[f'col{i}'] = Element([node_a, node_b])
    elements.append(globals()[f'col{i}'])

for i, (node_a, node_b) in enumerate(beam_pairs, start=1):
    globals()[f'beam{i}'] = Element([node_a, node_b])
    elements.append(globals()[f'beam{i}'])

# Constrain bottom nodes in u and w direction
con = Constrainer()
```

```
con.fix_dof (nodes[0], 0)
con.fix_dof (nodes[0], 1)
con.fix_dof (nodes[1], 0)
con.fix_dof (nodes[1], 1)
con.fix_dof (nodes[2], 0)
con.fix_dof (nodes[2], 1)
```

Start with the same section everywhere to calculate moments in the structure and initial deflections. In this case, HEA300 is chosen. The loads are split up into self-weight and imposed loads, then combined for ULS calculations from the Eurocode. The area loads need to be multiplied by the floor length in the lateral direction to accommodate the correct floor area which is loaded by the beam. Assuming that the length of the beam in the y-direction is the same or more than the x-direction, the loaded area which is carried by the beam is a triangle calculated with $\frac{1}{4}L^2$ meters squared. To generate a line-load on the beam in x-direction, it results in a multiplication of the area load by $\frac{1}{4}L$ meters. Since this regards an industrial building, $\psi_{0;i}$ can be set to 1 for general loads, resulting in the following calculation for ULS:

$$1.35G_k + \sum 1.5Q_{i;k}$$

$$G_{selfweight} = A * 7850 * \frac{9.81}{1000}$$

$$G_{suspendedceiling} = 1.7[kN/m^2] * \frac{1}{4}L$$

$$G_{floorfinish} = 1.2[kN/m^2] * \frac{1}{4}L$$

$$G_{roof} = 1.5[kN/m^2] * \frac{1}{4}L$$

$$Q_{floor} = 15[kN/m^2] * \frac{1}{4}L$$

$$Q_{roof} = 1[kN/m^2] * \frac{1}{4}L$$

[15]:
```
def set_section_loads(section_list):
    loads = []
    sections = []
    L = 4.5  # Specific for this case

    for CS in section_list:
        section = {}
        section['EI'] = data.loc[CS, 'EIy']
        section['EA'] = data.loc[CS, 'EA']
        sections.append(section)

        selfweight = data.loc[CS, 'm'] * 9.81 / 1000
        q_beam1 = 1.35 * (selfweight + (1.7 + 1.2) * 0.25 * L) + 1.5 * 15 * 0.25␣
↪* L
        q_beam2 = 1.35 * (selfweight + (1.7 + 1.5) * 0.25 * L) + 1.5 * 1 * 0.25␣
↪* L
        q_col = 1.35 * selfweight
```

```
        loads.append([[-q_col*2, 0], [-q_col*2, 0], [-q_col*2, 0],
                      [-q_col, 0], [-q_col, 0], [-q_col, 0],
                      [0, q_beam1], [0, q_beam1],
                      [0, q_beam2], [0, q_beam2]])

    for i in range(len(elements)):
        elements[i].set_section(sections[i])
        elements[i].add_distributed_load(loads[i][i])

# Initial cross-section
section_list = ['HEA300', 'HEA300', 'HEA300', 'HEA300', 'HEA300',
                'HEA300', 'HEA300', 'HEA300', 'HEA300', 'HEA300']
set_section_loads(section_list)
```

[16]:
```
global_k = np.zeros ((3*len(nodes), 3*len(nodes)))
global_f = np.zeros (3*len(nodes))
for e in elements:
    elmat = e.stiffness()
    idofs = e.global_dofs()
    global_k[np.ix_(idofs,idofs)] += elmat

for n in nodes:
    global_f[n.dofs] += n.p

print(f'The following matrix shows the global loads applied on the {len(nodes)}␣
 ↪nodes:')
df = pd.DataFrame(global_f.round(1).reshape((9, 3)),
                  columns=['Fx', 'Fz', 'M'], index=np.arange(1, 10))
df.index.name = 'Node number'
display(df)
```

The following matrix shows the global loads applied on the 9 nodes:

Node number	Fx	Fz	M
1	0.0	5.3	0.0
2	0.0	5.3	0.0
3	0.0	5.3	0.0
4	0.0	77.4	-52.1
5	0.0	146.9	0.0
6	0.0	77.4	52.1
7	0.0	20.0	-13.0
8	0.0	37.4	0.0
9	0.0	20.0	13.0

[17]:
```
Kc, Fc = con.constrain ( global_k, global_f )
u_free = np.matmul ( np.linalg.inv(Kc), Fc )
```

```
u_full = con.full_disp(u_free)

dofs = np.array([[0, 0, u_free[0]],
                 [0, 0, u_free[1]],
                 [0, 0, u_free[2]],
                 [u_free[3],u_free[4],u_free[5]],
                 [u_free[6],u_free[7],u_free[8]],
                 [u_free[9],u_free[10],u_free[11]],
                 [u_free[12],u_free[13],u_free[14]],
                 [u_free[15],u_free[16],u_free[17]],
                 [u_free[18],u_free[19],u_free[20]]])

print('The following matrix shows the nodal displacements ')
print('in x- and z-direction and rotation phi:')
df2 = pd.DataFrame(dofs, columns=['d_x', 'd_z', 'd_phi'], index=np.arange(1, 10))
df2.index.name = 'Node number'
display(df2)
```

```
The following matrix shows the nodal displacements
in x- and z-direction and rotation phi:
```

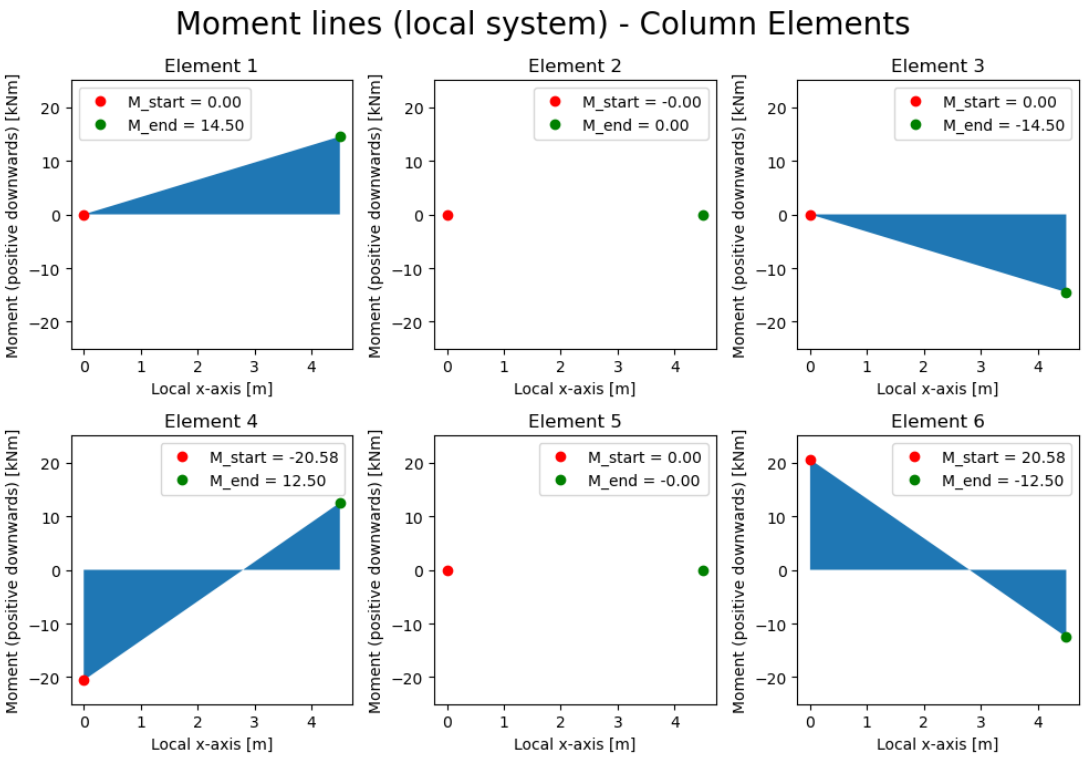
Node number	d_x	d_z	d_phi
1	0.000000e+00	0.000000	2.852898e-04
2	0.000000e+00	0.000000	-5.421011e-20
3	0.000000e+00	0.000000	-2.852898e-04
4	-7.863419e-06	0.000175	-5.653373e-04
5	2.168404e-19	0.000371	-4.743385e-20
6	7.863419e-06	0.000175	5.653373e-04
7	1.399806e-05	0.000214	-9.136758e-05
8	6.505213e-19	0.000442	2.710505e-20
9	-1.399806e-05	0.000214	9.136758e-05

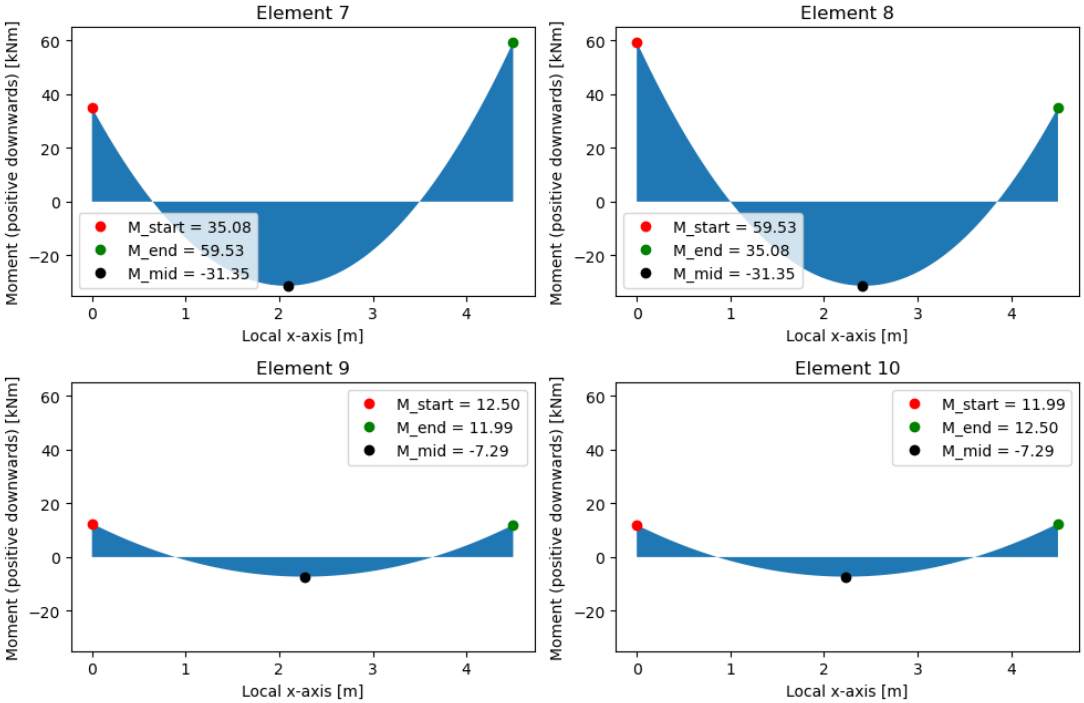
[18]:
```
print('Lastly, the support reactions on node 1, 2 and 3 are computed:')
df3 = pd.DataFrame(con.support_reactions(global_k,u_free,global_f).
 ↪reshape((3,2)),
                   columns=['Fx', 'Fz'],
index=['node 1', 'node 2', 'node 3'])
df3.index.name = 'Support reactions [kN]'
display(df3)
```

```
Lastly, the support reactions on node 1, 2 and 3 are computed:
```

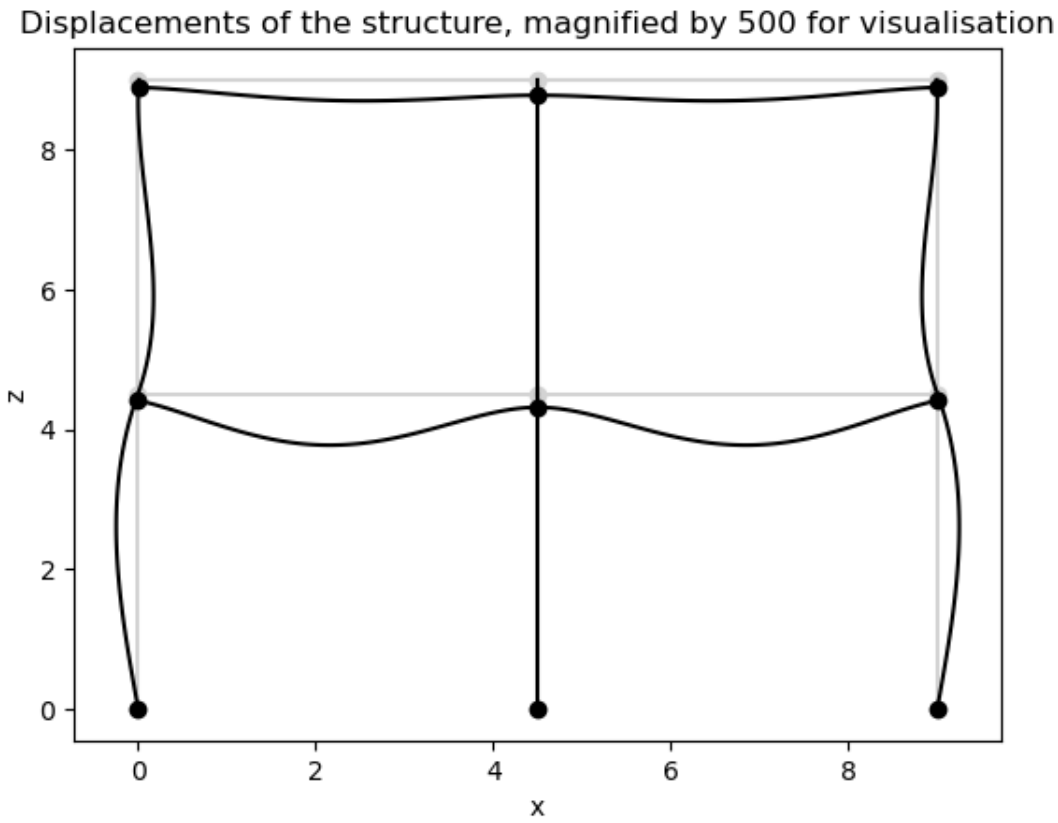
Support reactions [kN]	Fx	Fz
node 1	3.221545e+00	-97.323763
node 2	5.988142e-17	-200.141810
node 3	-3.221545e+00	-97.323763

[19]:
```
plt.figure(figsize=(10, 7), tight_layout=True)
plt.suptitle('Moment lines (local system) - Column Elements', fontsize=20)
m = []
w = []
for i, elem in enumerate(elements[:6]):
    plt.subplot(2, 3, i+1)
    u_elem = u_full[elem.global_dofs()]
    x = np.linspace(0, elem.L, 100)
    m.append(elem.bending_moments(u_elem, 100)[1])
    w.append(elem.bending_moments(u_elem, 100)[0])
    plt.title('Element ' + str(i+1))
    plt.xlabel('Local x-axis [m]')
    plt.ylabel('Moment (positive downwards) [kNm]')
    plt.ylim(-25, 25)
    plt.fill_between(x,-m[i], 0)
    plt.plot(x[0], -m[i][0], 'ro', label=f'M_start = {-m[i][0]:.2f}')
    plt.plot(x[-1], -m[i][-1], 'go', label=f'M_end = {-m[i][-1]:.2f}')
    plt.legend()
plt.show()

plt.figure(figsize=(10, 7), tight_layout=True)
plt.suptitle('Moment lines (local system) - Beam Elements', fontsize=20)
for i, elem in enumerate(elements[6:]):
    plt.subplot(2, 2, i+1)
    u_elem = u_full[elem.global_dofs()]
    x = np.linspace(0, elem.L, 100)
    m.append(elem.bending_moments(u_elem, 100)[1])
    w.append(elem.bending_moments(u_elem, 100)[0])
    plt.title('Element ' + str(i+7))  # Adjusting the element number
    plt.xlabel('Local x-axis [m]')
    plt.ylabel('Moment (positive downwards) [kNm]')
    plt.ylim(-35, 65)
    plt.fill_between(x,-m[i+6], 0)
    plt.plot(x[0], -m[i+6][0], 'ro', label=f'M_start = {-m[i+6][0]:.2f}')
    plt.plot(x[-1], -m[i+6][-1], 'go', label=f'M_end = {-m[i+6][-1]:.2f}')
    plt.plot(x[m[i+6].argmax(axis=0)], min(-m[i+6]), 'ko',
             label=f'M_mid = {min(-m[i+6]):.2f}')
    plt.legend()
plt.show()
```


Moment lines (local system) - Column Elements
Element 1
M_start = 0.00
M_end = 14.50
Element 2
M_start = -0.00
M_end = 0.00
Element 3
M_start = 0.00
M_end = -14.50
Element 4
M_start = -20.58
M_end = 12.50
Element 5
M_start = 0.00
M_end = -0.00
Element 6
M_start = 20.58
M_end = -12.50
Moment (positive downwards) [kNm]
Local x-axis [m]

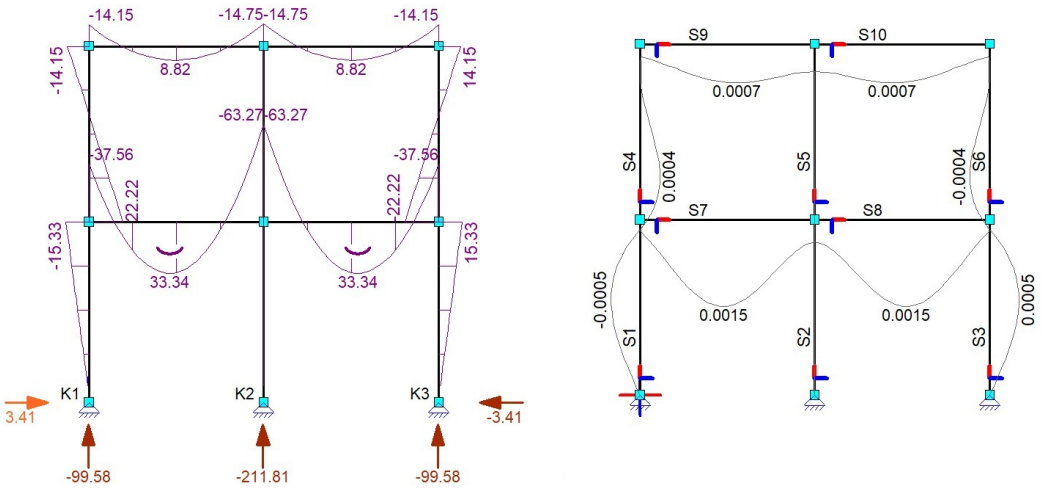

Moment lines (local system) - Beam Elements
Element 7
M_start = 35.08
M_end = 59.53
M_mid = -31.35
Element 8
M_start = 59.53
M_end = 35.08
M_mid = -31.35
Element 9
M_start = 12.50
M_end = 11.99
M_mid = -7.29
Element 10
M_start = 11.99
M_end = 12.50
M_mid = -7.29
Moment (positive downwards) [kNm]
Local x-axis [m]

[20]:
```
plt.title('Displacements of the structure, magnified by 500 for visualisation')
plt.xlabel('x')
plt.ylabel('z')

for i in coordinates:
    plt.plot(i[0], i[1], color='lightgray', marker='o')

a = np.linspace(0, 4.5, 100)
b = np.zeros(100)

plt.plot(b, a, color='lightgray')
plt.plot(b+4.5, a, color='lightgray')
plt.plot(b+9, a, color='lightgray')
plt.plot(b, a+4.5, color='lightgray')
plt.plot(b+4.5, a+4.5, color='lightgray')
plt.plot(b+9, a+4.5, color='lightgray')

plt.plot(a, b+4.5, color='lightgray')
plt.plot(a+4.5, b+4.5, color='lightgray')
plt.plot(a, b+9, color='lightgray')
plt.plot(a+4.5, b+9, color='lightgray')

# Displacement in z-direction need to be subtrackted as the axis in python is
# in a different direction as the z-axis of our structure
for i in range(len(coordinates)):
    plt.plot(coordinates[i][0] + dofs[i][0]*500,
             coordinates[i][1] - dofs[i][1]*500, 'ko')

plt.plot(w[0]*500, a, 'k')
plt.plot(w[1]*500 + 4.5, a, 'k')
plt.plot(w[2]*500 + 9, a, 'k')
plt.plot(w[3]*500, a + 4.5, 'k')
plt.plot(w[4]*500 + 4.5, a + 4.5, 'k')
plt.plot(w[5]*500 + 9, a + 4.5, 'k')

plt.plot(a, 4.5 - w[6]*500, 'k')
plt.plot(a + 4.5, 4.5 - w[7]*500, 'k')
plt.plot(a, 9 - w[8]*500, 'k')
plt.plot(a + 4.5, 9 - w[9]*500, 'k');
```



[21]:
```
m_max = []
w_max = []
for i in range(len(elements)):
    m_max.append(max(abs(m[i])))
    w_max.append(max(abs(w[i])))

print(f'The maximum deflection present in the frame is {max(w_max)*1000:.2f} mm')
print(f'The max moment present in the frame is {max(m_max):.2f} kNm')
```

```
The maximum deflection present in the frame is 1.45 mm
The max moment present in the frame is 59.53 kNm
```



5.1 Cross-section selection

[22]:
```
# Since gravity is not accounted for in this structure on its own,
# the forces of the upper beams and columns need to be added to the lower columns
# This is calculated by the following combination factors:
indices = [(1, 2, 0), (1, 2, 1), (1, 2, 2), (2, 0, 0), (2, 0, 1), (2, 0, 2)]
constrainers = ['half', 'half', 'half', 'fixed', 'fixed', 'fixed',
                'fixed', 'fixed', 'fixed', 'fixed']

def select_best_cs_beam(elem_no, data, threshold=0.85):
    for CS in data.index:
        UC = UC_beam(CS, elem_no, elements, constrainers)
        if max(UC) < threshold:
            return CS
    return None

def select_best_cs_col(elem_no, data, threshold=0.85):
    for CS in data.index:
        UC = UC_col(CS, elem_no, nodes, elements, constrainers, indices)
        if max(UC) < threshold:
            return CS
    return None

CS_beams = []
CS_cols = []
for i in [6, 7, 8, 9]:
    CS_beams.append(select_best_cs_beam(i, IPE))
for i in range(6):
    CS_cols.append(select_best_cs_col(i, HEB))

print(f'The optimal cross-sections of the beams = {CS_beams}')
```

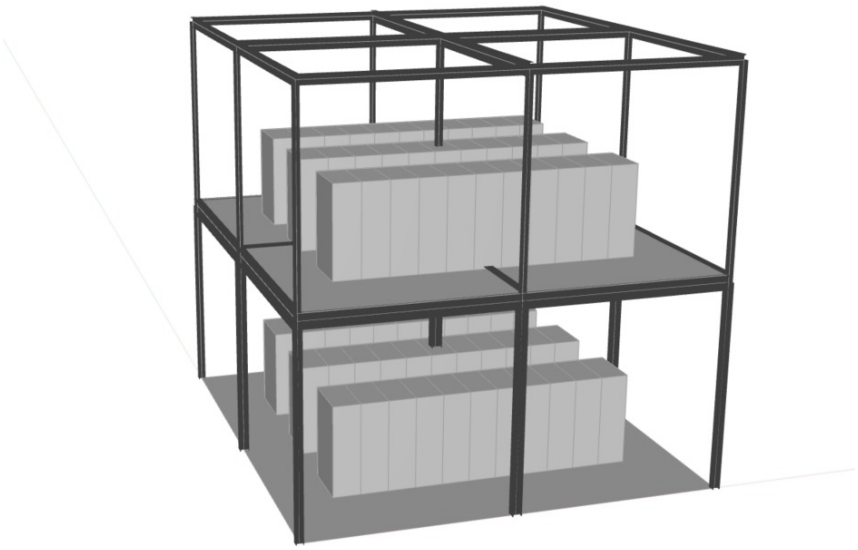
```
print(f'The optimal cross-sections of the lower columns = {CS_cols[:3]}')
print(f'The optimal cross-sections of the upper columns = {CS_cols[3:6]}')
```

```
The optimal cross-sections of the beams = ['IPE220', 'IPE220', 'IPE140',
'IPE140']
The optimal cross-sections of the lower columns = ['HEB120', 'HEB100', 'HEB120']
The optimal cross-sections of the upper columns = ['HEB140', 'HEB100', 'HEB140']
```

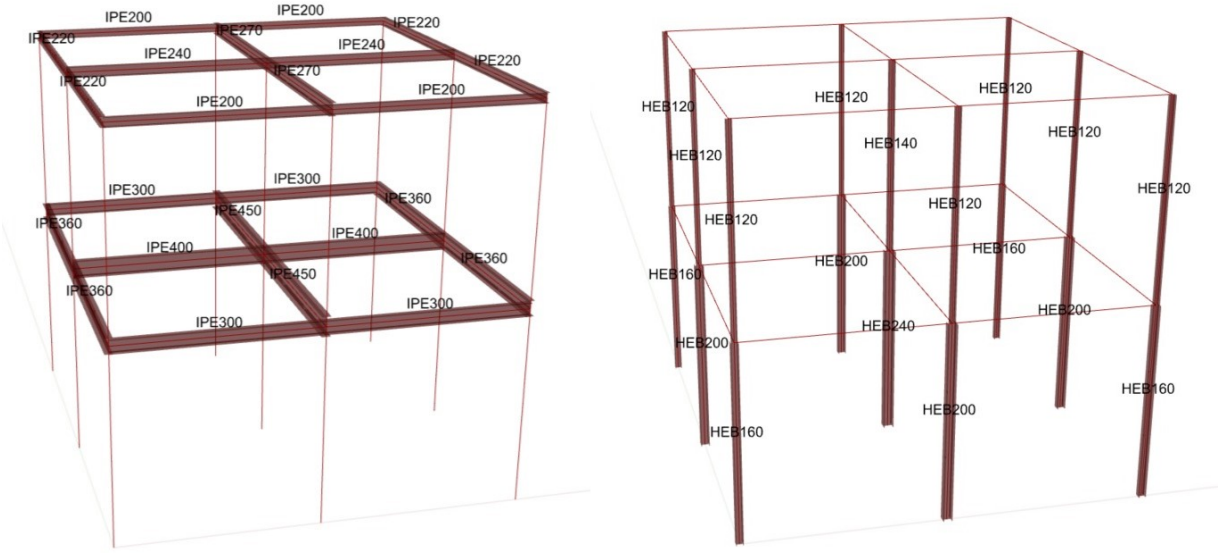
6 Grasshopper implementation and GSA validation

The code above is implemented in a grasshopper file to generate a structural design of a data centre. To check this application, a simple structure is calculated in Grasshopper and checked in Oasys GSA software. The following input parameters are chosen to create a clear and relatively small design:

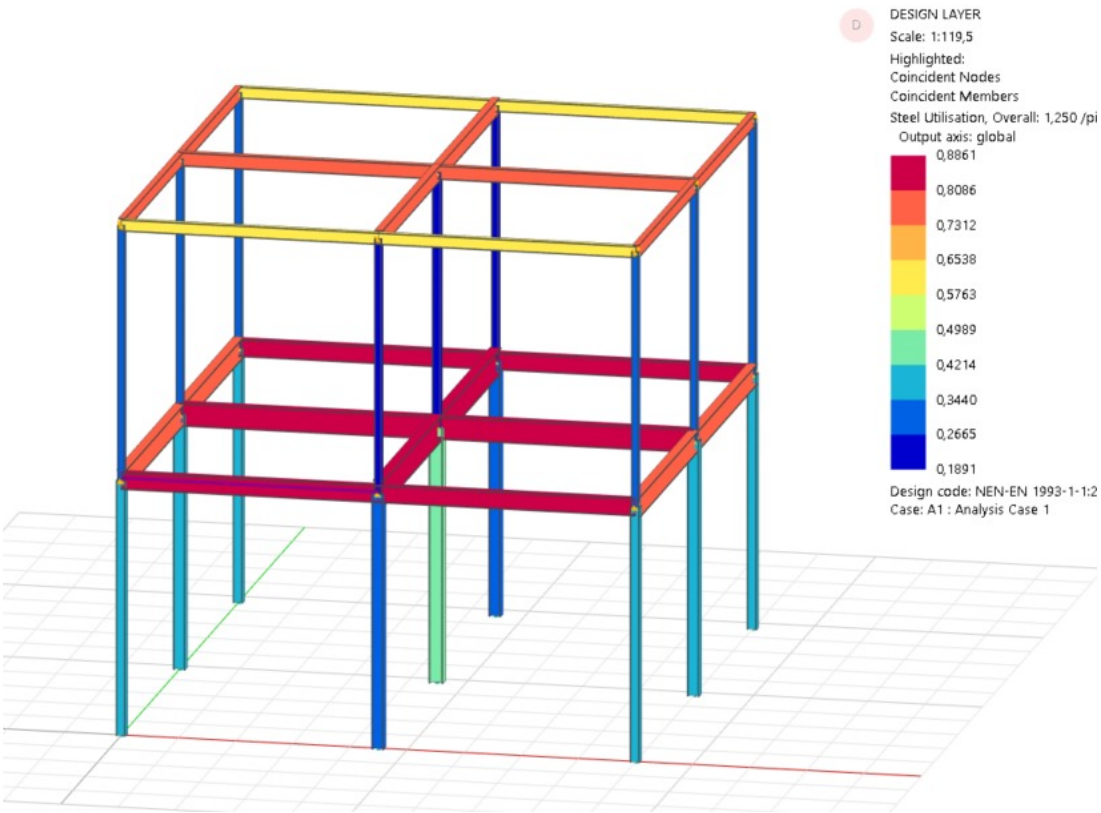
- 3 rows of 11 dataracks
- One column in floorplan at row 2, datarack 6
- This results in the following grid spacing in meters:
  - x-direction: [4.5, 4.5]
  - y-direction: [4.8, 4.8]
- The beams are chosen to be IPE sections
- The columns are chosen to be HEB sections
- Simply supported at ground level with moment-tight connections in other nodes



As can be seen in the render, the dataracks are located on the ground floor and the first floor. The total number of dataracks is then $2 * 3 * 11$ minus 2 where the column in the floorplan is located. Next, the Matrix-method is applied to calculate each load and unity check to find the optimal cross-sections. The following cross-sections are chosen in the given configuration:



These cross-sections and loads are then put into a FEM software for a more thorough design check. The following steel utilization were found:



D

Python Script of RNN

Python script for RNN model

T.C.S. van Oers

December 3, 2024

[ ]:
```
# !pip install --upgrade tensorflow
# !pip install -U tensorflow keras
```

[ ]:
```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences  # Use from tf.
 ↪keras for utils
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import regularizers
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, classification_report, confusion_matrix
```

[ ]:
```
# Load data and concatenate into a single DataFrame for consistency
# Each dataset represents different floorplan configurations
data1 = pd.read_excel('Config_2x11_3x13_4x15_4x21.xlsx')
data2 = pd.read_excel('Config_3x20_5x16.xlsx')
data3 = pd.read_excel('Config_2x29_7x11.xlsx')
data4 = pd.read_excel('Config_5x23.xlsx')
data5 = pd.read_excel('Config_6x26.xlsx')
data6 = pd.read_excel('Config_7x29.xlsx')
data = pd.concat([data1, data2, data3, data4, data5, data6])
```

[ ]:
```
# Load available profile information and element labels
CS_info = pd.read_csv('CS_info.csv')
all_profiles = np.concatenate([['0'], CS_info['Unnamed: 0'].values, ['N.A']])  #␣
 ↪Add extra profiles if needed
all_labels = ['0', 'B1x', 'B1y', 'B2x', 'B2y', 'C1r', 'C1m', 'C2r', 'C2m']
```

[ ]:
```
# Label encoding for profile and element type columns
profile_encoder = LabelEncoder()
profile_encoder.fit(all_profiles)

element_type_encoder = LabelEncoder()
```

```
element_type_encoder.fit(all_labels)

# Encode the 'element_type' and 'profile_label' columns using the fitted encoders
data['element_type_encoded'] = element_type_encoder.
 ↪transform(data['Element_type'])
data['profile_label_encoded'] = profile_encoder.transform(data['Profile_HEAb'])

# Split data into individual configurations based on normalized ID column
configs = []
current_config = []

for i, row in data.iterrows():
    # Start a new configuration when ID is close to 0
    if row['ID'] == 0 and current_config:
        configs.append(current_config)
        current_config = []
    # Append row data to the current configuration
    current_config.append([row['ID'], row['element_type_encoded'],␣
 ↪row['Length'], row['profile_label_encoded']])
    # End configuration when ID is close to 1
    if row['ID'] == 1:
        configs.append(current_config)
        current_config = []

# Add last config if not empty
if current_config:
    configs.append(current_config)

# Pad each configuration to the same length
max_length = max(len(config) for config in configs)
padded_configs = pad_sequences(configs, maxlen=max_length, dtype='float32',␣
 ↪padding='post')

# Convert to a 3D tensor
tensor_3d = np.array(padded_configs)
X_0 = tensor_3d[:, :, :3]
y_0 = tensor_3d[:, :, -1:]

print("3D Tensor shape:", tensor_3d.shape)   # (num_designs, max_length,␣
 ↪num_features)
```

[ ]:
```
# Split into input and target tensors
X = tensor_3d[:, :, :3]
y = tensor_3d[:, :, -1:]

# One-Hot Encode Element Type and Profile Labels
num_element_type_classes = len(element_type_encoder.classes_)
```

```
X_element_type = tf.keras.utils.to_categorical(X[:, :, 1],␣
↪num_classes=num_element_type_classes)

# Concatenate ID, Length, and one-hot element type encoding
X = np.concatenate([X[:, :, [0, 2]], X_element_type], axis=-1)

# One-hot encode target profiles
num_profiles = len(profile_encoder.classes_)
y_one_hot = np.array([tf.keras.utils.to_categorical(seq,␣
↪num_classes=num_profiles) for seq in y])

# Train-Test-Validation Split
X_train, X_temp, y_train, y_temp = train_test_split(X, y_one_hot, test_size=0.
↪20, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.25,␣
↪random_state=42)

# Final shapes check
print(f"X_train shape: {X_train.shape}")
print(f"X_val shape: {X_val.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_val shape: {y_val.shape}")
print(f"y_test shape: {y_test.shape}")
```

[ ]:
```
print(X_train.shape)  # Should be (batch_size, max_seq_length, num_features)
print(X_val.shape)
print(X_test.shape)
```

[ ]:
```
# Define model architecture
input_layer = tf.keras.Input(shape=(max_length, X_train.shape[-1]))

model = tf.keras.Sequential([
    input_layer,
    tf.keras.layers.Masking(mask_value=-1),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(256,␣
↪return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(256,␣
↪return_sequences=True)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation='relu',␣
↪kernel_regularizer=regularizers.l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(64, activation='relu', kernel_regularizer=regularizers.
↪l2(0.01)),
    tf.keras.layers.Dense(num_profiles, activation='softmax')
```

```
])

# Define the optimizer with the learning rate schedule
optimizer = tf.keras.optimizers.Adam()

# Compile the model
model.compile(optimizer=optimizer, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
# Define callbacks
lr_schedule = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6
)

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', patience=10, restore_best_weights=True
)

# Train model and validate with the validation set
history = model.fit(
    np.array(X_train), np.array(y_train),
    validation_data=(np.array(X_val), np.array(y_val)),
    epochs=150,
    batch_size=16,
    callbacks=[lr_schedule, early_stopping],
    verbose=1
)
```

[ ]:
```
# Plot the training and validation loss over epochs
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss of IPEm model')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig('RNN_HEAb.png', bbox_inches='tight')
plt.show()
```

[ ]:
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(np.array(X_test), np.array(y_test),␣
 ↪verbose=1)

print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

[ ]:
```
model.save('trained_model_HEAb.keras')
```

[ ]:
```
def evaluate_model(X_test, y_test, model, model_name):
    """Evaluates the model on test data, calculates metrics, and returns␣
 ↪flattened true and predicted profiles."""

    # Predict class probabilities and get predicted class indices
    y_pred = model.predict(X_test)
    y_pred_classes = np.argmax(y_pred, axis=-1).flatten()

    # Flatten the true labels, excluding padding (-1)
    y_true_classes = np.argmax(y_test, axis=-1).flatten()

    # Calculate metrics without padding values
    y_true_non_pad = y_true_classes[y_true_classes != -1]
    y_pred_non_pad = y_pred_classes[y_true_classes != -1]

    # Calculate and print test loss and accuracy
    test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=1)
    print(f"\n{model_name} - Test Loss: {test_loss:.4f}, Test Accuracy:␣
 ↪{test_accuracy:.4f}")

    return y_true_non_pad, y_pred_non_pad

# Call the function with pre-padded test data
y_true_IPEm, y_pred_IPEm = evaluate_model(X_test, y_test, model, "IPEm")

# Dictionary to store the true and predicted labels for each model
models = {
    "IPEm": (y_true_IPEm, y_pred_IPEm)
    # "HEAm": (y_true_HEAm, y_pred_HEAm),
    # "IPEb": (y_true_IPEb, y_pred_IPEb),
    # "HEAb": (y_true_HEAb, y_pred_HEAb)
}

# Loop through each model, calculate metrics, and print results
for model_name, (y_true, y_pred) in models.items():
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',␣
 ↪zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    # Print metrics for each model
    print(f"\nModel: {model_name}")
    print(f"  Accuracy: {accuracy:.4f}")
    print(f"  Precision: {precision:.4f}")
    print(f"  Recall: {recall:.4f}")
    print(f"  F1 Score: {f1:.4f}")
```

[ ]:
```
# Filter all profile types from the dataset and store them in separate lists to␣
 ↪create a standard order
order_IPE = [profile for profile in all_profiles if profile.startswith("IPE")]
order_HEB = [profile for profile in all_profiles if profile.startswith("HEB")]
order_IPHE = ["0"] + order_HEB + order_IPE + ["N.A"]
order_HEA = ["0"] + [profile for profile in all_profiles if profile.
 ↪startswith("HEA")]

order_IPHE_dict = {value: index for index, value in enumerate(order_IPHE)}
order_HEA_dict = {value: index for index, value in enumerate(order_HEA)}

from matplotlib.colors import LogNorm
import seaborn as sns

# Decode the true and predicted labels for the profiles using the profile encoder
y_true = profile_encoder.inverse_transform(y_true_IPEm)
y_pred = profile_encoder.inverse_transform(y_pred_IPEm)

y_label = np.unique(y_true)
x_label = np.unique(y_pred)

# Sort labels based on the predefined order in order_HEA_dict
y_label_ordered = np.array(sorted(y_label, key=lambda x: order_HEA_dict.get(x,␣
 ↪float('inf'))))
x_label_ordered = np.array(sorted(x_label, key=lambda x: order_HEA_dict.get(x,␣
 ↪float('inf'))))

# Calculate performance metrics for the predictions
precision = precision_score(y_true, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

# Generate a confusion matrix to evaluate the model's prediction accuracy
conf_matrix = confusion_matrix(y_true, y_pred)
print(f"Precision: {precision:.4f}, Recall: {recall:.4f}, F1-Score: {f1:.4f}")

# Plot the confusion matrix, focusing on non-zero rows/columns, with a␣
 ↪logarithmic color scale
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix[1:, 1:], fmt='d', cmap='Blues',
            xticklabels=y_label_ordered[1:], yticklabels=y_label_ordered[1:],␣
 ↪norm=LogNorm())
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix: HEAb model predictions of test set")
plt.savefig('ConfusionM_HEAb.png', bbox_inches='tight')
plt.show()
```

E

Grasshopper Script

Grasshopper_codes

October 30, 2024

1 Python scripts in Grasshopper

Throughout the Grasshopper scripts, multiple python components were applied to generate specifc outputs. The following appendix shows the main components which explain the data conversion.

1.1 Generate Grid-Spaces from input

The input of the data centre design is the number of rows, number of racks per row and a genepool indicating the location of the columns, thus determining the grid-spaces. The following code was used to transform the input data into the floorplan design based on specific design requirements for a data centre hall. The Brep of the data racks are set up, which determine the locations of the columns and the total floorplan size. The python component creates the total length and the grid-spaces from zero to one as an output. - Input: rows, racks, rows_loc, cols_loc - Output: dataracks, x_length, y_length, x_space, y_space, grid

[ ]:
```
import rhinoscriptsyntax as rs

# Define the size of a datarack
rack_width = 0.6
rack_depth = 1.2

# Define the space between rows
row_space = 1.2

# Define the extra space around the perimeter
perimeter_spacex = 1.2
perimeter_spacey = 1.8

print(rows_loc)

def create_datacentre_layout(rows, racks_per_row):

    # Initialize the list of rack rectangles
    rack_rectangles = []

    # Iterate over the rows
    for i in range(rows):
        # Iterate over the racks in each row
```

```
        for j in range(racks_per_row):
            # Calculate the position of the rack
            x = j * rack_width + perimeter_spacex
            y = i * (rack_depth + row_space) + perimeter_spacey

            # Create a rectangle for the rack
            rack_rectangle = rs.AddRectangle(rs.WorldXYPlane(), rack_width,␣
 ↪rack_depth)

            # Move the rectangle to the position of the rack
            rs.MoveObject(rack_rectangle, [x, y, 0])

            # Add the rectangle to the list of rack rectangles
            rack_rectangles.append(rack_rectangle)

    # Calculate the width and height of the datacentre
    datacentre_width = racks_per_row * rack_width + 2 * perimeter_spacex
    datacentre_height = rows * (rack_depth + row_space) + perimeter_spacey + 0.6

    # Return the rack rectangles and the perimeter rectangle
    return rack_rectangles, datacentre_width, datacentre_height

# Call the function to create a datacentre layout with x rows and y racks per row
dataracks, x_length, y_length = create_datacentre_layout(rows, racks)

def create_centers(rows, racks_per_row, row_number, rack_number):
    # Calculate the position of specified rack
    x = perimeter_spacex + rack_number * rack_width - rack_width / 2
    y = perimeter_spacey + row_number * (rack_depth + row_space) - row_space -␣
 ↪rack_depth / 2

    # Create center and normalise with total length
    center = [x / x_length, y / y_length]

    # Check if the row_number and rack_number are within the valid range
    if row_number > rows and rack_number + 3 > racks_per_row:
       print("Invalid row number or rack number. No lines will be created.")
        return None, None

    if row_number > rows:
       print("Invalid row number. No horizontal lines will be created.")
        return center[0], None

    if rack_number + 3 > racks_per_row:
       print("Invalid rack number. No vertical lines will be created.")
        return None, center[1]
```

```
    else:
        return center[0], center[1]

# Only use values of genepool which are within the datacentre perimeter
def filter_genepool(slider_value, genepool):
    result = []
    current_sum = 0
    # Loop over values within gene pool
    for value in genepool:
        if current_sum + value > slider_value:
            break
        result.append(value)
        current_sum += value
    return result

a = rows_loc
b = cols_loc

filtered_a = filter_genepool(rows, a)
filtered_b = filter_genepool(racks, b)

row_number_start = 0
rack_number_start = 0
x_list = [0]
y_list = [0]
max_length = max(len(filtered_a), len(filtered_b))

# Iterate over the range of the maximum length
for i in range(max_length):
    # Use the value from list 'a' if it exists, otherwise use a default value
    a_val = a[i] if i < len(a) else 0
    # Use the value from list 'b' if it exists, otherwise use a default value
    b_val = b[i] if i < len(b) else 0

    row_number_start += a_val
    rack_number_start += b_val
    center = create_centers(rows, racks, row_number_start, rack_number_start)
    x_list.append(center[0])
    y_list.append(center[1])

x_space0 = [list for list in x_list if list is not None]
y_space0 = [list for list in y_list if list is not None]

x_space0.append(1)
y_space0.append(1)

x_space = []
```

```
for item in x_space0:
    if item not in x_space:
        x_space.append(item)

y_space = []
for item in y_space0:
    if item not in y_space:
        y_space.append(item)

grid = [len(x_space) - 1, len(y_space) - 1]
```

2 Data gathering

In order to save the data and use it for the RNN model, the data needs to be adapted based on the generated design. The following python component transforms the design aspects into the resulting data by normalising the element IDs, generating their element type and output the generated profiles for the four different design choice combinations. Note that this code transforms all design configurations where a profile generated a warning, all profiles in that configuration are set to N.A. The warnings occur when there is no profile section possible that withstands the applied load. - Input: grid, Vectors, Lines, beams, warnings - Output: IDs_norm, Gridx, Gridy, Lengths, Element_type, Profiles

[ ]:
```
import rhinoscriptsyntax as rs

Element_type = []
IDs_norm = []
Lengths = Lines
Gridx = []
Gridy = []
Profiles = []

for i in range(len(Lines)):
    if Vectors[i][0] != 0:
        Element_type.append(beams[i].split("'")[1] + "x")
    elif Vectors[i][1] != 0:
        Element_type.append(beams[i].split("'")[1] + "y")
    else:
        Element_type.append("C")
    IDs_norm.append(i / (len(Lines) - 1))
    Gridx.append(grid[0])
    Gridy.append(grid[1])

status = "GOOD"
for message in warnings:
    if "bigger" in message.lower():
        status = "FAULT"
        break
```

```
for i in range(len(beams)):
    test_E = beams[i]
    if status == "GOOD":
        Profiles.append(test_E.split(':')[3][:6])
        if Profiles[i] == "IPE80;":
            Profiles[i] = "IPE80"
    elif status == "FAULT":
        Profiles.append("N.A")
```

3 Volume to weight

The following Python component calculates the weight of steel per match type, so for new steel, direct reuse, indirect reuse and accompanying cut-off waste. the division is made by linking the matchtype to the total volumes of each element, which is multiplied with the specific weight of the S235 steel. - Input: Matchtype, Volume, Reused_V, Cutoff_V, Weight - Output: Total_W, Virgin_W, D_Reuse_W, I_Reuse_W, I_Waste_W

[ ]:
```
import rhinoscriptsyntax as rs

Virgin_W = []
D_Reuse_W = []
I_Reuse_W = []
I_Waste_W = []

for i in range(len(Matchtype)):
    Virgin_W.append(0)
    D_Reuse_W.append(0)
    I_Reuse_W.append(0)
    I_Waste_W.append(0)

virgin_dict = {i: volume for i, volume in enumerate(Volume) if Matchtype[i] == 0}
dr_dict = {i: volume for i, volume in enumerate(Volume) if Matchtype[i] in [1,␣
↪2]}
ir_dict = {i: volume for i, volume in enumerate(Reused_V) if Matchtype[i] == 3}
iw_dict = {i: volume for i, volume in enumerate(Cutoff_V) if Matchtype[i] == 3}

for i in range(len(Matchtype)):
    if Virgin_W[i] == 0:
        Virgin_W[i] = virgin_dict.get(i, 0) * Weight
    if D_Reuse_W[i] == 0:
        D_Reuse_W[i] = dr_dict.get(i, 0) * Weight
    if I_Reuse_W[i] == 0:
        I_Reuse_W[i] = ir_dict.get(i, 0) * Weight
    if I_Waste_W[i] == 0:
        I_Waste_W[i] = iw_dict.get(i, 0) * Weight
```

```
Total_W = sum(Volume) * Weight
```

4 Calculate design variables

- Input:CO2_new, CO2_opt, Mass, Area, Dataracks
- Output: Ratio_reuse, CO2_reduction, Carbon_score, Carbon_per_rack

[ ]:
```
import rhinoscriptsyntax as rs

Floorplan_cols = (Dataracks[2] - 2) * (Dataracks[3] - 2)
Dataracks_tot = Dataracks[0] * Dataracks[1] - Floorplan_cols
print(Floorplan_cols)
print('The total amount of dataracks in the design = ', Dataracks_tot)

list = [Mass[0], Mass[1], Mass[2] + Mass[3]]
total = sum(Mass)
ratios = []
for i in range(len(list)):
    ratios.append(round(list[i] / total, 3) * 100)
Ratio_reuse = ratios[1] + ratios[2]
Ratio_newsteel = ratios[0]
print('The percentage of reuse in the new design = ', Ratio_reuse)

CO2_reduction = round((CO2_new - CO2_opt) / CO2_new * 100, 1)
print('The percentage of CO2 reduction = ', CO2_reduction)

Carbon_score = round(CO2_opt / Area, 2)
print('The carbon score of the optimized design = ', Carbon_score, 2)

Carbon_per_rack = round(CO2_opt / Dataracks_tot, 2)
print('The embodied carbon per datarack = ', Carbon_per_rack, 2)
```

4.1 Loading in trained RNN model

To load in the trained RNN models, the Grasshopper script is rewritten in Rhino V8, where normal Python 3 packages can be applied within the Python Grasshopper components by using the syntax "# requirements: numpy, tensorflow" - Input: input_data - Output: predicted_profiles

[ ]:
```
# requirements: numpy, tensorflow
import tensorflow as tf
import numpy as np

model_path = "C:\Users\Tessel.vanOers\OneDrive - Arup\Thesis\Notebooks/IPEm.
 ↪keras"

model = tf.keras.models.load_model(model_path)
predicted_profiles = model.predict(input_data)
```