Performance-Driven Design Exploration of Biocomposite Facades

Advancing Facade Design through Enhanced
Computational Efficiency, Accuracy, and Interpretability:
A Novel AI-Driven Facade Design Framework comprising
Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN)

Building Technology Graduation Studio
Djani Cerneus
Report
2 January 2025

Delft University of Technology
MSc Architecture, Urbanism & Building Sciences
Building Technology Track

Student:
Djani Cerneus | 5609305

Mentors:
Dr. M. Turrin | Design Informatics
Dr. A. Luna Navarro | Facade & Product Design

Delegate of the Board of Examiners:
Dr. R. Binnekamp

May 2024 - January 2025

Acknowledgement

I would like to express my deepest gratitude to my mentors, Michela Turrin and Alessandra Luna Navarro, for their invaluable guidance, support, and expertise, which have been foundational in shaping this thesis. Their knowledge and experience have elevated the quality of this thesis to a level I could not have achieved on my own. It has truly been an honour to learn from them, both through this thesis and during the regular courses of my masters. I would like to extend my sincere thanks to Ruud Binnekamp, delegate of the board of examiners, for his valuable insights during my P2 presentation. Finally, I could not have undertaken this journey without the unconditional support of my family, partner, and best friend. Their feedback, inspiration, encouragement, and moral support have been a constant source of strength throughout my academic career.

Abstract

With decision-making becoming increasingly data-driven in early design stages due to global environmental challenges, and qualitative metrics remaining crucial in facade design due to their huge architectural impact, performance-driven design exploration frameworks are emerging as powerful tools to explore vast design spaces based on geometry typology and approximated performance. However, those used in facade design based solely on Self-Organising Maps (SOM), face significant challenges in computational efficiency. While more advanced frameworks used in the AEC sector combining SOM and Multi-Layer Perceptrons (MLP) address this, they still face limitations in prediction accuracy, convergence speed, interpretability, reliability, and usability, reducing their effectiveness in decision-making. This thesis aimed to overcome these limitations, by developing a novel framework integrating SOM and Kolmogorov-Arnold Networks (KAN), applied in the design process of an aluminium-based biocomposite curtain wall facade. The results demonstrate that substituting heavy performance simulations with fast approximations using KAN leads to significant enhancements in computational efficiency. In addition, comparative analysis revealed that KAN outperforms MLP in prediction accuracy on highly-complex performance metrics, with much faster convergence and smaller architectures. Furthermore, KAN proved to be faster and more intuitive to train, as well as more consistent in predictions. Moreover, KAN enhanced interpretability between geometry and performance, enabling designers to focus on relevant design variables and adjust them strategically toward optimal performance, providing an integrated solution with transparent decision-making and faster processing compared to traditional sensitivity analysis tools. Finally, a novel approach to design exploration has enabled the integration of less-geometry related design variables, enhancing optimisation capabilities, as well as proven to balance human-AI interaction more efficiently than traditional frameworks, making design exploration more interactive, thereby more effective and intuitive. Ultimately, the SOM-KAN framework has proven to advance the facade design process by facilitating more efficient decision-making in early design stages, leading to superior architectural and sustainable solutions.

Keywords: Performance-Driven Design Exploration, Facade Design, Biocomposites, Self-Organising Maps (SOM), Kolmogorov-Arnold Networks (KAN), Multi-Layer Perceptrons (MLP).

Contents

1 Introduction

1.1 Introduction

Due to environmental challenges such as climate change, resource depletion, pollution, and biodiversity loss, many industries are strongly encouraged to be more sustainable (Leising et al., 2018). To address these global issues collectively, 196 countries, including the Netherlands, have signed an international climate accord in 2016, the Paris Agreement, to achieve a 100% circular economy by 2050 (Government of the Netherlands, 2023). In addition, the Netherlands' subgoal is to halve its raw material consumption by 2030. The thrust towards a circular economy is a necessary transition, aiming to reduce eco-impact, waste, and pollution, use environmentally friendly materials and extend the lifespan of resources by closing material loops (Ellen MacArthur Foundation, n.d.), as illustrated in Figure 1.1.1.

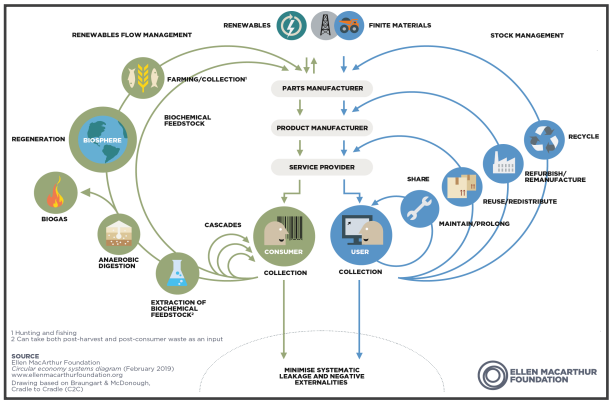


Figure 1.1.1. Circular economy diagram. From "*Ellen MacArthur Foundation*", by Ellen MacArthur Foundation, 2019 (https://www.ellenmacarthurfoundation.org/circular-economy-diagram).

In this circular framework, the construction sector is key, accounting for approximately 50% of raw material consumption, 42% of energy consumption, and 35% of greenhouse gas emissions in the EU (Gervasio & Dimova, 2018).

Especially housing corporations, which manage around 30% of the residential stock in the Netherlands - or about 2.4 million homes - play a huge role in the sustainability transition, despite their financial limitations. To reach the national circular objectives, they need to upgrade 100,000 dwellings annually towards an energy and CO2-neutral portfolio by 2050, with an average energy label B or higher (Dantuma & Van Sante, 2018). Facade renovations, which traditionally focus on insulation and infiltration enhancements, are key in this effort as they significantly affect the portfolio's sustainability. Not only because of their impact on energy performance but also due to their exposure to severe climatic conditions which negatively affect the lifespan of its materials (Overend et al., 2021). Using sustainable alternatives could positively impact the environment, by minimising waste and pollution at production and at the end of their lifecycle.

In the EU, construction and demolition waste is the largest sector at 37% of total waste with 839 million tonnes of waste in 2018. Even though recycling rates are high with an average of 74%, over 70% of it is downcycled (European Circular Economy Stakeholder Platform, 2021). Waste wood accounts for approximately 60 million tonnes annually, with around 49% going to energy-to-waste (ETW) plants and around 51% to products like chipboard (Business Waste, 2024), thus massively underutilised, especially as wood is considered to be a valuable bio-based and renewable resource.

In an interview with Hester ten Zijthoff, project manager of Ymere, one of the biggest housing corporations in the Netherlands, challenges in utilising waste wood to its fullest potential were highlighted. Particularly issues of chemical contamination as a result of paints and impregnating agents, metal presence, and varying shapes make high-quality reuse labor-intensive and complex. These issues significantly hinder mass production, leading to prohibitive costs. Additionally, the lack of high-quality reuse examples and an insufficient budget for innovation obstruct its implementation in housing corporations' renovation strategies, despite its potential as a valuable bio-based waste stream resource (Dantuma & Van Sante, 2018).

Wood dust, a byproduct of wood processing, although, offers huge potential for high-value facade applications, as discovered during an internship with "Circular Wood 4 The Neighbourhood," which focuses on reusing waste wood utilising robotic fabrication techniques for furniture projects. Traditionally, wood dust of housing corporations ends up in ETW plants, yet its small uni-

form nature offers vast possibilities for mass production and offers huge amount of design freedom. When wood dust fibers are mixed with a natural matrix, they can transform into a biocomposite with excellent insulating properties (Bahar et al., 2023; Abdallah et al., 2022; Lertwattanaruk & Suntijitto, 2015), which can be of significant value for housing corporations' facade renovations.

Biocomposites are gaining momentum as sustainable alternatives to synthetic composites, while insufficient processing methods and high fabrication costs previously limited their development (Zwawi, 2021). Biocomposites, also known as Natural Fiber Composites (NFC), consist of biopolymer-based matrices intermixed with lignocellulosic fibers (Jawaid & Khalil, 2011) - like wood dust - and are the main focus for research and development due to their renewable, biodegradable, and non-toxic nature (Jayamani et al., 2015). Alternatively, synthetic composites, so-called Fiber-Reinforced Plastics (FRP), consist of synthetic matrices intermixed with inorganic fibers (Jawaid & Khalil, 2011). Despite their significant mechanical properties over NFC, there are several drawbacks associated with synthetic composites; they cause pollution, emit toxic byproducts, require enormous amounts of energy for production, have poor recyclability of merely 25%, and negatively impact the depletion of finite petroleum (Zwawi, 2021).

Biocomposites hold the potential to eradicate the dependency on synthetic matrices and fibers, but their inferior mechanical properties and poor UV/water resistance lead to early degradation, affecting their long-term durability and possible applications (Zwawi, 2021). Consequently, the scientific community has been paying considerable attention to developing various methods to alter the properties of biocomposites to attain properties similar to those of synthetic composites (Jayamani et al., 2015).

Particularly biocomposite facade panels hold the potential not only to address insulation and aesthetic shortcomings, as the majority of the building stock is visually outdated, but also to improve the acoustic performance, thermal performance by offering shading, wind conditions, biodiversity, psychological well-being, and mitigate the urban heat island effect.

This circular approach of utilising waste wood dust fibers to create insulating biocomposite facade panels could offer housing corporations an environmentally-friendly - and potentially financially-friendly - solution for upgrading their residential stock to meet their environmental objectives, while simultaneously pushing the construction sector towards a sustainable future.

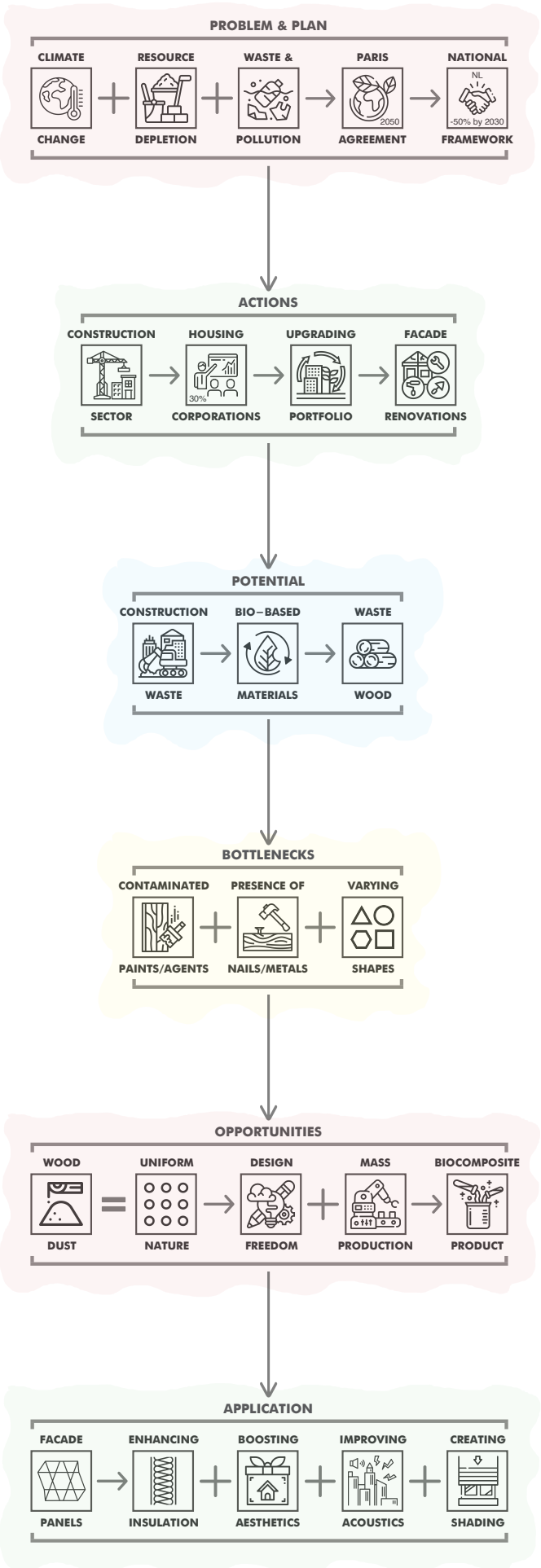


Figure 1.1.2. Introduction scheme. Icons retrieved from Flaticon.com

Current literature is widely spread: from examining the influence of wood dust content on the biocomposites' mechanical properties, UV/water resistance, and acoustic performance, to the effect of various fiber and matrix modification treatments and different fabrication methods. Many of these projects repeat similar experiments with minor changes in material composition or treatment content, and share the same objective of enhancing the biocomposites' properties. None of them have yet considered the bigger picture and combined all the individual pieces of research, making it hard to obtain insight into the potential properties of wood dust grafted biocomposite facade panels, especially because many of these interventions have non-linear and conflicting impact on its performance.

For this reason, this thesis aims to provide insight into the potential properties of wood dust crafted biocomposite facade panels by developing an interactive design tool that guides its user toward optimal performance based on various material properties and design options.

In a broader perspective, this research on wood dust crafted biocomposite facade panels is rather part of preliminary research than the main focus of this thesis, which instead focuses on improving the design process of building facades in general. Within this thesis, biocomposites serve as a relevant and sustainable case study material, where any facade material could have been chosen.

1.2 Problem analysis

Aligning with today's world, where everything is more data-driven than ever before, designers increasingly rely on quantifiable metrics for decision making, especially in the early design stage (Turrin et al., 2020). However, given the immense architectural impact of building facades, qualitative metrics will always remain of great importance. As an alternative to the traditional design framework where designers make decisions based on their knowledge and experience, optimisation frameworks employ genetic algorithms to search for optimal solutions in vast design spaces. Exploring design alternatives based on both geometry and performance is often challenging. While manual exploration of the design space is impractical due to the huge number of design alternatives, optimisation frameworks overlook a huge part of the qualitative aspects that are crucial in facade design. However, performance-driven design exploration frameworks are posing a solution, by leveraging machine learning techniques, giving designers the ability to navigate the entire design space according to geometry typology and performance (Danhaive & Mueller, 2021; Turrin et al., 2020).

Current literature related to the design process of building facades, generally focuses on multi-objective optimisation (Brown et al., 2016; Vazquez & Walker, 2021), potentially missing out on valuable design alternatives with high qualitative value. In some cases (Minaei & Aksamija, 2020; Choo & Janssen, 2015), surrogate models are employed to optimise the computational process of multi-objective optimisation frameworks, by substituting slow performance simulations with fast approximation models. Nevertheless, these frameworks still revolve around optimisation, and have the same limitations of overlooking crucial qualitative aspects. In very rare cases, when performance-driven design exploration frameworks are used in facade design, several critical parts are missing, making them significantly less effective.

As an example, Bertagna et al. (2021) employed a Self-Organising Map (SOM) in the design process of a load-bearing concrete diagrid facade, as illustrated in Figure 1.2.1, clustering 12,758 design variants onto a two-dimensional network of nodes, according to their geometric characteristics.

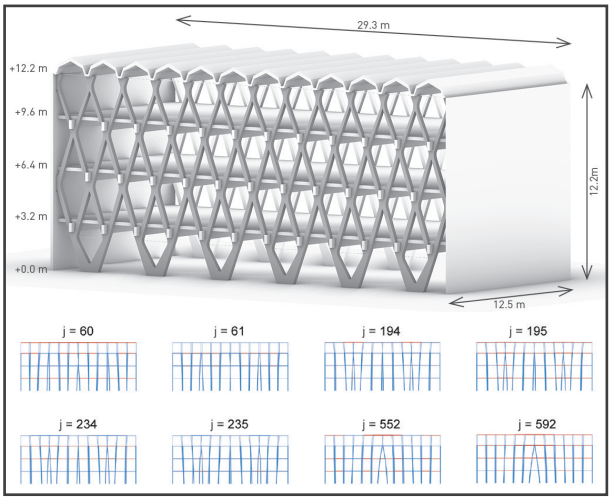


Figure 1.2.1. Diagrid facade. Adapted from "*Holistic Design Explorations of Building Envelopes Supported by Machine Learning*", by Bertagna et al., 2021 [17].

After assessing each design alternative' structural and thermal performance, the self-organising map allows for exploration, considering qualitative and quantitative metrics. However, the absence of a surrogate model not only makes the design framework computationally intensive, but also limits its ability to adapt to more complex design problems.

Additionally, non-geometry related design variables are often excluded from the design process due to limitations of the self-organising map, which clusters based solely on geometry features. This limits design optimisation, as non-geometry related design variables, such as facade panel perforations, hold the ability to influence quantitative performance as well, such as affecting the Sound Pressure Level (SPL) in front of the facade.

Furthermore, the complex relationship between geometry - driven by design variables - and performance lacks interpretability, for both optimisation frameworks and performance-driven design exploration frameworks. This is particularly true for complex design problems with high-dimensional design spaces and multiple performance criteria, as their relationship is often highly non-linear and conflicting. While both multi-objective optimisation' genetic algorithms and surrogate models are able to effectively showcase which combinations of design variables result in well-performing geometries across various performance criteria, they are, without employing sensitivity analysis tools, unable to inform the designer about how each individual design variable affects the performance.

An enhanced interpretability between geometry and performance would have a significant beneficial impact on the design process by offering the designer the ability to reconsider their initially selected design variables, as some of them might not have as big of an impact on the performance as expected and might predominantly affect the geometry, therefore able to be freely tweaked within their original domain. Selecting relevant design variables, affecting both geometry and performance, is crucial as most design problems require high-dimensional design spaces and Self-Organising Maps (SOM) face the curse of dimensionality where the number of design variables have to be limited to prevent sparsity and to ensure it is able to reflect the design space correctly, potentially selecting design variables that predominantly affect geometry, and neglecting those that are highly-relevant, thus significantly affecting the design process' efficiency.

As an example, Turrin et al. (2020) employed a performance-driven design exploration framework comprising a Self-Organising Map (SOM) and a Multi-Layer Perceptron (MLP), in the design process of a long-span roof structure of an indoor arena. Partly due to the curse of dimensionality, as it is asserted that a nine-dimensional design space can be effectively reflected by a two-dimensional SOM, only 4/30 design variables were considered without even knowing their effect on performance.

By solely focusing on design variables that are truly significant, designers can mitigate noise introduced by non-relevant design variables. This not only makes the design process more efficient, but also creates design vectors with stronger underlying patterns, enhancing the surrogate model's performance approximation capabilities. This is particularly beneficial for design problems with high-dimensional design spaces, as their high complexity generally make them more prone to overfitting and a lot harder to generalise.

Additionally, allowing designers to understand the complex non-linear relationships between design variables and performance criteria early on in the design process, based on just a small representative dataset of design vectors and their simulated performance values, would allow for an optimised stratified sampling logic for training the surrogate models. This not only reduces the amount of training data required, reducing the computational demands, especially for higher-complex design problems with detailed simulations, but also improves the surrogate model's generalisation capabilities by focusing on the most relevant design vectors.

Furthermore, it would empower designers with actionable insights to explore the design space faster and more effectively, allowing them to adjust design variables more strategically towards finding optimal solutions from a qualitative and quantitative perspective, facilitating more informed decision making in the early design stage, potentially leading to better architectural and sustainable solutions.

Currently, there are 3 main types of performance-driven design exploration frameworks which are used in the AEC sector, as illustrated in Figure 1.2.2. They all commence similarly by employing a parametric model, controlled by geometry-related design variables, creating a vast design space with numerous design alternatives, and all have some sort of performance assessment, often utilising surrogate models to optimise the computational process. Apart from their similarities, each design framework has its distinct characteristics and qualities, and choosing between them fully depends on the specific nature of the design problem.

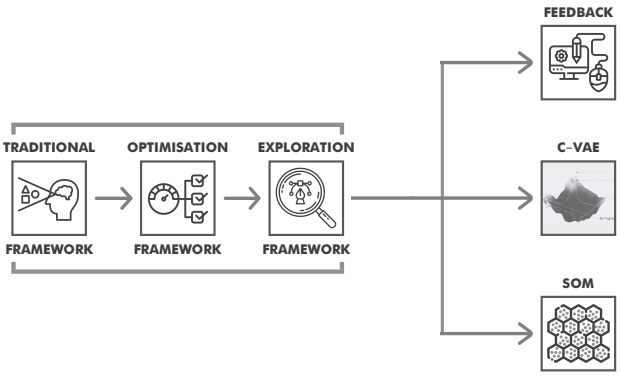


Figure 1.2.2. Design frameworks scheme. Icons retrieved from Flaticon.com

The first performance-driven design exploration framework that is utilised in the AEC sector is the real-time feedback framework. This design framework allows the designer to tweak design variables freely with real-time changing performance approximations guiding the design exploration process towards high performance. Despite empowering the human designer with control over the design process and AI assisting - rather than taking over

control completely, enhancing the design framework's reliability and making it easier to adapt to from a traditional design framework - it has two significant limitations: a huge number of design alternatives will not be explored and it is difficult for the designer to understand how to adjust the design variables to achieve higher performance, making the design framework more suitable for relatively simple and low-dimensional design problems.

The second performance-driven design exploration framework that is utilised in the AEC sector is called the performance conditioned variational autoencoder framework (c-VAE). This design framework, particularly employed for structural design problems (Danhaive & Mueller, 2021; Balmer et al., 2024), is able to compress high-dimensional data into low-dimensional representations, thereby creating distributions within a so-called latent space. Once created, it is able to project back out of this latent space into high-dimensional data again. In simple terms, the c-VAE projects design alternatives, paired with their approximated performance, from a huge design space onto a three-dimensional surface and focuses on the ones that perform well, based on an adjustable predefined threshold, enabling the designer to explore the design space. Despite offering excellent guidance towards high quantitative performance, the design framework lacks overview of the design space from a qualitative perspective, making it highly performance-oriented. Although this would be perfectly fine for most structural design problems, it is less suitable for most facade design problems due to their superior architectural impact.

The third performance-driven design exploration framework that is utilised in the AEC sector is the Self-Organising Map (SOM) framework. By clustering design alternatives onto a two-dimensional network of nodes, according to their geometric characteristics, the designer is able to navigate the entire design space according to geometry typology and approximated performance. While offering an excellent overview of the design space from a qualitative perspective, their design exploration process, consisting of between-cluster and inside-cluster exploration, lacks balanced human-AI interaction. Particularly, inside-cluster exploration, involving the exploration of hundreds of similar design alternatives, is overly AI-dominant and lacks interactivity with the human designer, making the design framework less effective and intuitive.

Consequently, the problem statement of this thesis is: "*The implementation of performance-driven design exploration frameworks in facade design is in a very early stage, and significant enhancements must be made to make them more effective.*"

1.3 Objective

In light of these challenges, this thesis proposes a novel performance-driven design exploration framework, inspired by the SOM framework, employed in the design process of a biocomposite facade. Instead of Multi-Layer Perceptrons (MLP), today's foundational deep learning models for approximating non-linear functions, this thesis hitches on recent groundbreaking research by Liu et al. (2024), from Massachusetts Institute of Technology, introducing Kolmogorov-Arnold Networks (KAN) as a highly-promising surrogate model alternative.

Different from MLPs static activation functions on neurons and learnable weights on edges, KANs have static sum operations on their neurons and learnable 1D parametrised B-spline univariate activation functions on their edges between their neurons, allowing them to fully understand how each individual input variable affects the surrogate model's predicted output (Liu et al., 2024). Where SOM-MLP-based design frameworks are highly inefficient at providing insights into the relationship between geometry and performance, due to their complex architectures based on weights, and the addition of sensitivity analysis tools result in over-engineered frameworks with poor usability, SOM-KAN-based design frameworks hold the ability to overcome these hurdles as an all-in-one solution.

Additionally, making the surrogate model's decision-making process transparent, unlike sensitivity analysis tools which are only able to offer output-level insights, fosters trust between human designers and AI, crucial for AI to be smoothly integrated into current design practices, especially as many designers remain hesitant to shift from traditional workflows to AI-driven workflows due to concerns about AI's reliability coupled with their partial loss of control over the design process.

Furthermore, Liu et al. (2024), Yang et al. (2024), Bozorgasl & Chen (2024), and Samadi et al. (2024) claim that KANs are capable of outperforming MLPs from an accuracy, convergence, neural scaling efficiency, and a catastrophic forgetting perspective as well, achieving this with much smaller model architectures. This not only reduces computational demands significantly, particularly due to its faster convergence, minimising the amount of training data required to yield acceptable prediction accuracies - assuming sufficient representativeness of the dataset - as well as for optimising the surrogate model's stratified sampling logic, but also make them more adaptable to higher-complex design problems and a lot easier to work with.

In addition to providing interpretability between geometry and performance through the integration of Kolmogorov-Arnold Networks (KAN) into

a SOM-based design framework, this thesis proposes a novel design exploration process, consisting of an orientation and fine-tuning phase, not only aiming to balance human-AI interaction more efficiently, making design exploration more interactive, thereby more effective and intuitive, but also to include less-geometry related design variables, enhancing its optimisation capabilities.

Consequently, the objective of this thesis is to develop a novel performance-driven design exploration framework integrating Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN), that advances the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, and usability, to facilitate more efficient decision-making in the early design stage.

1.4 Research questions

Hence, the main research question of this thesis is: *"How can a performance-driven design exploration framework, integrating Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN), advance the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, and usability, to facilitate more efficient decision-making in the early design stage?"*. Following this, the sub-questions are:

1. *"How can a parametric model using Rhino 3D and Grasshopper create a vast design space of biocomposite facade design alternatives, controlled by geometry-related design variables?"*

2. *"How can a Self-Organising Map cluster these biocomposite facade design alternatives onto a two-dimensional network of nodes, according to their geometric characteristics?"*

3. *"How can a Kolmogorov-Arnold Network predict the performance of these biocomposite facade design alternatives, while providing intepretability between geometry and performance?"*

4. *"How can interpretability between geometry and performance enhance the design exploration process during the fine-tuning phase, guiding the designer interactively towards optimal performance?"*

1.5 Relevance

The main relevance of this thesis lies in its ability to not only advance the facade design process by integrating Kolmogorov-Arnold Networks (KAN) as a surrogate model within a Self-Organising Map (SOM) based design framework, which is currently the most advanced performance-driven design exploration framework in facade design, overcoming limitations associated with its computational efficiency and adaptability to higher-complex design problems, but also to offer a design framework that can be extended to advance design practices across the entire AEC sector by offering more accurate performance approximations and more informative interactions between human designers and AI, ultimately making the design process more effective, intuitive, reliable, and less computationally intensive.

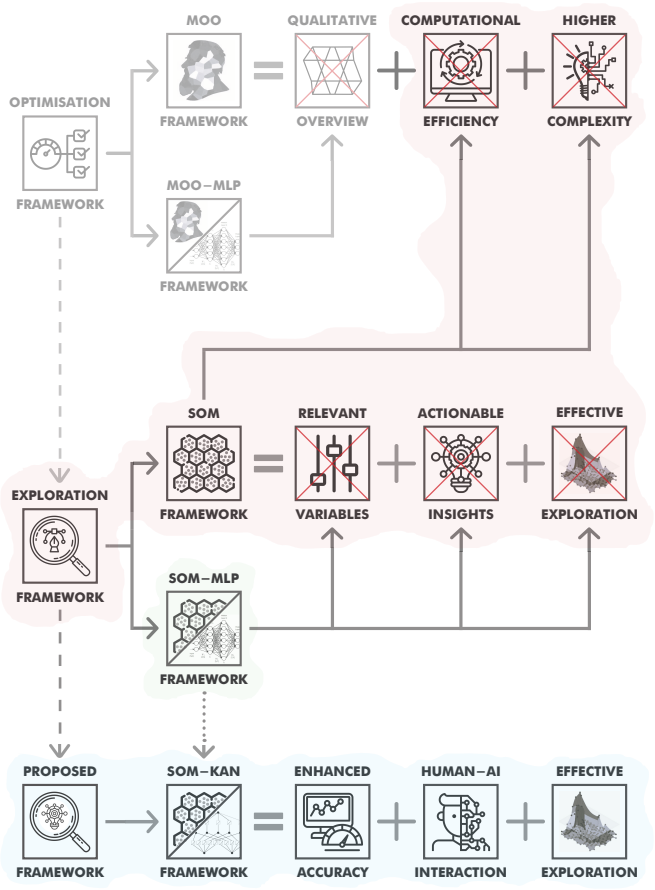


Figure 1.5.1. Relevance overview scheme. Icons retrieved from Flaticon.com

In a broader perspective, this thesis contributes to the research community by demonstrating the potential of Kolmogorov-Arnold Networks as a promising surrogate model alternative to traditional Multi-Layer Perceptrons, by systematically comparing KANs with identical MLPs, as well as those with four, eight, and sixteen times as many nodes, across varying levels of performance approximation complexity, evaluating R^2-scores at intervals of 300 samples up to 6,750 samples, offering insights into KAN's efficiency in approximating non-linear functions as training data and performance approximation complexity increases, relative to MLPs.

Additionally, this thesis advances the body of knowledge on biocomposites, through the development of an interactive design tool, providing insight into the potential properties and applications of those crafted from wood dust and polylactic acid.

1.6 Methodology

In phase I, a parametric model using Rhino 3D and GH will be employed to create a vast design space of biocomposite facade design alternatives, controlled by 9 out of 19 optional geometry related design variables, creating a nine-dimensional design space. Aiming to consider 27,000 design alternatives, each variable will be assigned an own range and interval.

In phase II, a SOM will be trained using all possible combinations of normalised geometry-related vectors of 3 out of 9 design variables, identified as the most influential on geometry, clustering 125 design variants onto a two-dimensional 15x15 hexagonal node network, according to their geometric characteristics. After optimising the SOM's clustering performance through a hyperparameter tuning process, the 125 node design vectors, together with 6,625 vectors sampled during stratification to ensure proper generalisation - representing 25% of the entire dataset - will be used to conduct performance simulations for material use (GH), solar heat gains (Ladybug), and sound pressure level (PachyDerm), creating representative labeled datasets for training three KANs, one for each metric.

In phase III, the KAN models will be trained to approximate the performance of all design alternatives with maximum accuracy. After optimising their prediction accuracy through a hyperparameter tuning process, they will be systematically compared with various MLP models to evaluate their prediction accuracy as training data and comlexity increases. Following this, the KAN models will be used to create plots showing the percentual impact of each design variable on performance predictions, enabling designers to identify and focus on relevant design variables that affect both, while also empowering them with actionable insights to adjust design variables strategically toward optimal performance. These plots will be validated through sensitivity analysis results and personal assessment.

In phase IV, two design exploration phases will be employed: an orientation phase allowing designers to navigate the SOM according to geometry typology and approximated performance, facilitating fast design orientation based on variables with high influence on geometry, and a fine-tuning phase focusing on smaller geometric changes within a selected best-performing design from the orientation phase, allowing designers to adjust design variables strategically based on real-time performance feedback and actionable insights provided by the KAN models. To validate the design exploration framework's effectiveness, it will be employed in practise and compared with traditional SOM-based between cluster and inside cluster design exploration phases with feedback obtained through a questionnaire.

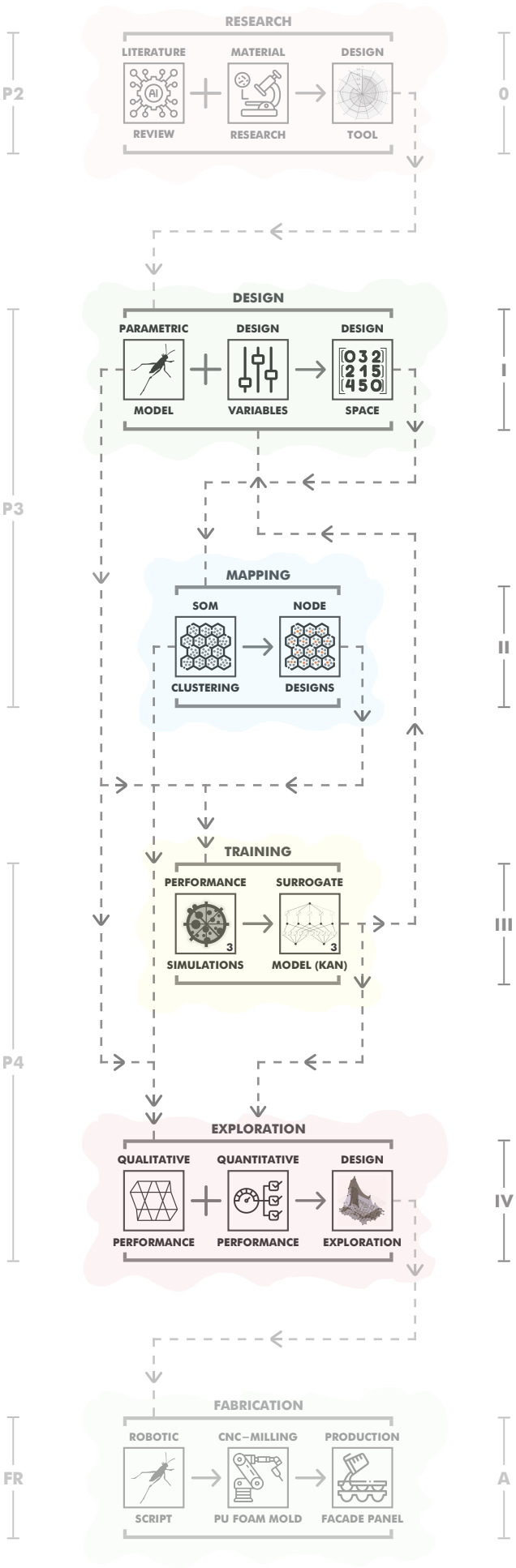


Figure 1.6.1. Methodology scheme. Icons retrieved from Flaticon.com

2

Literature review

2.1 Introduction

This literature review delves into the theory and practical implications underlying the proposed design framework, starting with an overview of the general principles of Artificial Intelligence (AI), its subfields Machine Learning (ML) with a particular focus on Deep Learning (DL), and Neural Networks (NN), followed by delving into both the theory as well as the training process of Self-Organising Maps (SOM), and Kolmogorov-Arnold Networks (KAN), addressing their theory, training process, neural scaling laws, and their enhanced performance compared to traditional Multi-Layer Perceptrons (MLP).

Artificial Intelligence (AI), also known as the science of creating smart machines that imitate human intelligence, has experienced significant developments in the last few years thanks to huge advancements made in GPU performance (Rafsanjani & Nabizadeh, 2023). Despite AI's recent growth, the principle of artificial intelligence as we know it today, was actually already introduced in the 1940s; where a lack of sufficient computational power hindered its breakthrough back then. Nowadays, numerous powerful AI-applications have already become indispensable in many people's daily workflows, by taking over various repetitive and time-consuming activities, allowing people to concentrate on the more important parts of their work. Especially in disciplines such as computer vision, robotics, and gaming, artificial intelligence has experienced huge developments (Rafsanjani & Nabizadeh, 2023). Although not as extensively, the use of AI has become increasingly prevalent in the field of architecture, engineering, and construction (AEC) as well. According to Darko et al. (2020), AI in the domain of AEC is primarily utilised to enhance the development of innovative and optimised architectural and structural designs, improve the construction and operational safety and expenses, minimise embodied carbon and energy consumption, and increase construction speed. The potential of artificial intelligence is as huge as ones imagination and capable of offering the AEC sector novel methods to solve highly complex problems by utilising extensive data to make informed design decisions. Despite the power of AI, emotion, empathy, and intuition - as part of human intelligence - can never be substituted completely, especially in the case of designing tasks (Maadi et al., 2021). It is anticipated that artificial and human intelligence will go hand in hand in the future, leading to reskilling opportunities rather than AI taking over control completely.

As a foundational branch of AI, machine learning (ML) focuses on leveraging huge amounts of data to learn underlying patterns, make predictions, and improve decision-making, without explicitly being programmed (Bastanlar & Ozuysal, 2013). Machine learning can be categorised into three distinct groups: supervised learning which utilises labeled data - where both the input and the desired output are known - to train models to predict outputs for unseen input data, unsupervised learning which leverages unlabeled data - where only the input is known - to train models to identify their underlying patterns, and reinforcement learning which learns from an interactive environment where rewards are maximised and penalties are minimised. Deep learning (DL) overarches all three categories, and focuses on mimicking the human brains' neural networks, particularly effective for highly complex tasks such as image and speech recognition (Bastanlar & Ozuysal, 2013). Over the years, various machine learning models have been developed for various tasks such as classification, prediction, clustering, and dimensionality reduction. Each of them have their own distinct qualities, heavily reliant on the quality of the dataset, the objective, and the time available for computation. The art lies in understanding their qualities and choosing the right ML model in each situation.

Artificial Neural Networks (ANN), part of DL, focuses on mimicking the human brains' neural networks and are employed for tasks with high complexity (Bastanlar & Ozuysal, 2013). They consist of interconnected neurons, also known as nodes, organised in multiple layers. Artificial neural networks generally consist of an input layer with multiple nodes, directly related to the number of input variables, two or more hidden layers with adjustable numbers of nodes, and an output layer with a single node, contingent upon its task. Deep neural networks employed for classification purposes for example, consist of two or more output

layer nodes - directly linked to the number of possible classes, while those used for prediction tasks only contain one outer layer node - related to the single normalised prediction value (Samek et al., 2021). Each hidden layer node has its own activation function, which allows the neural network to capture non-linear and complex patterns in the data. As shown in Figure 2.1.1, all nodes are interconnected, with - initially randomised - weights at the edges and biases at the nodes, which are added to the product of features · weights. During training, the neural network learns to adjust both the weights and biases to optimise data approximation in the outer layer node. This is done by backpropagation, which consists of calculating the difference between the predicted output and the desired output, generally measured using the mean squared error (MSE) based on the Euclidean distance function (Samek et al., 2021).

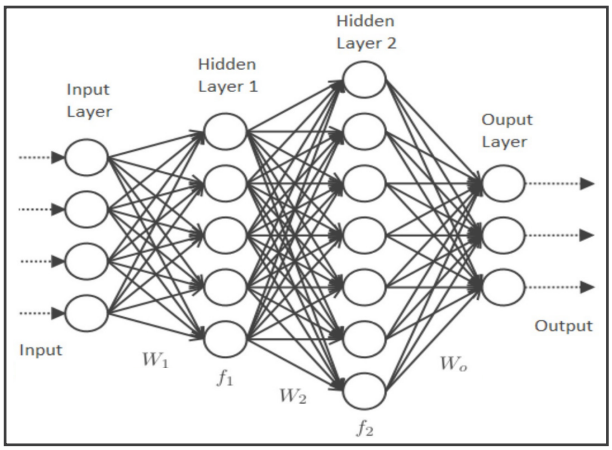


Figure 2.1.1. Neural network. From "*Medium*", by A. Chow, 2020 (https://miro.medium.com/v2/resize:fit:720/format:webp/1*lnhJA8vzxdniOTqygcdX2Q.png).

2.2 Self-Organising Map

Self-Organising Maps (SOM) are essentially types of unsupervised deep topological neural networks employed for two-dimensional clustering tasks, based on the similarity of the data, generally measured by the Euclidean distance function (Miljkovic, 2017). According to Turrin et al. (2020), they can also be considered as dimensionality reduction techniques, by projecting high-dimensional design spaces onto two-dimensional networks of nodes, also reffered to as feature maps. In essence, they work similarly to k-means clustering (KMC), although more constrained.

Adapted from an online lecture from professor H.R. Tizhoosh from University of Waterloo, Self-Organising Maps (SOM) typically include a rectangular or hexagonal lattice of neurons organised in a topological structure, as illustrated in Figure 2.2.1. Although both are capable of capturing clusters in the dataset, hexagonal grids typically offer better topological preservation than rectangular grids because they have a smaller difference in the number of adjacent neurons between the edges and the center (Licen et al., 2023).

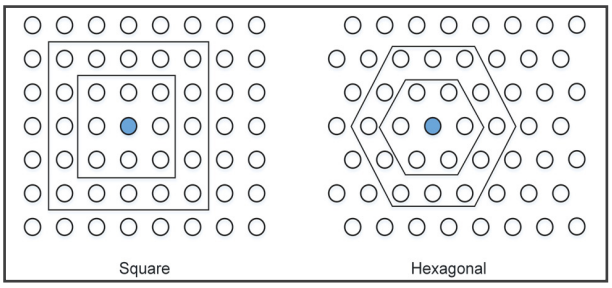


Figure 2.2.1. Most commonly used self-organising map grids. Adapted from "*Brief Review of Self-Organizing Maps*", by Miljkovic, 2017 [74].

Each neuron on the SOM grid, acting as cluster centroids representing similar data points, has its own neighbourhood of adjacent neurons and learns in a smooth and collaborative manner with other neurons. As illustrated in Figure 2.2.2, each input data is connected to each neuron on the grid with randomised weights at the start, similar to supervised neural networks. During training, each input data is assigned to the Best Matching Unit (BMU) with the closest weight factor, generally measured by the Euclidean distance function. Both the winning neuron, which shows most similarity to the data, and its neighbourhood have their weights updated to move closer to the data. This update works most strongly on the winning neuron and does not affect neurons outside the neighbourhood. In the beginning, each neuron competes to represent the input data, resulting in a relatively flat Gaussian distribution as it tries to collaborate with its neighbourhood. Towards the end, each neuron cluster centroid represents a pointier Gaussian distribution. As more input data is assigned to certain neurons, a topographical map emerges, showing different clusters in the dataset, as illustrated in Figure 2.2.3.

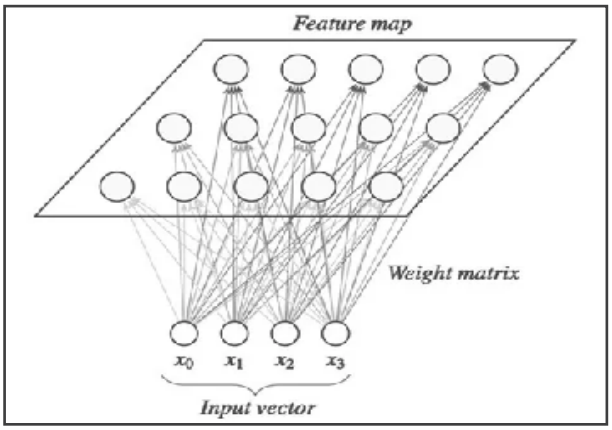


Figure 2.2.2. SOM structure. From "*Medium*", by A. Ali, 2019 (https://miro.medium.com/v2/resize:fit:640/format:webp/1*0PdY0c_2FFZ1-BY-_j0yRA.png).

To prevent Self-Organising Maps from overfitting, a subset of ~20% of representative input vectors is generally used during training. To ensure that

each variable within these input vectors contributes equally to the distance calculations during training, they are typically normalised (0-1). During training, the self-organising map learns to cluster based on the underlying patterns of this subset. After finalising training, the weights are set and the SOM allows for clustering the remaining dataset (Licen et al., 2023). During training, one can use a sequential approach where individual input vectors are presented one after another and weights are updated right after, or batch training where all input vectors are presented before any weight adjustments are made (Licen et al., 2023). While both methods have their own advantages, batch training is usually recommended for its faster convergence and no need for learning-rate tuning, as all vectors are presented at the same time (Kohonen, 2013).

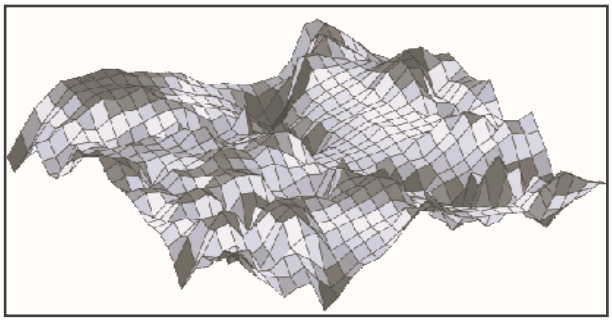


Figure 2.2.3. SOM topographical surface. From "*Viscovery*", by Viscovery, n.d. (https://www.viscovery.net/bilder/somine/UnfoldedMap.png).

To optimise the clustering performance of SOMs, given that a batch training algorithm (BTA) is used, one can iterate through different hyperparameter values for neighbourhood radius and number of epochs, and keep the model with the least amount of error. While the Mean Squared Error (MSE) can be used in supervised learning to compare the predicted output against the desired output to compute the loss, Self-Organising Maps are unable to use this error metric due to their unsupervised nature. However, quantisation error (QE) and topographic error (TE) can be used instead to evaluate the SOM's clustering performance. With regard to these error metrics, QE relates to the average distance between each input vector and the BMU, and influences the accuracy of input space representation, and TE relates to the amount of input vectors for which the first and second BMU are not adjacent on the SOM grid, and affects the quality of topological preservation (Licen et al., 2023). In other words, QE concerns how close the input vectors are to the neurons on the SOM grid, while TE concerns how well similar input vectors are close together on the SOM grid. It is recommended to use a multi-objective optimisation approach, where both error metrics are combined into a weighted sum and minimised during the hyperparameter tuning.

Finally, self-organising maps can adapt to various network sizes, determined by the number of neurons in columns and rows. Choosing the appropiate SOM-network size is essential and heavily reliant on the number of input data points that need to be clustered. If the number of nodes on the SOM grid is too high compared to the number of input data samples, some nodes may capture no data at all, indicating that the SOM might be overly complex, potentially causing overfitting, reduced generalisation, and poor interpretability. On the other hand, too little number of nodes could lead in a map that fails to capture the intricate relationships of the data (Licen et al., 2023). Choosing the appropiate SOM-network size is quite challenging, especially because the number of input data points is not the only aspect influencing this decision-making. As a general guideline, Kohonen (2013) suggests that the total number of nodes on the SOM grid should be around $5 \cdot \sqrt{}$(number of input samples). However, it is recommended to increase the size of the SOM-network if the dimensionality of the input vectors is above average.

2.3 Kolmogorov-Arnold Network

Kolmogorov-Arnold Networks (KAN) are types of supervised neural networks, also known as fully connected feedforward neural networks, which can be used for both classification and prediction tasks (Liu et al., 2024). Different from MLPs universal approximation theorem, which states that any continuous function regardless of its complexity can be approximated as long as the neural network has one hidden layer and enough neurons (Xu et al., 2022), KANs are, not unsurprisingly, based on the Kolmogorov-Arnold representation theorem, named after two very prominent mathematicians: Andrey Kolmogorov and Vladimir Arnold.

This theorem established that: "If f(x) is a multi-variate continuous function on a bounded domain, then f(x) can be written as a finite composition of continuous functions of a single variable and the binary operation of addition." (Liu et al., 2024, pp. 3-4). In other words, a complex function depending on multiple variables, can be precisely rewritten by combining less complicated functions that depend on just one single variable (Yang et al., 2024). This theorem can be expressed as follows:

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

Where the complex multi-variate continuous function is expressed as $f(x_1, \ldots, x_n)$, the simple univariate functions are given by $\phi_{q,p}(x_p)$, and where Φ_q combines all univariate functions (Liu et al., 2024).

Different from MLPs static activation functions on their neurons and learnable weights on their edges, Kolmogorov-Arnold Networks (KAN) have static sum operations on their neurons and learnable 1D parametrised B-spline univariate activation functions on their edges between their neurons, allowing them to understand how each input variable affects the output (Liu et al., 2024). Consequently, KANs hold the ability to address the black box problem, providing insights into the decision-making process within neural networks. Because of their unique architecture, KANs require significantly less parameters than MLPs, referring to the number of hidden layers and neurons, leading to better generalisation and interpretability capabilities (Chen & Bozorgasl, 2024). At the foundation of KANs are the learnable polynomial B-spline activation functions, as illustrated in Figure 2.3.1, which are excellent for data fitting because of their remarkable smoothness and accuracy in representing highly-complex functions (Liu et al., 2024).

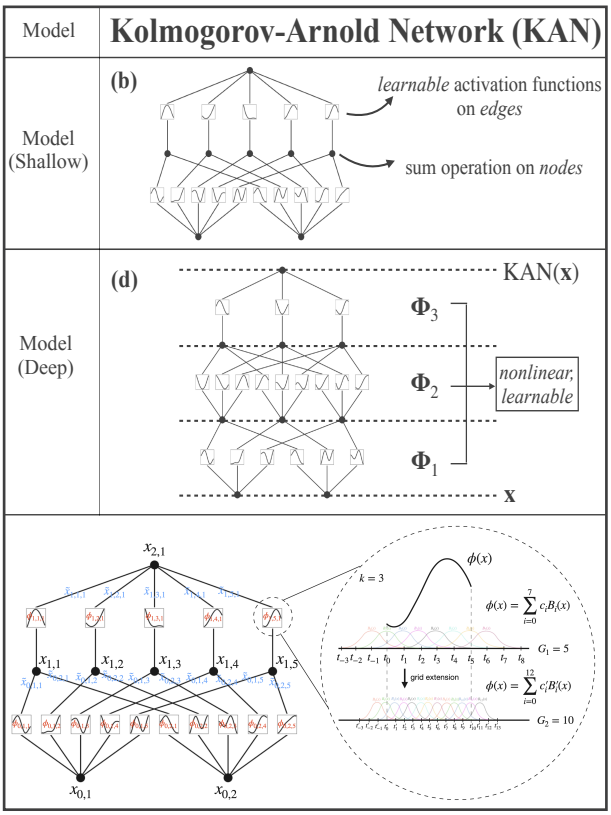


Figure 2.3.1. Kolmogorov-Arnold networks (KAN) model overview. Adapted from "*KAN: Kolmogorov-Arnold Networks*", by Liu et al., 2024 [68].

The training process of KANs is similar to that of MLPs, with a specific loss function, optimiser, and epochs. In training, each B-spline activation function tweaks its control points through backpropagation, increasing its accuracy (Liu et al., 2024). In Figure 2.2.9, the training process of a Kolmogorov-Arnold Network with four input variables, a single hidden layer with five nodes, and three output variables, as part of a classification task, is illustrated.

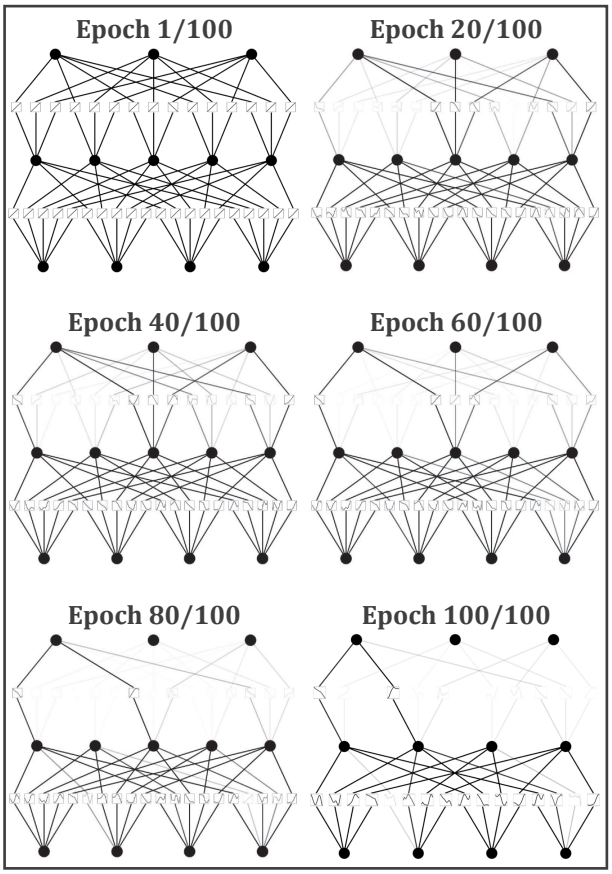


Figure 2.3.2. Training process of Kolmogorov-Arnold network for classification. From "*Daniel-Bethell*", by D. Bethell, 2024 (https://i.imgur.com/v8D90ml.gif).

As can be seen, KANs B-spline activation functions change during training until converging into a particular shape, capturing the underlying patterns of the data with maximum accuracy. As shown, the connections with minimal influence on the prediction are switched off, not only making the model more efficient, but also giving insights into its decision-making process (Bethell, 2024).

In order to adapt to more complex functions, neural scaling laws dictate to increase the size of the neural networks' architecture, by the addition of hidden layers or the increment of their neurons. This is possible for both MLPs and KANs, however, this type of neural scaling is quite slow. Furtunately, KANs hold the ability to do this more effectively. After initialising training with just a few parameters, neural scaling can be applied by increasing the resolution of the B-spline activation functions' grid, also referred to as grid extension, without requiring the larger network to be trained again (Liu et al., 2024).

Additionally, KANs hold the ability to significantly reduce catastrophic forgetting, which is a major concern in machine learning where neural networks forget previously learned information when they are employed for new tasks (Bozorgasl & Chen, 2024). This is because of the local control points of KANs B-spline activation functions, which create a concept of locality, unlike MLPs global ac-

tivation functions. As a simple proof of concept, Liu et al. (2024) have employed a MLP and a KAN for a 1D regression task of five gaussian peaks, presented sequentially in five different training phases. As illustrated in Figure 2.3.3, the KAN is able to avoid catastrophic forgetting perfectly because of their locality, while the MLP fails significantly.

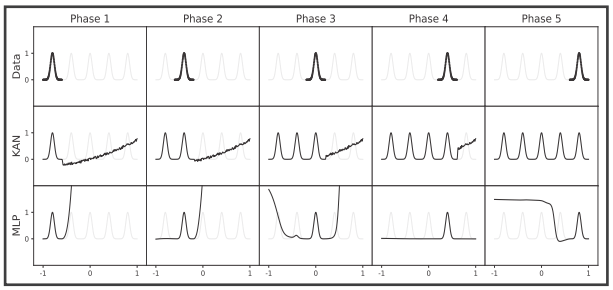


Figure 2.3.3. Comparison of KANs to MLPs based on catastrophic forgetting. Adapted from "*KAN: Kolmogorov-Arnold Networks*", by Liu et al., 2024 [68].

Supported by Liu et al. (2024), Yang et al. (2024), Samadi et al. (2024), and Bozorgasl & Chen (2024), KANs are able to outperform MLPs from a neural scaling efficiency perspective as well. In the following experiment, carried out by Liu et al. (2024), KANs and MLPs are compared by examining their efficiency in representing four mathematical functions through a supervised regression task with labeled x and y values. In other words, where $f(x) = y$: aiming to find the function that transforms x into y. As illustrated in Figure 2.3.4, these mathematical functions increase in complexity (from a to d), therefore requiring varying model dimensionalities. As an example, a KAN [4,4,2,1] with 4 input variables, 2 hidden layers with 4 and 2 nodes respectively, and 1 output variable, aims to represent the most complex function d, KAN [2,2,1] aims to represent a, KAN [2,1,1] aims to represent b, and KAN [100,1,1] aims to represent c. For each mathematical function (a-d), four different MLPs with varying depths, directly related to the number of layers, are employed with the same task. After training, all of the five neural networks are evaluated on the root mean squared error (RMSE) as a function of an increasing number of parameters. With respect to the MLPs, this increment of parameters relates to an increase in architecture width, directly related to the number of neurons in the hidden layers. Concerning KANs, it relates to grid extension, which is considerably faster due to their locality (Liu et al., 2024). Interpreting the results (a to d), it can be confirmed that KANs significantly outperform MLPs from a neural scaling efficiency perspective. While MLPs have relatively small performance improvements as a function of an increasing number of parameters and plateau quickly, KANs show incredibly fast neural scaling, indicating they have a better performance while requiring significantly less parameters (Liu et al., 2024).

Additionally, an experiment was executed by Liu et al. (2024) to evaluate KANs training and neural scaling efficiency compared to MLPs, when employed for a partial differential equation task. Based on graphs a & b from Figure 2.3.5, it can be concluded that, besides achieving lower losses in general, KANs converge faster, meaning they are reaching lower L2/H1 error squared losses faster during training, and that with much smaller architectures. Interpreting the results from graphs c & d, KANs have much steeper scaling laws than MLPs, indicating their superior performance with significantly fewer parameters required (Liu et al., 2024).

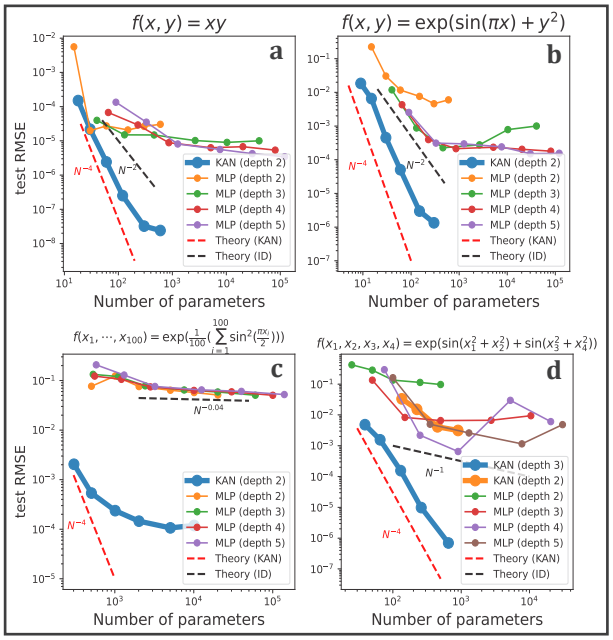


Figure 2.3.4. Comparison of KANs to MLPs based on neural scaling efficiency, with a) KAN [2,2,1], b) KAN [2,1,1], c) KAN [100,1,1], and d) KAN [4,4,2,1]. Adapted from "*KAN: Kolmogorov-Arnold Networks*", by Liu et al., 2024 [68].

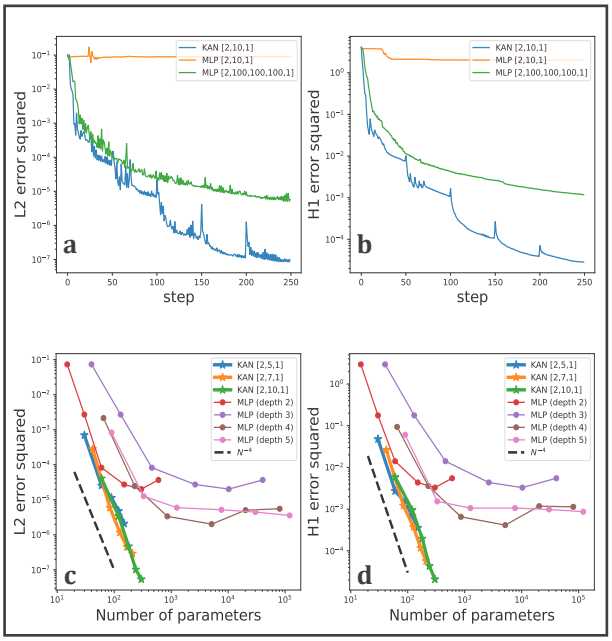


Figure 2.3.5. Comparison of KANs to MLPs based on their training accuracy, with a) KAN [2,2,1], b) KAN [2,1,1], c) KAN [100,1,1], and d) KAN [4,4,2,1]. Adapted from "*KAN: Kolmogorov-Arnold Networks*", by Liu et al., 2024 [68].

3
Biocomposites

3.1 Introduction

This chapter aims to provide insights into the potential properties of wood dust crafted biocomposite facade panels through the development of an interactive design tool that guides its user toward achieving optimal performance based on various material properties and design options. As aforementioned, this study on biocomposites is part of preliminary research and not the main focus of this thesis, which instead focuses on advancing the design process of building facades in general. Within this thesis, biocomposites serve as a relevant and sustainable case study material, where any facade material could have be chosen, aiming to advance its body of knowledge. This chapter is optional to read and starts with an overview of the theory and vulnerabilities of wood dust crafted biocomposite facade panels, followed by delving into the effect of various fiber modification treatments on its water repellency, the effect of various nanoparticle treatments on its UV-resistance and mechanical properties, the effect of annealing time on its mechanical properties, the effect of two fabrication methods on its water repellency and mechanical properties, and the working principle behind the design tool.

Wood dust is a byproduct of wood processing, particularly originating from uncontaminated soft- and hardwood, with a particle size of around 75-300 μm (Singh et al., 2022). Softwood dust fibers generally consist of 42% cellulose, 27% hemicellulose, 29% lignin, and 2% extractives, while hardwood dust fibers contain 46% cellulose, 28% hemicellulose, 22% lignin, and 4% extractives (Dai & Fan, 2014; Bahrami et al., 2020). Where cellulose determines the mechanical properties of the fiber, hemicellulose binds the cellulose fibers together, lignin protects the hemicellulose from exposure to moisture, and the additional extractives offer resistance to decay (Thakur, 2014).

While wood dust fibers are mainly used as reinforcements in biocomposites, bio-matrices are utilised to bind them together (Zwawi, 2021). Bio-matrices are mainly derived from agricultural and byproducts, and are completely hydrophobic, unlike natural fibers such as wood dust (Jayamani et al., 2015). The most commonly used matrices in biocomposites are Polylactic-Acid (PLA), Polybutylene-Succinate (PBS), Poly-Hydroxyalkanotes (PHA), and Poly-Caprolactone (PCL). Especially PLA, a thermoplastic polyester produced from L-lactic and D-Lactic acid (Fortunati et al., 2012), has drawn significant attention from the research community due to its renewable nature, biodegradability, and low energy demand and carbon emissions for production (Zwawi, 2021).

Aiming to replace Fiber Reinforced Plastics, biocomposites' inferior mechanical properties and poor UV-/water resistance, lead to early degradation, affecting their long-term durability and possible applications. Especially building facades, which have huge climatic exposure levels, require facade panels which are anti-fungal, UV and moisture resistant, and capable of maintaining their dimensional stability (Zwawi, 2021).

The hemicelluloses in natural fibers are responsible for the hydrophilic nature of biocomposites, as its hydroxyl groups have a strong tendency to absorb moisture from the environment (Al-Maharma et al., 2019). Consequently, intermolecular hydrogen bonds are formed, resulting in swelling of the fiber. This can cause an inconsistent fiber dispersion in the matrix, negatively impacting its ability for stress transfer at the interfacial regions (Mohammed et al., 2022). This polarity can lead to microcracks, mold growth, dimensional instability, and fiber-matrix debonding, and therefore reduce the biocomposites' mechanical properties and ultimately lead to its early degradation (Azka et al., 2024). Lignin in natural fibers are accountable for the poor UV-resistance of biocomposites, as they absorb 80-95% of UV-radiation (Gonzalez-Lopez et al., 2020). This leads to cutting of the covalent bonds in natural fibers, resulting in loss of molecular weight and reduced mechanical properties.

Polylactic-Acid (PLA), as the bio-matrix in wood dust crafted biocomposites, also contributes to its poor performance, as PLA is very brittle with low plastic deformation (Dhal et al., 2023), and susceptible to photo-degradation. Especially UV below 270nm is absorbed and degraded via the Norrish II reaction in which bonds adjacent to the carbonyl groups in PLA are split, resulting in a decreased molecular weight, reduced mechanical properties, and loss of colour (Wang et al., 2019).

To examine the effect of weathering on the performance of neat PLA and PLA/30wt.% wood dust crafted biocomposites, Gonzalez-Lopez et al (2020) has subjected both materials to accelerated weathering conditions up to 2000 hours. As illustrated in Figure 3.1.1, crystallinity (stiffness) increases after accelerated weathering for neat PLA, with an increase of 232% after 2000h, while the PLA/30wt.% wood dust biocomposite' crystallinity remained almost unchanged after 600h, with only a relative increase of 6% from 10% at 600h to 16% at 2000h. Figure 3.1.2 shows the effect of accelerated weathering on the tensile strength of the same two materials up to 1200 hours. As can be seen, there is a general trend of decreasing tensile strength after accelerated weathering for neat PLA, with a decrease of 66% after 1200h. For the PLA/30wt.% wood dust biocomposite however, only one data point at 1200h was available, showcasing a slight decrease of 33%.

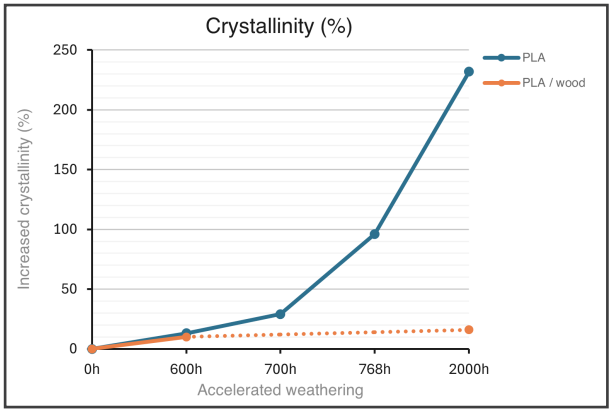


Figure 3.1.1. Effect of accelerated weathering on crystallinity of PLA and PLA/30wt.% wood dust. Adapted from "*Accelerated weathering of poly(lactic acid) and its biocomposites: A review*", by González-López et al., 2020 [44].

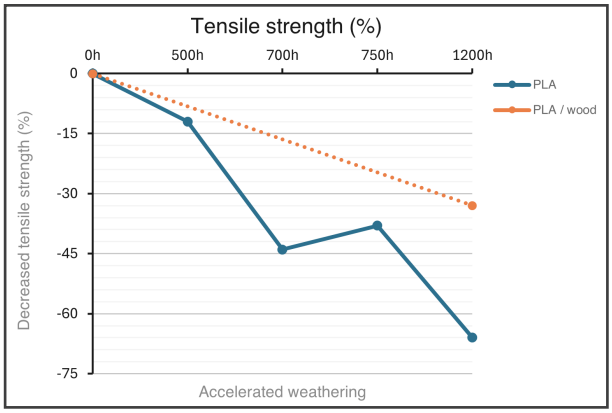


Figure 3.1.2. Effect of accelerated weathering on tensile strength of PLA and PLA/30wt.% wood dust. Adapted from "*Accelerated weathering of poly(lactic acid) and its biocomposites: A review*", by González-López et al., 2020 [44].

3.2 Modification treatments

Aiming to overcome early degradation, the scientific community has been paying considerable attention to creating methods to change the properties of biocomposite fibers, also known as fiber modification treatments, to attain properties similar to synthetic composites (Jayamani et al., 2015). Many treatments have been developed and can be categorised into two main groups: physical and chemical treatments, both with a similar goal of enhansing the water repellency of natural fibers (Mohammed et al., 2022). Both the physical treatments, such as the corona, plasma, UV, and thermal treatment, and the chemical treatments, such as the alkaline, silane, and acetylation treatment, focus on enhancing the fiber-matrix bonding at the interfacial region (Mohammed et al., 2022). Although all treatments have shown significant results regarding water repellency of natural fibers, only the most promising ones are highlighted in this study.

Thermal treatment involves heating the natural fiber to temperatures close to their fiber-degradation temperature, resulting in changes in chemical composition (Al-Maharma & Al-Huniti, 2019). These macromolecular rearrangements of natural fibers significantly enhance their resistance to moisture, but also reduce their mechanical properties and make them even more prone to UV-degradation (Kelleci et al., 2022).

Alkaline treatment involves submerging the natural fiber in a sodium hydroxide (NaOH) solution for a short period of time, altering its molecular structure significantly. This results in fibrillation, leading to higher surface roughness. As a result, water absorption decreases and the ability of stress transfer increases, positively influencing its strength and stiffness (Mohammed et al., 2022). The poor UV-resistance of the fiber remains unchanged, posing a serious danger in outside applications (Singh et al., 2022).

Silane treatment involves covering the micropores on the surface of the natural fiber with Silane coupling agents (SiH_4). These agents react with the fibers' alkoxy and hydroxyl groups, thus creating covalent bindings to the cell wall of the fiber, also known as chemisorption. This closes the micropores that were previously exposed and reduces the number of water-absorbing hydroxyl groups. Additionally, the formed cross-linked network constricts fiber-swelling, consequently enhancing fiber-matrix adhesion and dimensional stability (Mohammed et al., 2022). Similar to the Alkaline treatment, no significant enhancements in UV-repellency of the fiber were observed (Abdallah et al., 2022).

Acetylation treatment involves substituting the natural fibers' hydroxyl groups with acetyl groups, by treating the fiber with acetic anhydride ($CH_3–C(¼O)–O–C(¼O)–CH_3$). While significantly enhancing the water repellency and dimensional stability of the natural fibers, it also negatively impact its UV-resistance (Mohammed et al., 2023).

To overcome UV-degradation of PLA/wood dust crafted biocomposites, the scientific community has been actively focusing on developing new and hybrid combinations of nanoparticles (NP) to fill the micropores on the biocomposites' surface, preventing lignin in the wood dust fibers from absorbing UV-radiation and the bonds adjacent to the carbonyl groups in PLA from being split (Mohammed et al., 2022). Additionally, nanoparticles hold the ability to significantly enhance the mechanical properties, water-repellency, fire-resistance, and anti-fungal performance of biocomposites. These enhancements have led to the implementation of nano-biocomposites (NBc) in fields such as aerospace, automotive, construction, and medical industry (Mohammed et al., 2022).

Various types of nanoparticles have been developed throughout the years, and can be categorised into three groups: nanolayers, nanotubes, and nanofillers. These nanoparticles can originate from polymers, metals, metal-oxides, and carbon; therefore categorised into organic and inorganic groups (Bahrami et al., 2020). While organic NP such as cellulose nanocrystals (CNC), lignin nanoparticles (LNP), and their numerous modified variants, are known for their renewability, biodegradability, and environmentally-friendly nature (Muzata et al., 2023), inorganic NP such as TiO_2, CeO_2, Ag, and ZnO, are non-biodegradable and potentially harmful to the environment (Sringam et al., 2023). Although all the aforementioned nanoparticles have shown significant enhancements in UV-resistance of biocomposites, inorganic nanoparticles generally outperform their organic alternatives. It is because of their unsustainable nature that they are left out of this study, with an exception made for ZnO, which is considered to be safe and non-toxic (Wang et al., 2019; Fortunati et al., 2015; Yang et al., 2023).

Cellulose nanocrystals (CNC) are considered as the most promising organic nanoparticles on the market due to their sustainability, mechanical properties, UV-resistance, easy processability, and low energy demand for production (Ling et al., 2022). They are generally derived from bacterial cellulose, sea life matter, or plant lignocellulosic biomass, through controlled acid hydrolysis. For example, Wang et al. (2019) extracted CNC by mixed acid hydrolysis of microcrystalline cellulose (MCC), Fortunati et al. (2015) derived CNC by acid hydrolysis of Posidonia Oceanica Alga (POA) with an average length of 180nm and diameter of 5nm, and Ling et al. (2022) retrieved CNC from softwood pulp with an average length of 200nm and diameter of 5nm, employing the same method of extraction. Despite CNC's significant performance, resulting in numerous modified variants throughout the years, their hydrophilic nature limits their compatibility with hydrophobic PLA, resulting in decreased mechanical properties (Agbakoba et al., 2023). As a result, many CNC-modified variants aimed to enhance the fiber-matrix dispersion, by modifying its hydroxyl groups through covalent functionalisation using esters, ethers, carboxylic acids, or SiH_4 coupling agents (Sringam et al., 2023).

Surfactant-modified cellulose nanocrystals (s-CNC) are one of these variants which aim to improve the fiber-matrix dispersion by treating CNC with an acid phosphate ester of ethoxylated nonylphenol in a ¼ (wt/wt) ratio, followed by raising the pH to 8.5-9.0 using a 0.25wt.% NaOH solution (Fortunati et al., 2016; Luzi et al., 2016).

Lignin-coated cellulose nanocrystals (L-CNC) are another one of these variants aiming to enhance the distribution of CNC in PLA, by spray drying lignin onto its surface (Boruvka & Prusek, 2016). Lignin is generally derived from pulp and lignocellulosic biomass, and despite it being significantly prone to UV-absorption, lignin reduces agglomeration, which improves fiber-matrix dispersion, and therefore actually improves the biocomposites' UV-resistance. Additionally, Van der Waal interactions are formed between lignin and CNC, leading to water shielding (Ling et al., 2022).

Cinnamate-grafted cellulose nanocrystals (Cin-CNC) are the final CNC-modified variant discussed in this study. By grafting cinnamate groups onto the surface of CNC through an esterification reaction with cinnamoyl chloride, also known as acylation, the fiber-matrix dispersion is improved significantly. As a result, the biocomposites' mechanical properties are enhanced. On top of that, cinnamates are remarkable UV-shielders, by absorbing UV and dissipating heat into the environment (Sringam et al., 2023).

3.3 Results

Aiming to find the most optimal-performing biocomposite from a material properties perspective, aligning with today's environmental and building standards, this study utilises existing research to develop an interactive design tool that guides its user toward optimal performance based on various material properties and design options, including wood dust filler content, type/content of fiber modification treatment, type/content of nanoparticle treatment, annealing time, and type of fabrication.

As the foundation of this interactive design tool, 43 different research papers have been examined and utilised to create graphs, providing insight into the effect of the aforementioned aspects on the performance of neat PLA. As a result, a total

of 226 graphs have been created, as illustrated in Figures A.1-A.130 and A.131-A.226 of appendix A. While the first set of figures examine each research results individually, giving insight into exact numbers (e.g. MPa, kg/m^3, W/m·K, dB/mm), the second set combines these results to give insights into the average relative change of performance compared to PLA. As an example, Figure 3.3.1, also known as Figure A.183, showcases the effect of CNC content on the initial tensile strength of neat PLA. As can be seen, multiple research papers have been utilised, creating an averaged line graph with error regions based on outliers. The different colours in the line graph highlight the various research papers supporting the data. As an examplary interpretation of the results, a rapid increase in tensile strength of PLA up to 30.7% at 1.0wt.% of CNC content is observed, followed by a gradual decrease with an ultimate low of -26.2% at 10wt.% of CNC content.

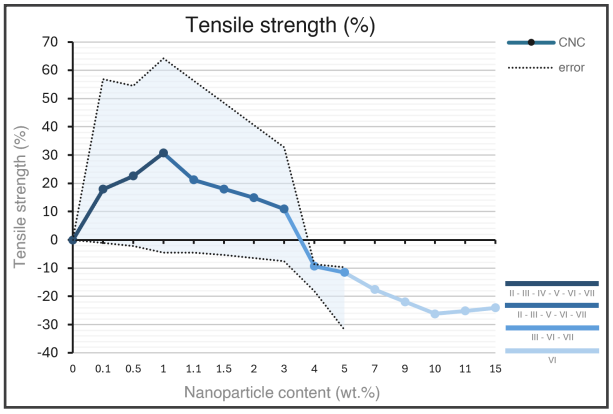


Figure 3.3.1. Combined effect of CNC content in PLA on the tensile strength (relative % compared to neat PLA). Adapted from II. Fortunati et al., 2015 [39]; III. Wei et al., 2016 [100]; IV. Agbakoba et al., 2023 [3]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88].

Before diving into the results, several limitations of this study will be discussed. First of all, a relatively small number of research papers is used. Including more data would significantly enhance its reliability. On top of that, each research paper had their own starting performance value for neat PLA, which resulted in unrealistic relative changes in performance in those with extreme starting values. Despite already aiming to remove all outliers, this needs to be done even more carefully. Finally, this study interpolates the values between known data points up to the maximum wood or nanoparticle content value for each source. Because these maximum content values significantly differ among the sources, it generally leads to scenarios where high content values becomes less reliable, as they are supported by less sources. This undesirable phenomenon, illustrated in Figure 3.3.2, also known as Figure A.176, shows increasing error regions as a result of a decreasing number of sources, up until one source remains. Despite the limitations of this study, it still has a significant contribution to the research community by offering a framework for the design process of optimal-performing biocomposites from a material properties' perspective, with acceptable reliability and the adaptibility to include more information without the need of restructuring the framework of the design tool.

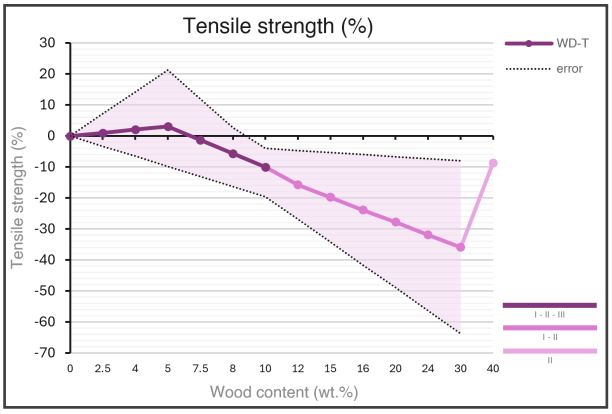


Figure 3.3.2. Combined effect of thermally-treated wood dust (WD-T) content in PLA on the tensile strength (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55]; II. Gregorova et al., 2011 [46]; III. Ayrilmis et al., 2021 [8].

As a starting point, this study aims to acquire as much insights into the effect of (un)treated wood dust content on the performance of neat PLA, examining ten options based on eleven research papers. Option one, illustrated in Figures A.1-A.15, A.24-A.33, and A.131-A.139 of appendix A, is adapted from Abdallah et al. (2022), Jasinksi & Szymanowski (2023), Csizmadia et al. (2013), Mohammadsalih et al. (2023), Narlioglu et al. (2021), and Kelleci et al. (2022), and covers the effect of untreated wood dust content on PLA's performance.

Options two and three, illustrated in Figures A.11-A.12 and A.140-A.143 of appendix A, are adapted from Csizmadia et al. (2013), and cover the effect of 1.0wt.% and 3.0wt.% phenolic resin-treated wood dust content on PLA's performance.

Option four, illustrated in Figures A.1-A.6 and A.144-A.149 of appendix A, option five, illustrated in Figures A.3-A.4 and A.150-A.151 of appendix A, option six, illustrated in Figures A.3-A.4 and A.152-A.153 of appendix A, and option seven, illustrated in Figures A.1-A.6 and A.154-A.159 of appendix A, all originate from the same source (Abdallah et al., 2022), and cover the effect of silane-treated wood dust content on PLA's performance. With regard to this, option four concerns treating wood dust with 3.0wt.% SA in a 50:50vo.% ratio of acetone:water solvent, while option five, six, and seven concern treating wood dust with 1.0wt.%, 2.0wt.%, and 3.0wt.% SE in a 90:10vo.% ratio of ethanol:water solvent respectively.

Option eight, illustrated in Figures A.16-A.22, A.40, and A.160-A.167 of appendix A, together with option nine, illustrated in Figures A.16-

A.23 and A.168-A.175 of appendix A, are adapted from Singh et al. (2022), Lendvai et al. (2022), and Altun et al. (2012), and cover the effect of 2.0wt.% alkaline-treated wood dust content on PLA's performance and the effect of 2.0wt.% alkaline-treated wood dust/rice husk biocomposite (in 50:50wt.% ratio) content on PLA's performance.

Finally, option ten, as illustrated in Figures A.28-A.39 and A.176-A.182 of appendix A, is adapted from Kelleci et al. (2022), Ayrilmis et al. (2021), and Gregorova et al. (2011), and covers the effect of thermally-treated wood dust content on PLA's performance.

Aiming to provide an overview of the effect of (un)treated wood dust content on the performance of neat PLA, a total of ten plots have been created, and illustrated in Figures A.227-A.236 of appendix A. These plots offer insights into the relative change of performance of tensile strength, compressive strength, flexural strength, elongation at break, Young's modulus, bending modulus, flexural modulus, density, thermal conductivity, and water absorption, compared to neat PLA, as a function of increasing wood dust content. As an example, Figure 3.3.3 shows the relative change of tensile strength of all ten options compared to neat PLA. As illustrated, there is a general trend across all options of decreasing tensile strength as a function of increasing wood dust content. This is due to the poor compatibility between the hydrophilic wood dust fibers and hydrophobic PLA, leading to poor interfacial adhesion and stress concentrations (Abdallah et al.,2022; Kelleci et al., 2022; Singh et al., 2022; and Jasinksi & Szymanowski, 2023).

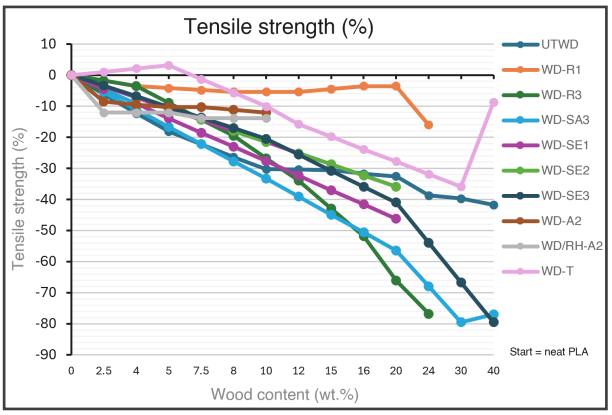


Figure 3.3.3. Combined effect of various PLA fillers (UTWD: untreated wood dust {Adapted from Abdallah et al., 2022 [1]; Jasinski & Szymanowski, 2023 [51]; Csizmadia et al., 2013 [29]; Narlioglu et al., 2021 [79]; Mohammadsalih et al., 2023 [77]; and Kelleci et al., 2022 [55]}, WD-R1/WD-R3: 1/3wt.% phenolic resin-treated wood dust {Adapted from Csizmadia et al., 2013 [29]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from Abdallah et al., 2022 [1]}, WD-SE1/WD-SE2/WD-SE3: 1/2/3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from Abdallah et al., 2022 [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from Singh et al., 2022 [87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from Singh et al., 2022 [87]}, and WD-T: thermally-treated wood dust {Adapted from Kelleci et al., 2022 [55]; Gregorova et al., 2011 [46]; and Ayrilmis et al., 2021 [8]}) on the tensile strength (relative % compared to neat PLA).

As the second step in preparation of the interactive design tool, this study aims to acquire as much insights into the effect of various nanoparticle treatments on the performance of neat PLA, examining nine options based on twenty-six research papers. Option one, illustrated in Figures A.41-A.47, A.51-A.56, A.61, A.98-A.112, and A.183-A.188 of appendix A, is adapted from Marmol et al. (2020), Chi & Catchmark (2017), Sringam et al. (2023), Wei et al. (2016), Fortunati et al. (2012, 2013, 2015), Karkhanis et al. (2018), and Agbakoba et al. (2023), and covers the effect of cellulose nanocrystals (CNC) content on PLA's performance.

Option two, as illustrated in Figures A.44-A.50, A.61, A.108-A.112 and A.189-A.194 of appendix A, is adapted from Luzi et al. (2016), Fortunati et al. (2012, 2013, 2015), and Chi & Catchmark (2017), and covers the effect of surfactant-modified cellulose nanocrystals (s-CNC) content (with a phosphate ester of ethoxylated nonylphenol in a 1:4vo.% ratio) on PLA's performance.

Option three, illustrated in Figures A.57-A.60, A.113-A.119, and A.195-A.199 of appendix A, is adapted from Zijia et al. (2023), Wei et al. (2017), Boruvka & Prusek (2016), and Shojaeiarani et al. (2022) and covers the effect of lignin-coated cellulose nanocrystals (L-CNC) content on PLA's performance.

Option four, as illustrated in Figures A.51-A.56 and A.201-A.206 of appendix A, is adapted from Sringam et al. (2023), and covers the effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content on PLA's performance.

Option five, as illustrated in Figures A.57-A.60, A.71-A.83, and A.207-A.212 of appendix A, is adapted from Yang et al. (2015), Boarino et al. (2022), Daassi et al. (2023), cavallo et al. (2021), and Shojaeiarani et al. (2022), and covers the effect of lignin nanoparticles (LNP) content on PLA's performance.

Option six, illustrated in Figures A.63-A.65, A.84-A.97, and A.213-A.218 of appendix A, is adapted from Luzi et al. (2016), Li et al. (2023), Kim et al. (2019), Yang et al. (2023), Chong et al. (2023), Tan et al. (2023), and Jamnongkan et al. (2022), and covers the effect of zinc-oxide (ZnO) content on PLA's performance.

Option seven, illustrated in Figures A.66-A.70 and A.219-A.224, is adapted from Wang et al. (2019) and Luzi et al. (2016), and covers the effect of CNC-zinc-oxide (CNC-ZnO) content on PLA's performance.

Option eight, as illustrated in Figures A.62 and A.225, is adapted from Fortunati et al. (2012), and covers the effect of CNC + 1.0wt.% silver nanopowder (CNC-Ag) content on PLA's performance.

Finally, option nine, illustrated in Figures A.62 and A.226, adapted from Fortunati et al. (2013), covers the effect of surfactant-modified cellulose nanocrystals (s-CNC-Ag) content (with a phosphate ester of ethoxylated nonylphenol in a 1:4vo.% ratio and 1.0wt.% Ag nanopowder) on PLA's performance.

Aiming to provide an overview of the effect of various nanoparticle treatments on the performance of neat PLA, a total of six plots have been created, and illustrated in Figures A.237-A.242 of appendix A. These plots offer insights into the relative change of performance of tensile strength, elongation at break, Young's modulus, UV-A/UV-B absorption, and water vapour permeability, compared to neat PLA, as a function of increasing nanoparticle content. As an example, Figure 3.3.4 shows the relative change of UV-B absorption of all six options compared to neat PLA. As illustrated, there is a general trend across all options of increasing UV-B absorption as a function of increasing nanoparticle content. This is because nanoparticles fill the micropores on the biocomposites' surface, preventing lignin in the wood dust fibers from absorbing UV-radiation and the bonds adjacent to the carbonyl groups in PLA from being split (Mohammed et al., 2022).

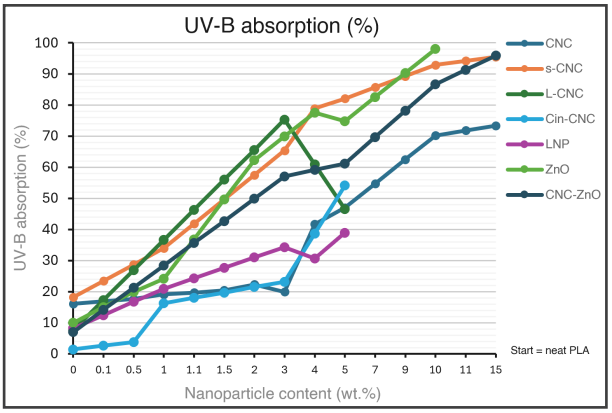


Figure 3.3.4. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from Karkhanis et al., 2018 [54]; Fortunati et al., 2015 [39]; Marmol et al., 2020 [72]; Chi & Catchmark, 2017 [25]; and Sringam et al., 2023 [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from Chi & Catchmark, 2017 [25]; and Fortunati et al., 2015 [39]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from Shojaeiarani et al., 2022 [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from Sringam et al., 2023 [88]}, LNP: lignin nanoparticles {Adapted from Shojaeiarani et al., 2022 [85]; and Cavallo et al., 2021 [24]}, ZnO: Zinc oxide {Adapted from Li et al., 2023 [63]; and Kim et al., 2019 [57]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from Wang et al., 2019 [97]}) on the UV-B absorption (relative % compared to neat PLA).

As the third step in preparation of the interactive design tool, the effect of annealing on the performance of neat PLA was examined. As a result, a total of six plots have been created, and illustrated in Figures A.120-A.125 of appendix A, providing insights into the absolute changes in compressive strength, elongation at break, Young's modulus, thermal conductivity, water absorption, and density, compared to neat PLA, as a function of time.

Aiming to provide an overview, all of the results are combined in Figure 3.3.5, showing the relative change of performance, compared to neat PLA. As illustrated, annealing of PLA significantly enhances its compressive strength and Young's modulus, but negatively affects its elongation at break, thermal conductivity and water absorption.

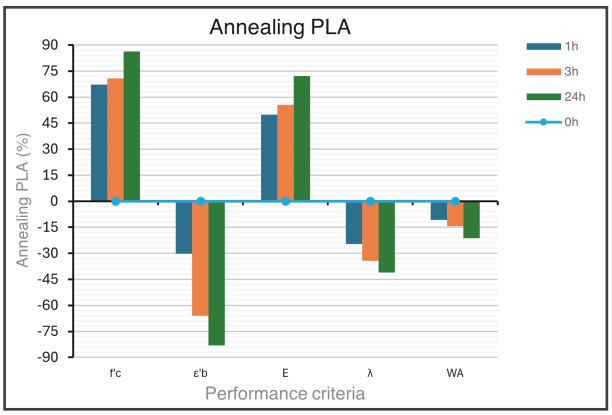


Figure 3.3.5. Effect of PLA annealing time on the compressive strength, elongation at break, Young's modulus, thermal conductivity, and water absorption (relative % compared to neat PLA). Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

As the final step in preparation of the interactive design tool, the effect of injection molding and 3D-printing on the performance of neat PLA was examined. As a result, a total of five plots have been created, and illustrated in Figures A.126-A.130 of appendix A, providing insights into the absolute changes in tensile strength, elongation at break, Young's modulus, thermal conductivity, and sound transmission loss, compared to neat PLA. Aiming to provide an overview, all of the results are combined in Figure 3.3.6, showing the relative change of performance, compared to neat PLA. As illustrated, 3D-printing significantly outperforms injection molding with regards to thermal conductivity, but performs worse in tensile strength, elongation at break, Young's modulus and sound absorption.

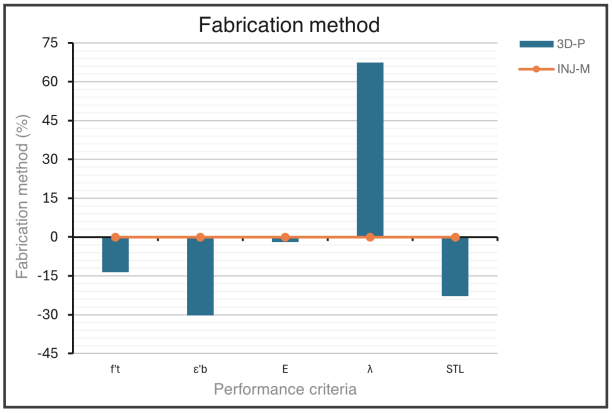


Figure 3.3.6. Effect of fabrication method of PLA/30wt.% recycled pinewood particles on the tensile strength, elongation at break, Young's modulus, thermal conductivity, and sound transmission loss (relative % compared to neat PLA). Adapted from "*The Thermal and Mechanical Behavior of Wood-PLA Composites Processed by Additive Manufacturing for Building Insulation*", by Bahar et al., 2023 [10].

3.4 Design tool

In the following section, the working principle behind the interactive design tool, guiding its user toward optimal-performing biocomposites based on various material properties and design options, is explained. The design tool, developed in Microsoft Excel, comprises four main phases (I-IV), addressing the effect of un(treated) wood dust content, type/content of nanoparticle treatment, annealing time, and type of fabrication method respectively. Each phase consists of four steps (a-d), where a) provides an overview of the current performance of the biocomposite compared to various building materials based on a normalised score, b) provides insights into the performance criteria that can be influenced in the current phase based on absolute values, c) asks the user to assign weights to each of these performance criteria based on their need of performance improvement, and d) showcases the optimal design option based on these weights.

Phase I, illustrated in Figure 3.4.1, covers the effect of (un)treated wood dust content on the biocomposites' (initially neat PLA) tensile strength, elongation at break, Young's modulus, water absorption, and density. After assigning weights to each of these performance criteria and sustainability, covering the amount of waste wood that is used, a total of six performance score graphs are created, and illustrated in Figures A.256-A.261 of appendix A. These normalised graphs utilise the results from Figures A.227-A.236 to highlight the most optimal type and content of PLA filler for each performance criteria. As an example, as illustrated in Figure 3.4.1 (c), 40wt.% of untreated wood dust fibers would be the most optimal PLA filler from a Young's modulus perspective. As the final step of phase I, all six performance score graphs are combined into Figure 3.4.1 (d), and the overall most optimal PLA filler is highlighted, in this case 30wt.% thermally-treated wood dust. Before moving on, the biocomposites' performance is updated.

Phase II, illustrated in Figure 3.4.2, covers the effect of nanoparticle content on the biocomposites' tensile strength, elongation at break, Young's modulus, water vapour permeability, and UV-resistance, and follows the same steps as phase I, as illustrated in Figures A.262-A.268 of appendix A.

Phase III, illustrated in Figure 3.4.3, covers the effect of annealing time on the biocomposites' compressive strength, elongation at break, Young's modulus, water absorption, and thermal conductivity, and follows the same steps as the ones before.

Phase IV, illustrated in Figure 3.4.4, covers the effect of fabrication on the biocomposites' tensile strength, elongation at break, Young's modulus, thermal conductivity, and sound transmission loss.

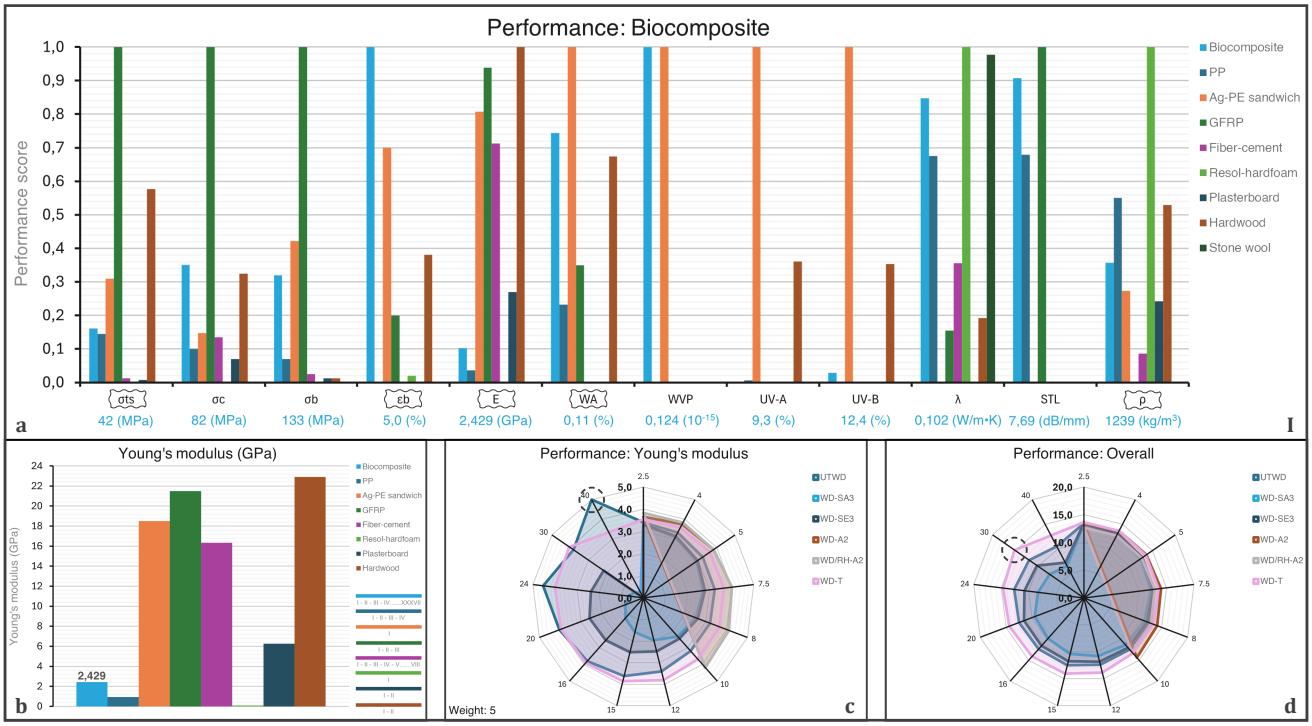


Figure 3.4.1. a) Performance overview biocomposite vs. building materials (tensile strength, compressive strength, bending strength, elongation at break, Young's modulus, water absorption, water vapour permeability, UV-A absorption, UV-B absorption, thermal conductivity, sound transmission loss, and density). Biocomposite (Adapted from [1]; [51]; [29]; [79]; [87]; [35]; [54]; [41]; [40]; [39]; [85]; [88]; [70]; [97]; [14]; [103]; [104]; [30]; [24]; [63]; [26]; [90]; [19]; [50]; [57]; [100]; [99]; [72]; [25]; [20]; [3]; [61]; [77]; [55]; [46]; [8]; [6]), polypropylene (Adapted from [2]; [68]; [65]; and Edupack, 2023), aluminium-polyethylene sandwich (Adapted from Edupack, 2023), glass fiber-reinforced polymer (Adapted from [86]; [12]; and Edupack, 2023), treated natural fiber-cement (Adapted from [105]; [92]; [98]; [59]; [62]; [64]; [84]; and [47]), resol-hardfoam (Adapted from Edupack, 2023), plasterboard (Adapted from [49]; and Edupack, 2023), treated hardwood (Adapted from [48]; and Edupack, 2023), and stone wool (Adapted from Edupack, 2023). **b)** Young's modulus overview. **c)** Performance score of various PLA fillers (UTWD {Adapted from [1]; [51]; [29]; [79]; [77]; and [55]}, WD-SA3 {Adapted from [1]}, WD-SE3 {Adapted from [1]}, WD-A2 {Adapted from [87]}, WD/RH-A2 {Adapted from [87]}, and WD-T {Adapted from [8]; [46]; and [55]}) on the Young's modulus. **d)** Weighted overall performance score of various PLA fillers.

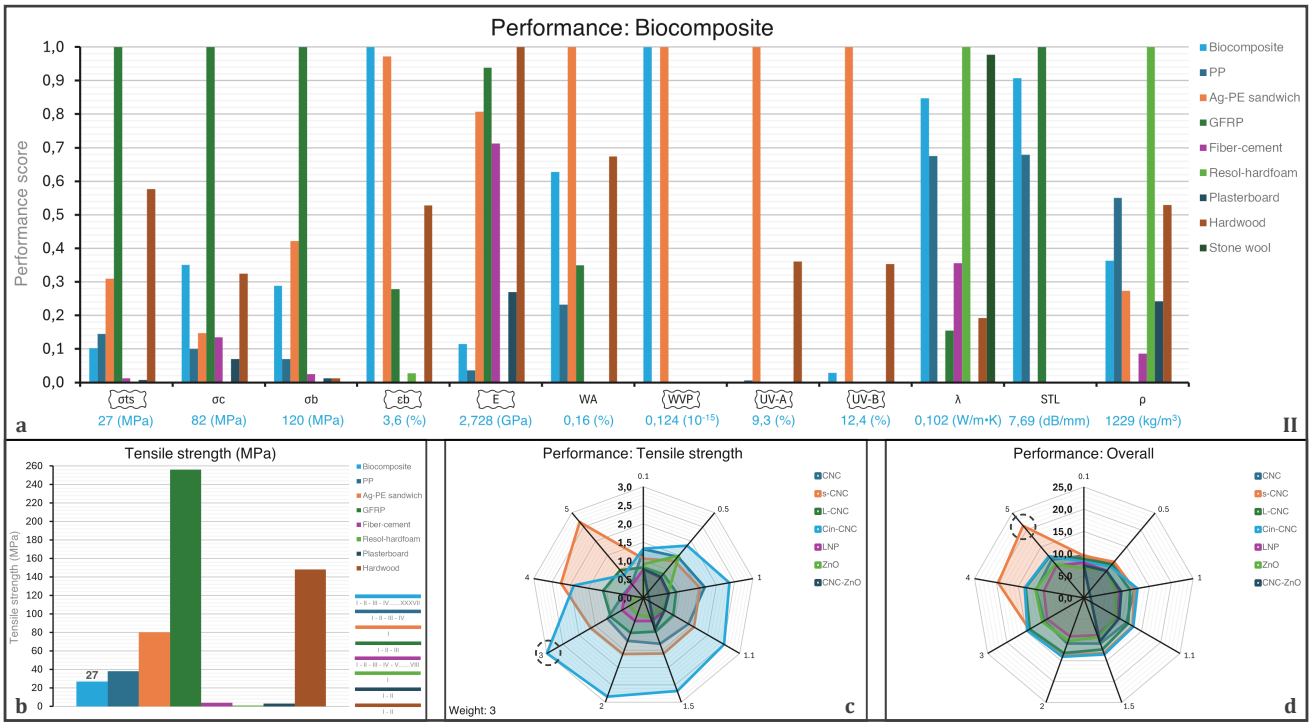


Figure 3.4.2. a) An updated (1) performance overview biocomposite vs. building materials (tensile strength, compressive strength, bending strength, elongation at break, Young's modulus, water absorption, water vapour permeability, UV-A absorption, UV-B absorption, thermal conductivity, sound transmission loss, and density). Biocomposite (Adapted from [1]; [51]; [29]; [79]; [87]; [35]; [54]; [41]; [40]; [39]; [85]; [88]; [70]; [97]; [14]; [103]; [104]; [30]; [24]; [63]; [26]; [90]; [19]; [50]; [57]; [100]; [99]; [72]; [25]; [20]; [3]; [61]; [77]; [55]; [46]; [8]; [6]), polypropylene (Adapted from [2]; [68]; [65]; and Edupack, 2023), aluminium-polyethylene sandwich (Adapted from Edupack, 2023), glass fiber-reinforced polymer (Adapted from [86]; [12]; and Edupack, 2023), treated natural fiber-cement (Adapted from [105]; [92]; [98]; [59]; [62]; [64]; [84]; and [47]), resol-hardfoam (Adapted from Edupack, 2023), plasterboard (Adapted from [49]; and Edupack, 2023), treated hardwood (Adapted from [48]; and Edupack, 2023), and stone wool (Adapted from Edupack, 2023). **b)** Updated (1) tensile strength overview. **c)** Performance score of various PLA nanoparticles (CNC {Adapted from [39]; [100]; [3]; [72]; [25]; [88]}, s-CNC {Adapted from [25]; [70]; [39]}, L-CNC {Adapted from [99]; [20]; [85]}, Cin-CNC {Adapted from [88]}, LNP {Adapted from [85]; [19]; [30]; [24]}, ZnO {Adapted from [70]; [104]; [26]; [90]; [50]}, and CNC-ZnO {Adapted from [70]}) on the tensile strength. **d)** Weighted overall performance score of various nanoparticles.

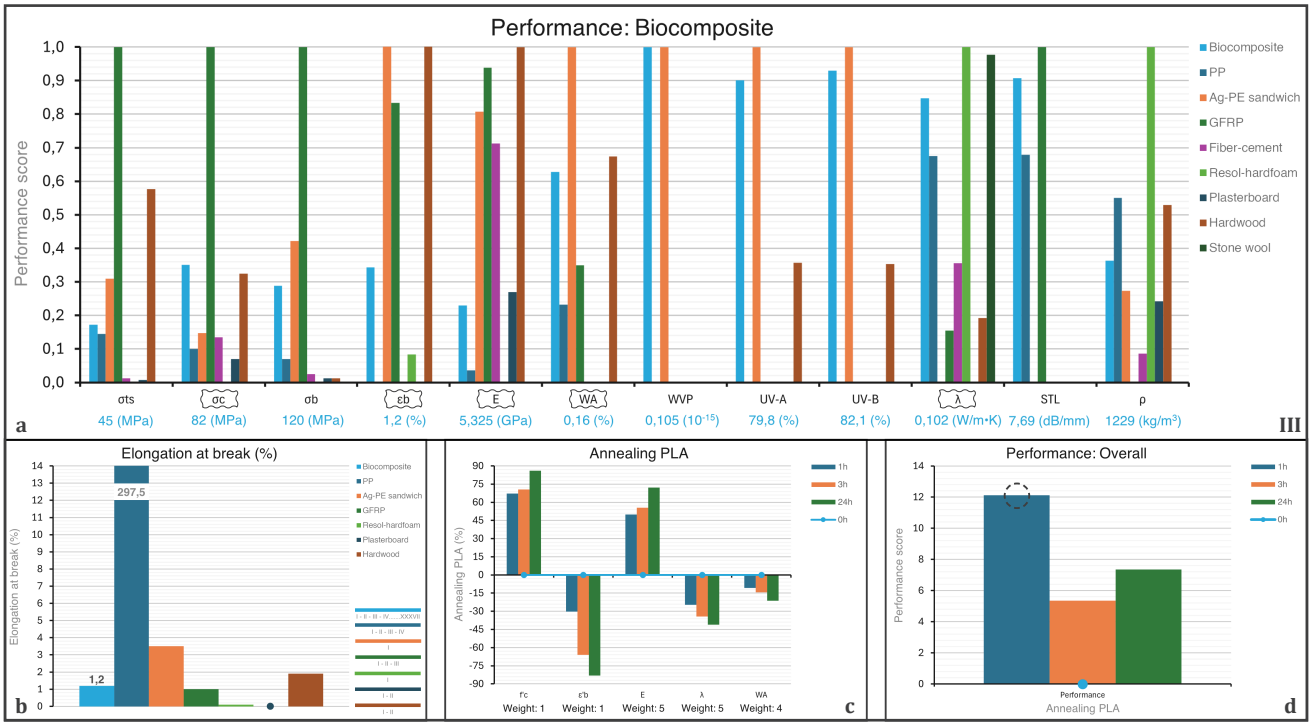


Figure 3.4.3. a) An updated (2) performance overview biocomposite vs. building materials (tensile strength, compressive strength, bending strength, elongation at break, Young's modulus, water absorption, water vapour permeability, UV-A absorption, UV-B absorption, thermal conductivity, sound transmission loss, and density). Biocomposite (Adapted from [1]; [51]; [29]; [79]; [87]; [35]; [54]; [41]; [40]; [39]; [85]; [88]; [70]; [97]; [14]; [103]; [104]; [30]; [24]; [63]; [26]; [90]; [19]; [50]; [57]; [100]; [99]; [72]; [25]; [20]; [3]; [61]; [77]; [55]; [46]; [8]; [6]), polypropylene (Adapted from [2]; [68]; [65]; and Edupack, 2023), aluminium-polyethylene sandwich (Adapted from Edupack, 2023), glass fiber-reinforced polymer (Adapted from [86]; [12]; and Edupack, 2023), treated natural fiber-cement (Adapted from [105]; [92]; [98]; [59]; [62]; [64]; [84]; and [47]), resol-hardfoam (Adapted from Edupack, 2023), plasterboard (Adapted from [49]; and Edupack, 2023), treated hardwood (Adapted from [48]; and Edupack, 2023), and stone wool (Adapted from Edupack, 2023). **b)** Updated (2) elongation at break overview. **c)** Performance score of various PLA annealing times (0h, 1h, 3h, and 24h {Adapted from [14]}) on the compressive strength, elongation at break, Young's modulus, thermal conductivity, and water absorption. **d)** Weighted overall performance score of various annealing times.

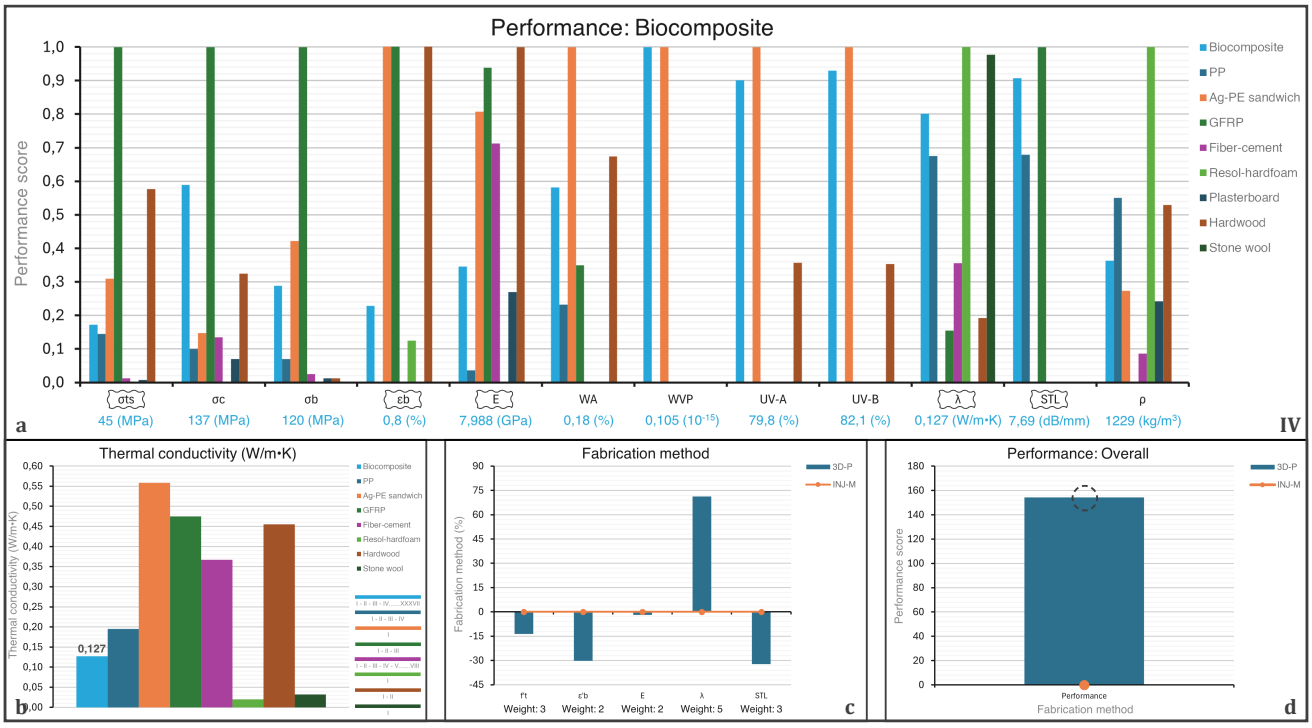


Figure 3.4.4. a) An updated (3) performance overview biocomposite vs. building materials (tensile strength, compressive strength, bending strength, elongation at break, Young's modulus, water absorption, water vapour permeability, UV-A absorption, UV-B absorption, thermal conductivity, sound transmission loss, and density). Biocomposite (Adapted from [1]; [51]; [29]; [79]; [87]; [35]; [54]; [41]; [40]; [39]; [85]; [88]; [70]; [97]; [14]; [103]; [104]; [30]; [24]; [63]; [26]; [90]; [19]; [50]; [57]; [100]; [99]; [72]; [25]; [20]; [3]; [61]; [77]; [55]; [46]; [8]; [6]), polypropylene (Adapted from [2]; [68]; [65]; and Edupack, 2023), aluminium-polyethylene sandwich (Adapted from Edupack, 2023), glass fiber-reinforced polymer (Adapted from [86]; [12]; and Edupack, 2023), treated natural fiber-cement (Adapted from [105]; [92]; [98]; [59]; [62]; [64]; [84]; and [47]), resol-hardfoam (Adapted from Edupack, 2023), plasterboard (Adapted from [49]; and Edupack, 2023), treated hardwood (Adapted from [48]; and Edupack, 2023), and stone wool (Adapted from Edupack, 2023). **b)** Updated (1) thermal conductivity overview. **c)** Performance score of two fabrication methods (3D-Printing and injection molding {Adapted from [10]}) on the tensile strength, elongation at break, Young's modulus, thermal conductivity, and sound transmission loss. **d)** Weighted overall performance score of two fabrication methods.

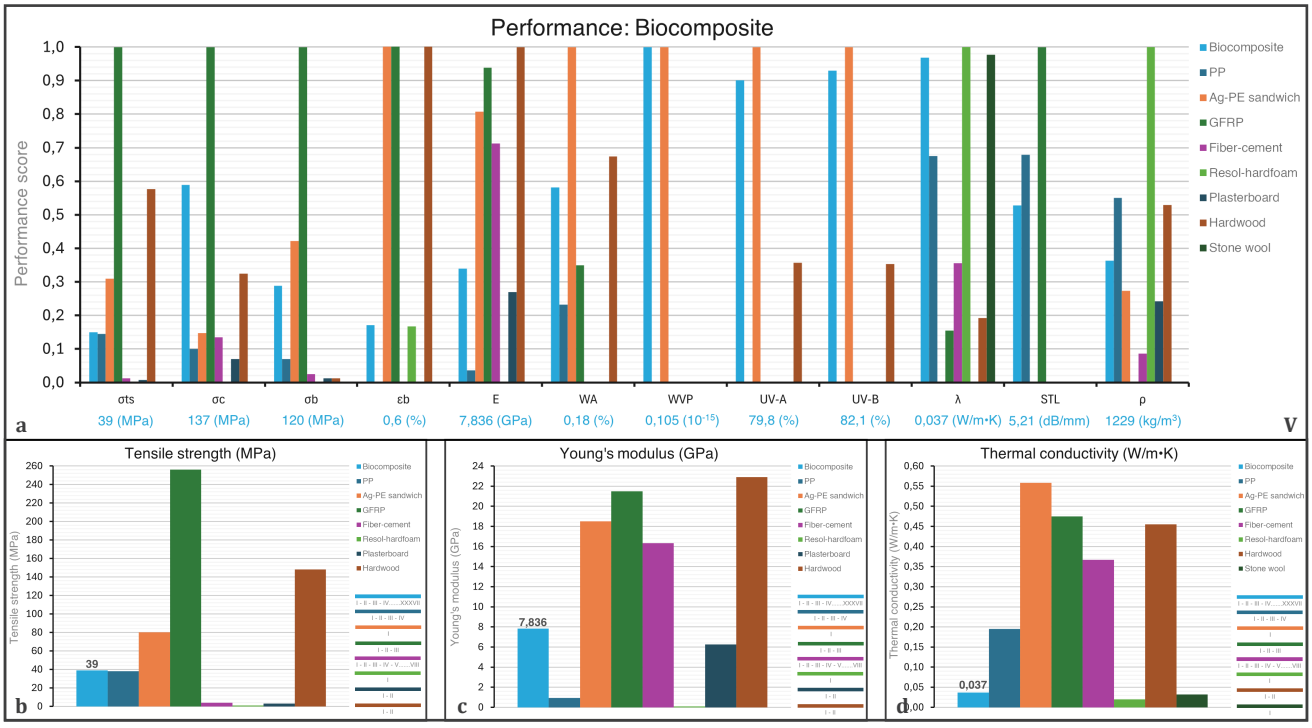


Figure 3.4.5. a) Final performance overview biocomposite vs. building materials (tensile strength, compressive strength, bending strength, elongation at break, Young's modulus, water absorption, water vapour permeability, UV-A absorption, UV-B absorption, thermal conductivity, sound transmission loss, and density). Biocomposite (Adapted from [1]; [51]; [29]; [79]; [87]; [35]; [54]; [41]; [40]; [39]; [85]; [88]; [70]; [97]; [14]; [103]; [104]; [30]; [24]; [63]; [26]; [90]; [19]; [50]; [57]; [100]; [99]; [72]; [25]; [20]; [3]; [61]; [77]; [55]; [46]; [8]; [6]), polypropylene (Adapted from [2]; [68]; [65]; and Edupack, 2023), aluminium-polyethylene sandwich (Adapted from Edupack, 2023), glass fiber-reinforced polymer (Adapted from [86]; [12]; and Edupack, 2023), treated natural fiber-cement (Adapted from [105]; [92]; [98]; [59]; [62]; [64]; [84]; and [47]), resol-hardfoam (Adapted from Edupack, 2023), plasterboard (Adapted from [49]; and Edupack, 2023), treated hardwood (Adapted from [48]; and Edupack, 2023), and stone wool (Adapted from Edupack, 2023). **b)** Final tensile strength overview. **c)** Final Young's modulus overview. **d)** Final thermal conductivity overview.

4

Facade design

As aforementioned, this thesis introduces a novel performance-driven design exploration framework integrating Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN), which strives to advance the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, and usability, to facilitate more efficient decision-making in the early design stage, potentially leading to better architectural and sustainable solutions.

As the foundation of the proposed AI-driven facade design framework, this chapter delves into the creation of a parametric model in Rhino 3D and Grasshopper, which focuses on balancing design variables with varying influence on geometry - essential to demonstrating the framework' design optimisation capabilities - rather than creating the most optimal facade design. Additionally, this chapter delves into the performance simulations conducted to generate representative labeled datasets for training the KAN models, focusing on balancing varying levels of performance approximation complexity - essential to demonstrating KANs efficiency in approximating non-linear functions as complexity increases, offering a comparison against MLPs.

Although any type of facade system or performance criteria could have been considered, this thesis focuses on an aluminium-based biocomposite curtain wall facade fragment due to its modularity, and total material use, solar heat gains, and sound pressure level performance simulations due to their balanced range of varying complexities and societal relevance in affecting sustainability, energy demands, and urban noise pollution, respectively.

4.1 Parametric model

Aiming to create a vast design space of 27,000 biocomposite facade design alternatives, while ensuring the SOM will be able to reflect the design space correctly, given the curse of dimensionality, nine distinct and carefully selected geometry-related design variables, from a set of nineteen options, are considered and given a specific range and interval. Before addressing each of them individually, it is worth mentioning that the facade, as illustrated in Figure 4.1.1, is six meters wide, four meters high, oriented south, positioned on the ground floor, divided into four vertical sections, with the top and bottom closed modules each being 0.75 meters in height, and all single modules containing windows.

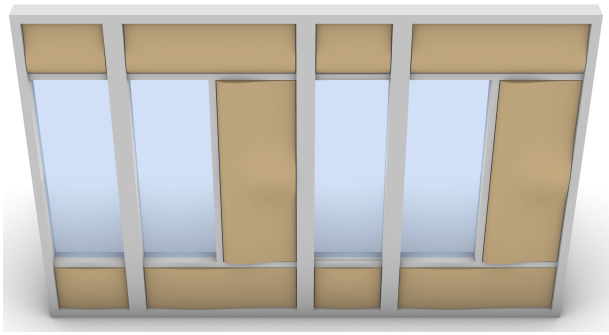


Figure 4.1.1. Standard facade design configuration without design variables.

As aforementioned, the facade is divided into four vertical sections, creating three control points that form design variables, Y1, Y2, and Y3. These points are able to move either one meter outward, inward, or remain stationary, in increments of 0.5 meters, allowing for a wide range of possible facade design configurations, as illustrated in Figures 4.1.2-4.1.5.

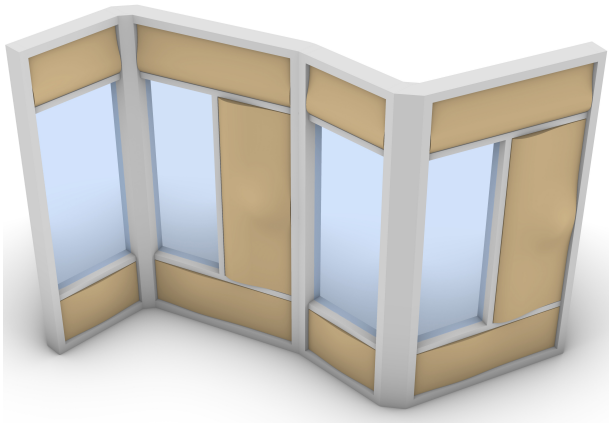


Figure 4.1.2. Facade design configuration based on changes in Y1, Y2, and Y3.

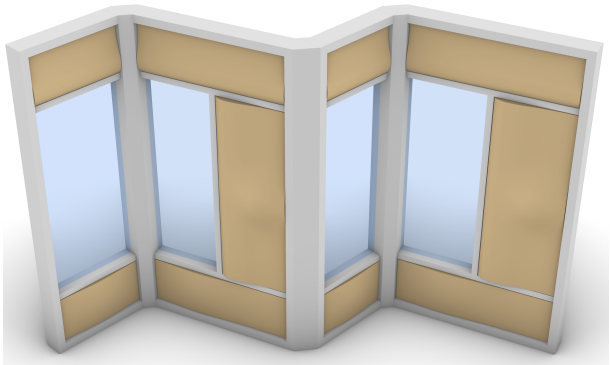


Figure 4.1.3. Facade design configuration based on changes in Y1, Y2, and Y3.

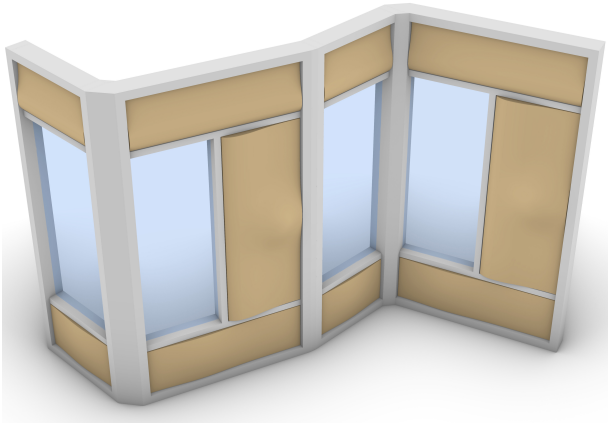


Figure 4.1.4. Facade design configuration based on changes in Y1, Y2, and Y3.

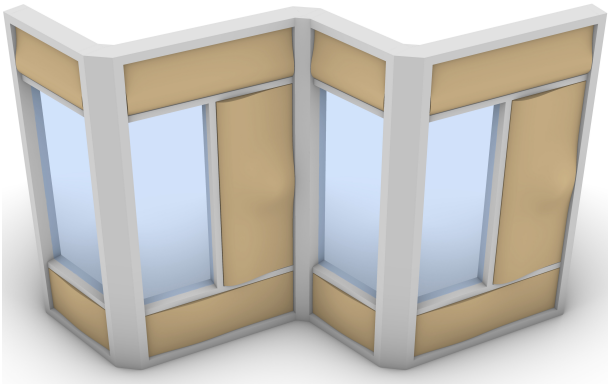


Figure 4.1.5. Facade design configuration based on changes in Y1, Y2, and Y3.

The fourth design variable, X3, influences the same control point as Y3, but adjusts its movement in a perpendicular direction, either with 0.5 meters to the left or 0.5 meters to the right, adding another layer of design variation, as shown in Figure 4.1.6.

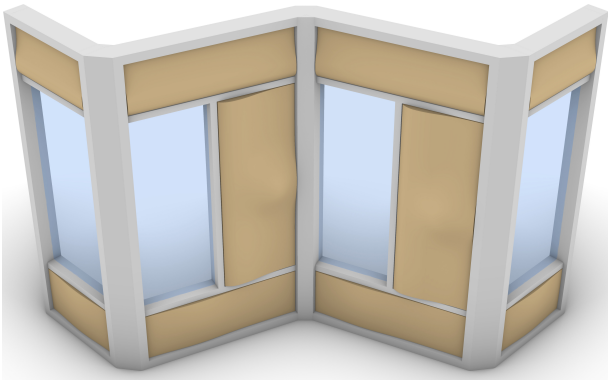


Figure 4.1.6. Facade design configuration based on changes in variable X3.

The fifth design variable, W2, determines the positioning of the window in the right-hand side double module, allowing for placement either on the left or the right. The sixth and seventh design variables, TS1 and SCD1, control the curvature of the biocomposite facade panels on the left-hand side. While TS1 influences the top panel of the single module by moving material outward horizontally, SCD1 influences the double module by vertical adjustment. Both variables can tune between 0.05, 0.10, or 0.15 meters, creating shading for different sun angles. The eighth variable, BS2, controls the curvature of the bottom panel of the right-hand single module by attracting material outward from its midpoint. It can extend by 0.05 or 0.15 meters, primarily for scattering acoustic rays away from the street. The nineth variable, SPD1, has the same 0.15-meter outward curvature as BS2, and also affects acoustics, but moves vertically on the left-hand double module, able to be positioned at either 3/4, 1/2, or 1/4 of the facade panel's height. The design variables W2, TS1, SCD1, BS2, and SPD1, as shown in Figure 4.1.7, are considered significantly less-influential on geometry, therefore guiding the design exploration's fine-tuning phase later on, together with X3.

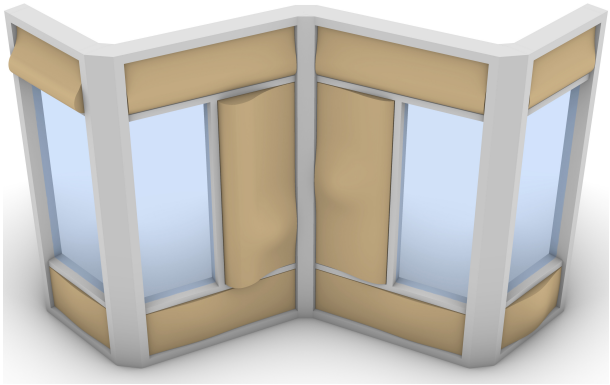


Figure 4.1.7. Facade design configuration based on changes in other variables.

To ensure realistic performance simulations, the facade is assigned actual material properties in Rhino 3D. While using predefined settings for most materials, the biocomposite facade panels required custom settings, followed from the best-performing compound identified by the design tool, consisting of 30wt.% thermally-treated wood dust fibers combined with 1h annealed polylactic acid (PLA), treated with 5wt.% surfactant-modified cellulose nanocrystals (s-CNC), using a phosphate ester of ethoxylated nonylphenol in a 1:4vo.% ratio, and fabricated through 3D-printing. Additionally, to offer insights into how the facade is constructed, facade sections were made, as illustrated in Figure 4.1.8.

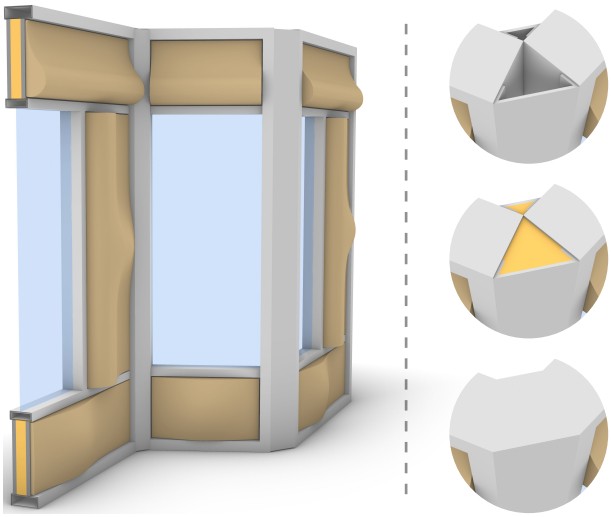


Figure 4.1.8. Biocomposite facade design close-ups for insight into construction.

4.2 Performance simulations

As aforementioned, performance simulations for total material use, solar heat gains, and sound pressure level are chosen due to their balanced range of varying complexities and societal relevance in affecting sustainability, energy demands, and urban noise pollution, respectively. The total material use is simply calculated using the volume component in Grasshopper, determining the m^3 of biocomposite facade material, directly influenced by design variables TS1, SCD1, and BS2. Although the volume of other materials within the facade are also affected by design variables Y1, Y2, and Y3, influencing the width of the vertical facade sections, they are excluded from the total material use calculation to specifically focus on the relationship between biocomposite material usage and its impact on both solar heat gains, providing shading at different sun angles, as well as sound pressure level, scattering acoustic rays away from the street. Different from the total material use calculation, both solar heat gains and sound pressure level calculations required an experimental setup in Grasshopper, as illustrated in Figure 4.2.1, featuring a wide street with a small studio containing the biocomposite facade.

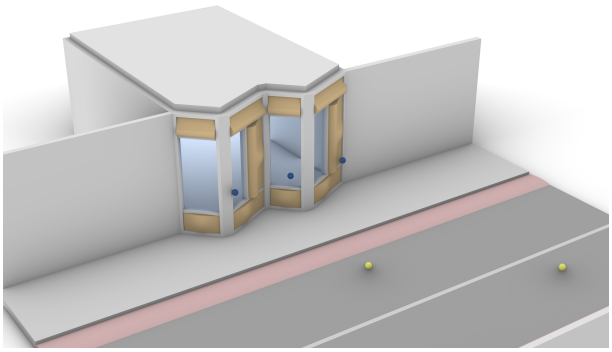


Figure 4.2.1. Experimental setup for solar/acoustic performance simulations.

Moving on with the solar heat gains calculation, the second most complex, and conducted using Ladybug in Grasshopper. The simulation consists of calculating kWh/m^2 for each of the five surfaces inside the studio, by analysing both direct and diffuse radiation based on Dutch weather conditions over a full year, as shown in Figures 4.2.2-4.2.3, followed by dividing the total kWh by the total surface area, creating an average solar heat gains in kWh/m^2.

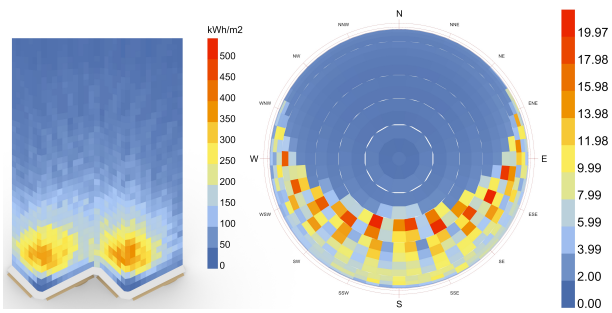


Figure 4.2.2. Solar heat gains performance simulation using Ladybug in GH.

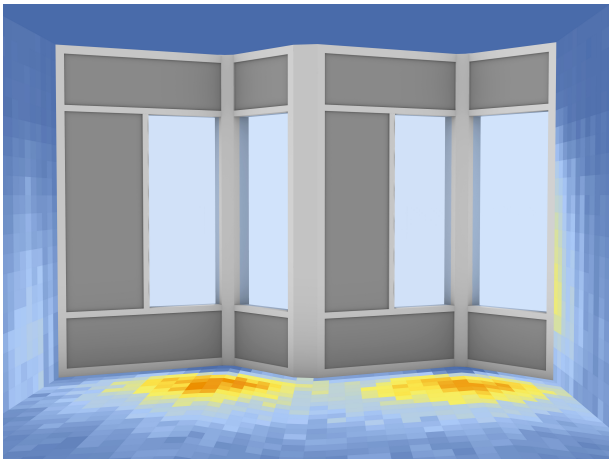


Figure 4.2.3. Solar heat gains performance simulation using Ladybug in GH.

The final and most complex simulation is the sound pressure level (SPL) calculation, conducted using Pachyderm Acoustics in Grasshopper. The simulation consists of three sound sources at 0.5 meters height, representing cars, emitting ranging sound levels from 60-75dB across octave bands, and three sound receivers in front of the facade at 1.6 meters height, capturing direct and reflected sound waves. The acoustic simulation is based on image sourcing and raycasting, sending one million rays from each source towards the facade in an omni-directional way, being either absorbed or reflected, as shown in Figures 4.2.4 and 4.2.5. During simulation, each absorber calculates the average SPL, based on all three sources. These values are then used to calculate the facade's overall sound pressure level (dB).

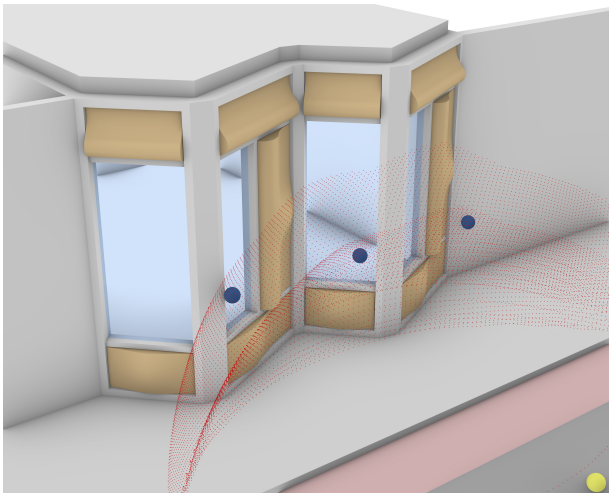


Figure 4.2.4. Sound Pressure Level performance simulation using Pachyderm.

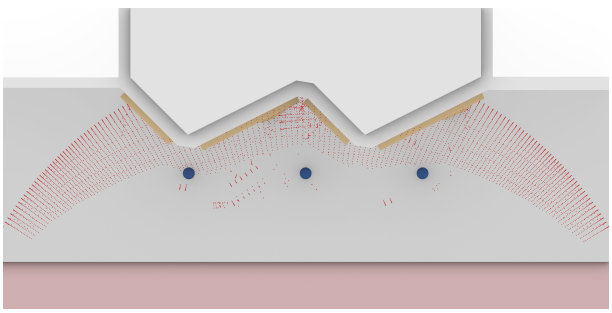


Figure 4.2.5. Sound Pressure Level performance simulation using Pachyderm.

5

Self-Organising Map

As the second phase of the proposed AI-driven facade design framework, this chapter delves into the process of training a Self-Organising Map (SOM), which uses all possible geometry-related vectors of design variables Y1, Y2, and Y3, identified as the most influential on the facade's geometry, therefore leading the design exploration's orientation phase later on, to cluster 125 facade design variants onto a two-dimensional network of nodes based on their geometric characteristics, enabling the designer to navigate the entire design space according to geometry typology. The chapter starts by examining the training process of the unsupervised deep topological neural network, in other words, the self-organising map, covering all aspects from preparing the training vectors and discussing challenges encountered to initialising the SOM model, hyperparameter tuning, and examining clustering performance, followed by delving into the final clustering results and their integration back into Rhino 3D and Grasshopper, and concluding by highlighting the stratified sampling method utilised to extract representative labeled datasets for training the Kolmogorov-Arnold Network (KAN) models.

5.1 Training process

Foundational to training the Self-Organising Map (SOM) effectively, is preparing the training vectors, focusing solely on those affecting geometry, as including non-geometry related vectors disrupts the clustering process. As aforementioned, these training vectors are based on all possible combinations of values within the ranges of design variables Y1, Y2, and Y3. In other words, the 125 three-dimensional arrays of values used to train the SOM, are the exact same values that can be used to generate their geometries with the parametric model. To ensure that each design variable within these vectors contributes equally to the Euclidean distance calculations during training, they are normalised (0-1).

Consequently, the number of values within each design variable range plays a significant role in SOM clustering, as it directly affects the step size which influences ED calculations during training, thereby determining the amount of impact each design variable has on clustering. When the number of values within these ranges is not balanced across design variables, normalisation can skew their original relative influence on geometry, causing some variables to disproportionately affect the clustering process. As an example, the normalisation process of all three original Y-variable ranges, from [-1, -0.5 0, 0.5, 1] to [0, 0.25, 0.5, 0.75, 1], reduced the step size from 0.5 meter, reflecting actual increments, to 0.25 meter after normalisation. Because all the Y-variables share the same number of values (5) in their range, their relative influence on geometry remained preserved. However, issues arose when design variable X3 - which was originally intended for use in the design exploration's orientation phase as well, creating four-dimensional SOM training vectors - was given only two values in its range. Originally, design variable X3 spanned from -0.5 to 0.5 meter, accurately reflecting its greater influence on geometry compared to the Y-variables, indicated with a step size of 1 meter compared to 0.5 meter. After the normalisation process, the value range of X3 became [0, 1], preserving their original 1-meter step size, making it four times as influential on clustering as the Y-variables, twice as much as intended. Aiming to overcome this issue, it was attempted to train the SOM without normalisation, but the unbalanced absolute values negatively affected the clustering performance. Therefore, it is crucial to take the effect of normalisation into account early on in the design process by carefully setting up the parametric model, ensuring design variables' original relative geometric influence remain preserved.

After preparing the 125 three-dimensional normalised training vectors, such as [0.25, 1, 0.75], [0.5, 0, 0.25], and [1, 0.25, 0.5], a pairwise distance plot was created to assess their diversity. As illustrated in Figure 5.1.1, the range of Euclidean distances between all pairs of training vectors peaks around 0.55, 0.65, 0.75, and 0.95, and ranges from 0.25 to 1.70, indicating a reasonable spread of vectors within the design space. This is crucial as it directly affects SOM's ability to effectively distinguish between design variants, based on their vectors.

The second step in training the Self-Organising Map (SOM) involved initialising the model in Python 3.9.13, using MiniSOM 2.3.3. Aiming to balance the number of vectors to be clustered and the SOM-network size determined by the number

of nodes in columns and rows, ensuring each design variant has sufficient space to adapt to a distinct best matching unit (BMU) on the SOM, various network sizes were examined. After considering grids of 12x12, 15x15, and 20x20, ensuring full 125 node activations with as small as possible grid to enhance design exploration later on, a 15x15 grid was selected. Furthermore, a hexagonal grid was chosen, as they generally offer better topological preservation than rectangular ones due to a smaller difference in the number of adjacent nodes between the edges and the center (Licen et al., 2023).

Additionally, the SOM was initialised with an ordering and tuning phase, both using decreasing neighbourhood radius and learning rate over epochs to allow the SOM to first capture the general topology and later fine-tune the feature map by focusing on more localised adjustments.

Finally, to assess and optimise SOM's clustering performance during training, a multi-objective optimisation approach was used, where both quantisation error (QE) - which relates to the average distance between each training vector and its best matching unit, influencing input space representation accuracy - and topographic error (TE) - which relates to the number of training vectors for which the first and second best matching unit are not adjacent on the SOM grid, affecting topological preservation - are combined into a weighted sum and minimised during hyperparameter tuning.

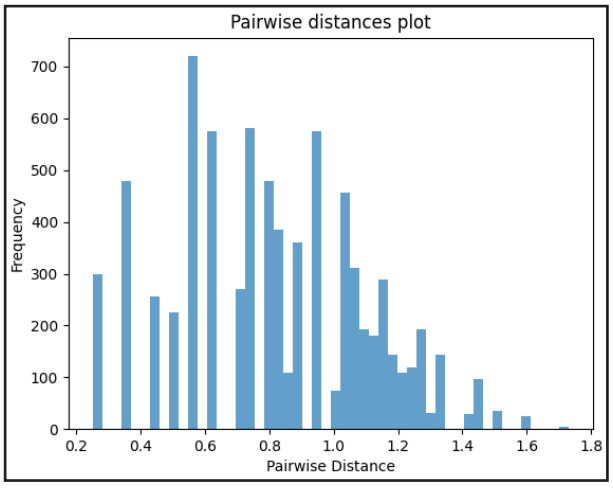


Figure 5.1.1. Pairwise distances plot to assess the diversity of training vectors.

The third step in training the Self-Organising Map (SOM), the hyperparameter tuning process, involved iterating through different values for initial and final neighbourhood radius and learning rate, and batch sizes, to find the least loss model. Aiming to approach SOM's optimal clustering performance, the tuning process was carried out in two distinct phases: rough-optimisation and fine-optimisation, with each of them having a fixed number of ordering and tuning epochs set at 900 and 100, respectively. During rough-optimisation, a total of 1250 iterations were conducted, with the initial neighbourhood radius ranging from 0.5 to 2.5 with increments of 0.5, and 1.0 yielding the best results, the final neighbourhood radius varying from 0.1 to 0.5 in steps of 0.1, and 0.2 providing the lowest losses, the initial learning rate also spanning from 0.1 to 0.5 with intervals of 0.1, and 0.4 producing the most favorable results, the final learning rate ranging from 0.01 to 0.05 with increments of 0.01, and 0.02 being the best-performing value, and fifinally the batch size either set to 5 or 10, with 5 yielding most optimum results. During fine-optimisation, another 625 iterations were conducted to fine-tune the model, with the initial neighbourhood radius ranging from 0.9 to 1.1 with increments of 0.05, and 0.95 being the best-performing value, the final neighbourhood radius varying from 0.19 to 0.21 in steps of 0.005, and 0.195 yielding the best results, the initial learning rate spanning from 0.36 to 0.44 with intervals of 0.02, and 0.42 providing the lowest losses, the final learning rate ranging from 0.016 to 0.024 with steps of 0.002, and 0.020 yielding optimum results, and the batch size being fixed at 5. These optimal hyperparameter settings resulted in a quantisation error (QE) of 0.0771 and a topographic error (TE) of 0 at 691 epochs, both of which are excellent results, as illustrated in Figure 5.1.2, which shows the training error losses over epochs of the best-performing SOM model.

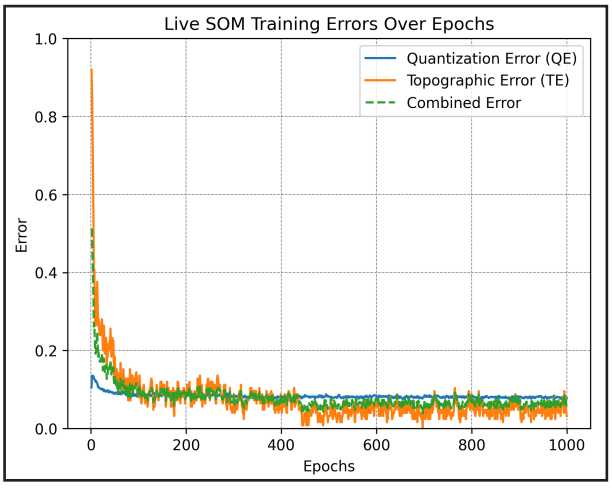


Figure 5.1.2. Training errors over epochs plot to assess SOM's performance.

After optimising the SOM's clustering performance through a hyperparameter tuning process, several plots were created to provide visual insights into the clustering quality. As illustrated in Figure 5.1.3, the quantisation error (QE) plot provides rough insights into the distribution of facade designs across the feature map, with the x and y axis relating to the nodes in columns and rows, respectively. As can be seen, the quantisation error plot shows a considerably uniform distribution of input space represen-

tation errors across the nodes, ranging from 0 to 0.3 with an average error of 0.0771, indicating excellent quantisation accuracy across the entire map.

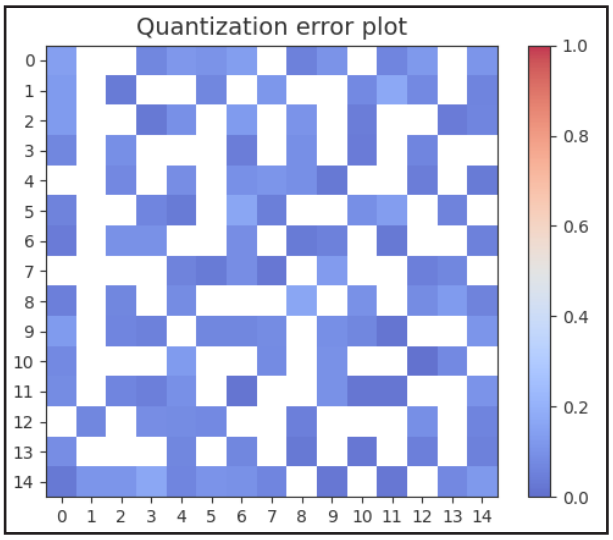


Figure 5.1.3. Quantization error (QE) plot showing input space representation.

As illustrated in Figure 5.1.4, the U-matrix plot provides insights into the Euclidean distances between neighbouring node vectors, aiming to identify cluster boundaries, with red colors representing more different clusters and blue colors showing closer relationships. As can be seen, the U-matrix plot displays a gradient of colours, with around 15% light blue nodes within a range of 0.10 to 0.15, 45% light red nodes within a range of 0.15 to 0.20, 35% red nodes within a range of 0.20 to 0.25, and 5% dark red nodes within a range of 0.25 to 0.30, indicating clear cluster boundaries across the feature map.

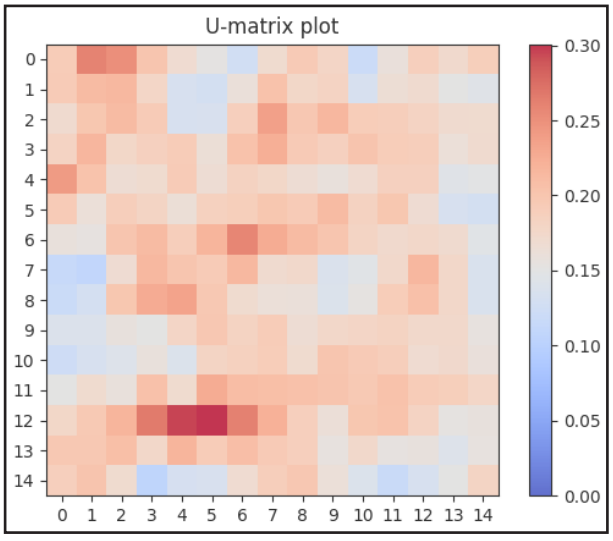


Figure 5.1.4. U-matrix plot showing the cluster boundaries across the map.

As illustrated in Figure 5.1.5, the variable distribution plot provides clear insights into the spread of facade designs across the feature map, by plotting the distribution of normalised training values for each Y-variable, with red nodes representing high values and blue nodes illustrating low values. As can be seen, the variable distribution plot allows for quick assessment of the SOM's clustering performance due to its colour-based distribution, showing smooth and subtle colour transitions locally, and more significant differences between distant nodes, indicating meaningful clustering of facade design alternatives, critical for design exploration.

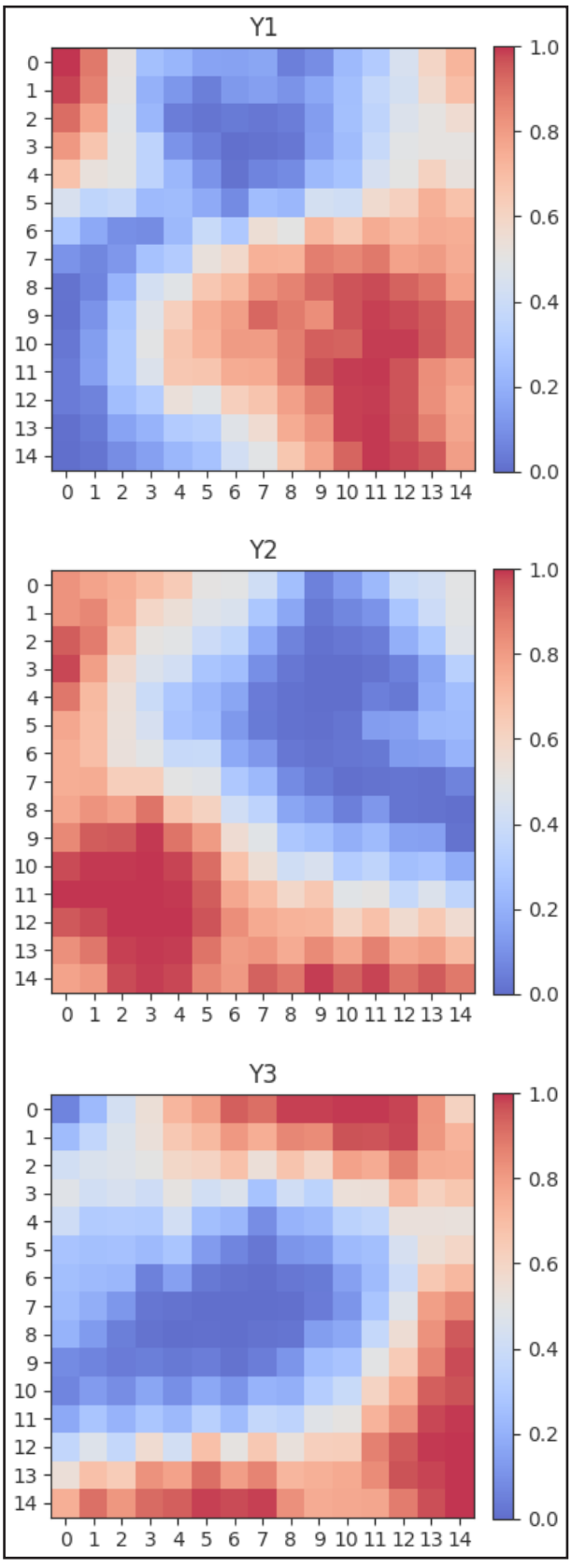


Figure 5.1.5. Variable distribution plot showing the spread of facade designs.

5.2 Results

Following the training of the Self-Organising Map (SOM), the final clustering results are integrated back into Grasshopper by extracting the geometry-related node design vectors along with their corresponding coordinates on the SOM grid, creating a hexagonal 15x15 grid in Grasshopper, baking the facade design geometries based on the extracted node design vectors using the parametric model, and positioning them on the SOM grid according to their coordinates, as illustrated in Figure 5.2.1.

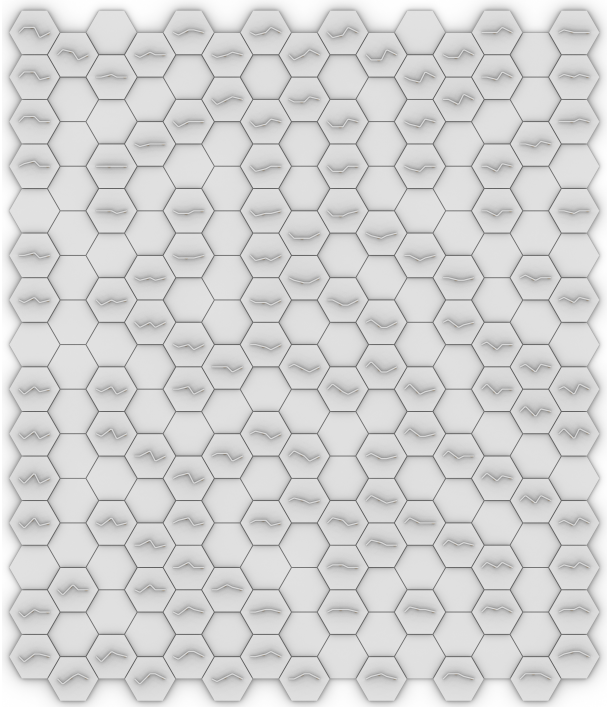


Figure 5.2.1. Top view facade design clustering on 15x15 hexagonal SOM grid.

As a result, the foundation of the design exploration's orientation phase is established by effectively clustering the 125 most geometrically influential facade design variants on the SOM based on their vectors, enabling the designer to navigate the entire design space according to geometry typology.

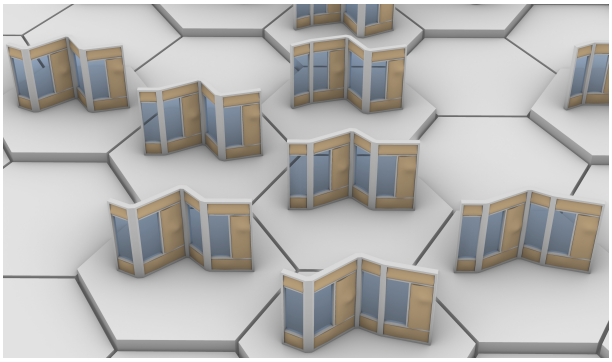


Figure 5.2.3. Zoomed in perspective view facade design clustering on SOM grid.

As the final step, a total of 6,750 nine-dimensional geometry-related vectors were extracted through a stratified sampling method, creating a subset representing 25% of the entire dataset, to train the Kolmogorov-Arnold Network (KAN) models using the vectors as x-input training data. Stratification was carefully executed to ensure the subset was as small as possible while maintaining maximum representativeness of the entire dataset, as the size of the training data directly affects computational demands, with larger datasets leading to requiring more time-consuming performance simulations, and as the representativeness of the training data influences the surrogate model's ability to generalise well to unseen data. As a result, the stratified sampling method first involved randomly selecting a small number of vectors, followed by obtaining their y-input training data through performance simulations, and evaluating the prediction accuracy of all three KAN models, one for each metric. This process was repeated until the R^2-score exceeded 0.80 for all models, which happened at 1200 training vectors. These KAN models were used to generate interpretability plots, showing the percentual effect of each design variable on the performance predictions, and the average influence percentages were then used to ensure stratification was most representative of changes in influential variables.

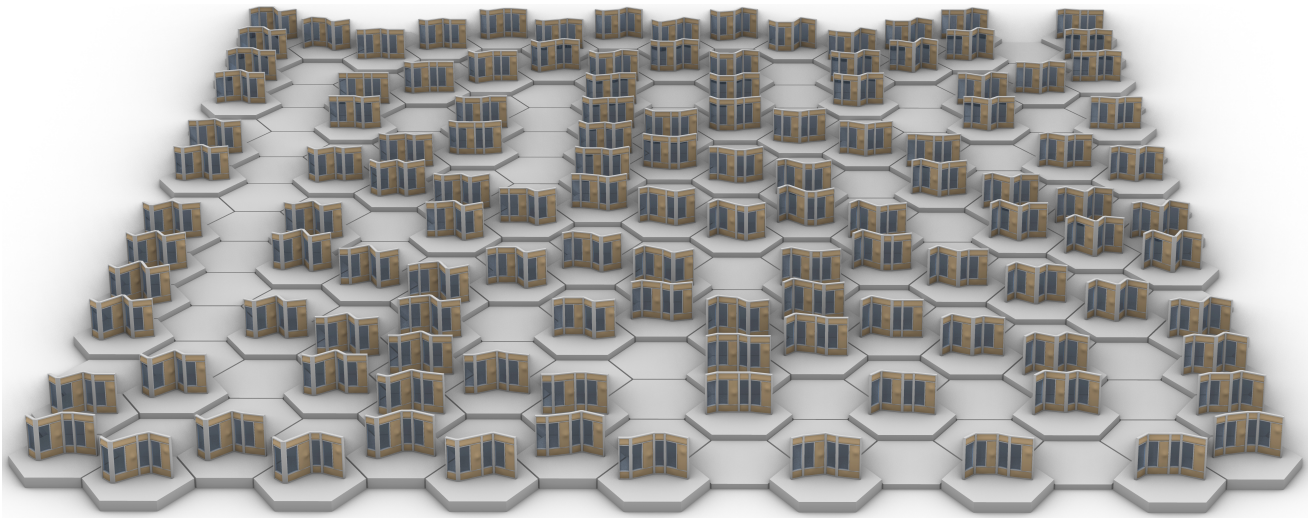


Figure 5.2.2. Perspective view of final biocomposite facade design clustering in Grasshopper on 15x15 hexagonal grid using Self-Organising Map (SOM) clustering method.

6

Kolmogorov-Arnold Network

As the most fundamental phase of the proposed AI-driven facade design framework, this chapter delves into the process of training three Kolmogorov-Arnold Networks (KAN), one for each objective, to approximate the performance of all design alternatives with maximum accuracy, and to allow for interactive human-AI feedback loops, by identifying how each individual input variable affects the predicted output. The chapter starts by examining the training process, covering all aspects from preparing the training vectors, to initialising the KAN models and hyperparameter tuning, followed by delving into the final performance approximation results and the creation of interpretability plots, and concluding by systematically comparing KANs with MLPs, providing insight into KAN's efficiency in approximating non-linear functions as both training vectors and complexity increases, relative to MLP.

6.1 Training process

As the first step in training the Kolmogorov-Arnold Network (KAN) models, after extracting the 6,750 nine-dimensional x-input training vectors during stratification, the corresponding y-inputs are obtained by conducting performance simulations based on these nine-dimensional vectors for material use, solar heat gains, and sound pressure level. Aiming to manage time efficienty, the process was automated using a combination of Anemone, a Grasshopper extension looping through all 6,750 geometry-related vectors and activating performance simulations after each change in design, and C# scripts saving the performance values automatically to csv format. In total, the process required around 335 hours of processing time on an Intel Core i7 9th Generation CPU and Nvidia GeForce RTX2070 GPU, with around 11 hours dedicated to material use, 45 hours to solar heat gains, and 279 hours to sound pressure level simulations, highlighting their varying complexities. The performance values were calculated with six-decimal accuracy, with ranges for material use from 0.50 to 0.83 m^3, solar heat gains from 32.47 to 41.07 kWh/m^2, and sound pressure level from 54.96 to 56.52 dB, creating 12-dimensional vectors such as: [0.5, -0.5, 1.0, 0.5, 1.0, 0.15, 0.15, 0.05, 2.0, 0.683376, 34.210462, 55.370307], corresponding to: Y1, Y2, Y3, X3, W2, TS1, SCD1, BS2, SPD1, total material use, solar heat gains, and sound pressure level, respectively. To ensure the KAN models are trained effectively, the 9-dimensional x-input training vectors are standardised (z-score), ensuring they have a mean close to zero and a variance close to one, optimising KAN's learning stability and convergence by mitigating the impact of varying value range scales among design variables. As an example, standardisation of the aforementioned twelve-dimensional vector transformed it into: [0.71, -0.72, 1.43, 1.02, 0.99, 1.24, 1.24, -1.00, 1.23, 0.683376, 34.210462, 55.370307]. Following this, the 6,750 twelve-dimensional standardised training vectors were converted into three sets of 6,750 ten-dimensional vectors, each corresponding to one of the three KAN models to be trained for each performance metric. Next, each ten-dimensional dataset was split into a 60% training set, 20% validation set, and 20% testing set, using the same random seed (42) for all three. This was followed by splitting each set into x-input (9) and y-input (1) vectors and transforming them into torch tensors to enable efficient back-propagation in PyTorch.

The next step in training the Kolmogorov-Arnold Network (KAN) models involved initialising the models in Python 3.9.13, using pyKAN 0.2.6 and its prerequisite set of libraries, including numpy 1.24.4, scikitlearn 1.1.3, setuptools 65.5.0, sympy 1.11.1, torch 2.2.2, and tqdm 4.66.2. Aiming to optimise computational efficiency, CUDA 12.4 was integrated for GPU acceleration, enabling parallel computing. Next in initialising the KAN models, the so-called Limited memory Broyden Fletcher Goldfarb Shanno (LBFGS) optimiser was chosen to minimise the Mean Squared Error (MSE) between model predictions and ground truth during training, by tweaking control points of all B-spline activation functions through back-propagation. Different from the Adam optimiser, typically used in Multi-Layer Perceptrons (MLP) because of its effective combination of adaptive learning rates and momentum - relating to speeding up the process of finding the minimum of a function, also known as gradient descent, by adding a part of the previous weight update to the current one - leading to fast convergence and stable weight updates, particularly effective for large models like MLP, the LBFGS optimiser uses so-called

second-order approximations - relating to utilising the 2nd derivative of a function to navigate the loss landscape - allowing for precise convergence without requiring adaptive learning rates, particularly effective for small high-accuracy models like KAN.

Moving on, the KAN models were initialised with singularity avoiding and update grid set to true, preventing extreme values to destabilise training and enhance accuracy by dynamically adjusting the B-spline activation functions' grid points during training, respectively. Additionally, a learning rate of one was used, as recommended by Liu et al. (2024), as well as a lambda of zero, a L1-regularisation term of one, a lambda entropy of two, a lambda coefficient of zero, a beta of one hundred, an output threshold of one thousand, and a k-value of three, affecting KAN's speed and stability of convergence during training, efficiency of regularisation to avoid overfitting by penalising the model from becoming too complex or adapting to noise to improve its generalisation capabilities, transparency of connections between neurons for visualisation, ability to prevent extreme values from skewing predictions, and ability to represent complex patterns smoothly by using polynomial cubic splines (k=3), balancing accuracy and interpretability.

As the final step in initialising the KAN models, each model architecture was set up with nine inputs, two hidden layers, and one output, with the number of nodes in the hidden layers tuned according to their varying performance approximation complexities. As a result, the KAN models for both material use and solar heat gains have 4-2 nodes in their hidden layers, while the KAN model employed for sound pressure level comprises double the number of nodes in its hidden layers. On top of that, each model has a fixed number of epochs set at 100, paired with an early stopping mechanism that stops training after 10 epochs without improvement, keeping the best-performing model state.

The third step in training the Kolmogorov-Arnold Network (KAN) models, the hyperparameter tuning process, aims to optimise performance approximation accuracy by iterating through different values for grid size, which affects the granularity of the B-spline activation functions, indicating an exceptionally straightforward training process. This is unlike the training process of Multi-Layer Perceptrons (MLP), which typically require extensive iterations through multiple hyperparameters such as learning rate, batch size, and dropout probability, often leading to over fifteen times more iterations than are needed for KAN models, making them significantly more intuitive to train. In addition to requiring fewer iterations, KAN requires around ten times less number of epochs as well, due to their faster convergence - achieving lower losses more quickly during training - making them considerably faster to train, despite their inability to leverage batch computation. During hyperparameter tuning, a total of eight iterations were conducted for each KAN model, with the grid size ranging from 3-10 with increments of 1, and 5 yielding the lowest Mean Squared Errors (MSE) across all KAN models, assessed on their validation set.

6.2 Results

Following the training of the Kolmogorov-Arnold Network (KAN) models, their final performance approximation accuracy was assessed by examining how closely each model's predictions aligned with the ground truth of the test set, evaluated based on their R^2-values (0-1), indicating how well each model captures variance within the data, with values close to one reflecting a better fit. Aiming to provide insights into each model's prediction accuracy consistency across different value ranges - potentially capturing areas with substantially higher errors - scatter plots were created, illustrating predictions (y-axis) against ground truth (x-axis) for all datapoints of the test set paired with a red line that fits these predictions, and a perfect fit line as reference.

As illustrated in Figure 6.2.1, the KAN model trained to approximate the total material use of the facade design alternatives, achieved a near-perfect R^2-value of 1.00, with highly consistent and low error predictions across the entire value range.

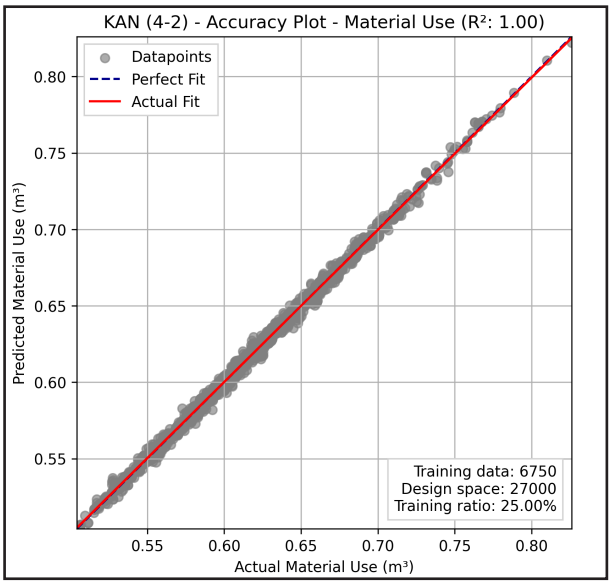


Figure 6.2.1. Prediction accuracy - KAN [9-4-2-1] - material use (R2: 1.00).

As shown in Figure 6.2.2, the KAN model trained to approximate solar heat gains, the second-most complex performance metric, achieved an impressive R^2-value of 0.98, with comparably consistent and low error predictions across the entire value range, though with slightly higher variance overall.

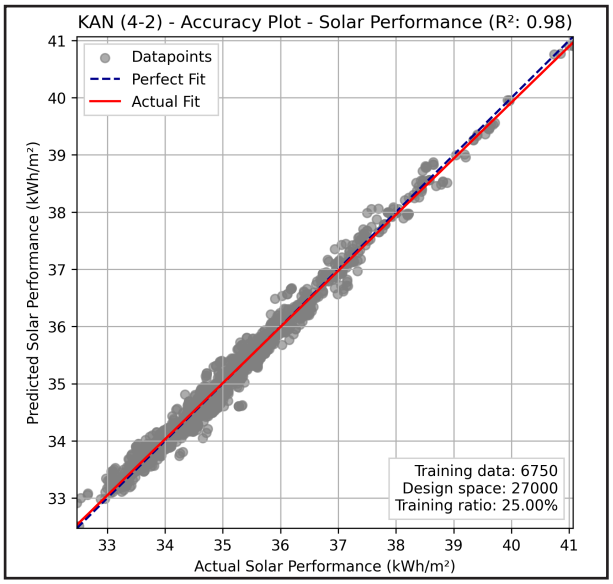


Figure 6.2.2. Prediction accuracy - KAN [9-4-2-1] - solar heat gains (R2: 0.98).

Finally, as illustrated in Figure 6.2.3, the KAN model trained to approximate the highly-complex sound pressure level performance metric, achieved an R^2-value of 0.89, which is an impressive result given its high-complexity and high-dimensionality of x-input vectors. As can be seen, the KAN model has relatively consistent and moderately low error predictions across the entire value range, though with slightly more underprediction at 55.7 and 55.9 dB.

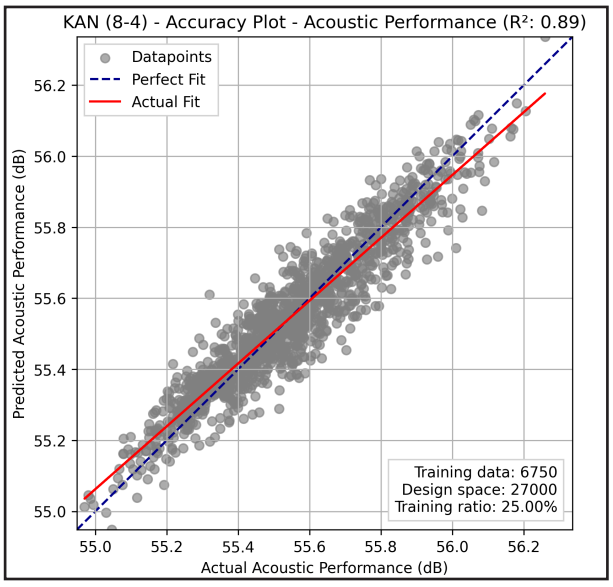


Figure 6.2.3. Prediction accuracy - KAN [9-8-4-1] - acoustic SPL (R2: 0.89).

As aforementioned, this thesis introduces a novel performance-driven design exploration framework integrating Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN), which strives to advance the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, and usability, to facilitate more efficient decision-making in the early design stage. Foundational to enhancing the interpretability between geometry and performance is KAN's ability to offer insight into its decision-making process by creating plots showing how each individual input variable influences its predicted output. As the first step in generating interpretability plots for each trained KAN model, the model's plot function is called in Python, followed by specifying visualisation parameters, including a beta of one hundred, variable scale of 0.175, tick and random sampling set to false, and metric set to forward_n, affecting the transparency of connections between nodes, size of input and output variable names, amount of network updates during plotting - with tick set to false focusing on stable representations and reducing computational demands, reproducibility, and the calculation of influence percentages for each input variable - with forward_n normalising each of them between 0-1.

As illustrated in Figure 6.2.4, the interpretability plot of the KAN model trained to approximate material use shows the model's architecture with nine input variables, two hidden layers with four and two nodes respectively, and one performance output. As can be seen, the network's connections are visualised with varying transparencies, highlighting their influence on performance approximation - thereby making KAN's decision-making process transparent, unlike sensitivity analysis tools which are only able to offer output-level insights, fostering trust between human designers and AI, which is crucial for AI to be smoothly integrated into current design practices, especially as many designers remain hesitant to shift from traditional workflows to AI-driven workflows due to concerns about AI's reliability coupled with their partial loss of control over the design process. As shown, KAN's interpretability is further enhanced by its integration of B-spline activation functions, which tweak their control points during training through backpropagation, until converging into a particular shape, capturing the underlying patterns of the data with maximum accuracy, making KAN's architecture exceptionally straightforward, unlike Multi-Layer Perceptrons (MLP), which generally require much larger architectures based on less-interpretable linear weights. As the final layer of insight into KAN's decision-making process, the interpretability plot showcases how much each design variable influences the predicted performance output, expressed as a percentage. As shown, the influence percentages logically align with the expectations, with W2 and SPD1 showing negligible effects at 0.06% and 0.17%, respectively, with W2 just mirroring certain biocomposite facade panels and SPD1 only affecting the biocomposite facade panel curvature position. The design variables directly controlling total material use show balanced

influence percentages as well, according to sensitivity analysis results and personal assessment, with SCD1 having most influence at 22.20%, followed by BS2 at 7.57%, and TS1 at 6.03%. Finally, the variables affecting the shape of the facade are also logical, with X3 at 7.70%, Y1 at 8.14%, Y2 at 18.97%, and Y3 at 29.17%. Although all Y-variables are similar with only their position in the facade being different, and equally representative in the training dataset, their influence percentages differ significantly. This is mainly due to the fact that Y3 affects half of the entire facade together with BS2, X3, and W2, which are summed up responsible for 15.33% of KAN's predicted output, while Y1 influences half of the entire facade together with TS1, SCD1, and SPD1, responsible for 28.40%, thereby accounting for Y1's lower influence percentage than that of Y3.

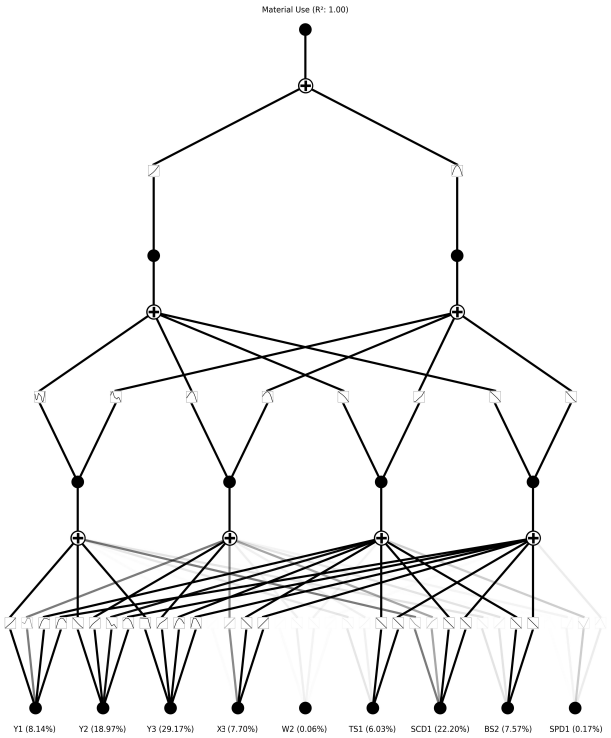


Figure 6.2.4. Original interpretability plot KAN for material use performance.

Moving on with the interpretability plot of the KAN model trained to approximate solar heat gains, as illustrated in Figure 6.2.5. Similar to the sensitivity analysis results and the expectations, BS2 and SPD1 have negligible effect on performance at 0.01% and 0.02%, closely followed by TS1 and SCD1 at 0.07% and 0.09%, lower than anticipated given these variables roles in creating horizontal and vertical shading. The remaining design variables show balanced influence percentages, with W2 at 11.95% influencing the position of the window, X3 at 40.15% affecting glass surface orientation the most, Y2 at 27.82% as the most-influential Y-variable due to its central position, and variables Y3 and Y1 at only 17.73% and 2.16%, due to being more off-centered.

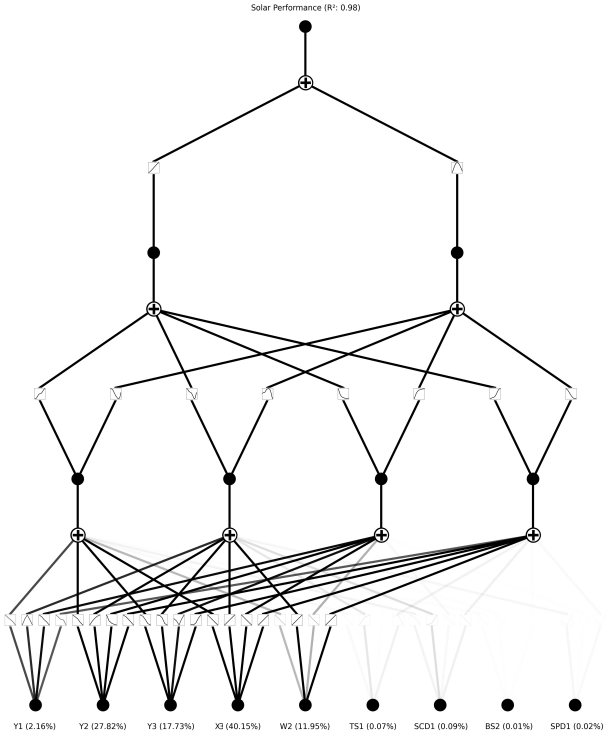


Figure 6.2.5. Original interpretability plot KAN for solar heat gains performance.

Finally, the interpretability plot of the KAN model trained to approximate the sound pressure level, as illustrated in Figure 6.2.6 shows balanced influence percentages for TS1 at 0.09%, BS2 at 0.15%, SCD1 at 0.34%, and SPD1 at 0.54%. The remaining variables, with W2 at 5.52%, X3 at 24.58%, Y1 at 7.03%, Y2 at 31.15%, and Y3 at 30.06%, are too difficult to validate for most users, even though possible when the activation functions are investigated carefully, underscoring the importance of providing the designer with a model which results can be trusted.

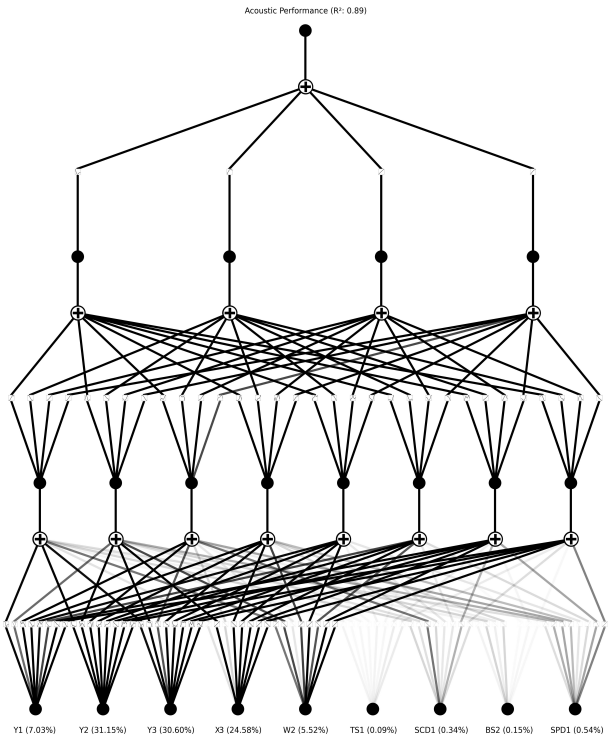


Figure 6.2.6. Original interpretability plot KAN for acoustic SPL performance.

As an additional step in enhancing interpretability, the created plots can be pruned by calling the model's prune function in Python, removing the nodes and edges with minimal influence on the predicted output, based on a predefined threshold. After iterating through different values and comparing the plotting results, a minimum threshold of 0.01 for both node and edge pruning was chosen, simplifying the model's architecture significantly, thereby enhancing interpretability, as illustrated in Figures 6.2.7-6.2.9, though with slightly less precision.

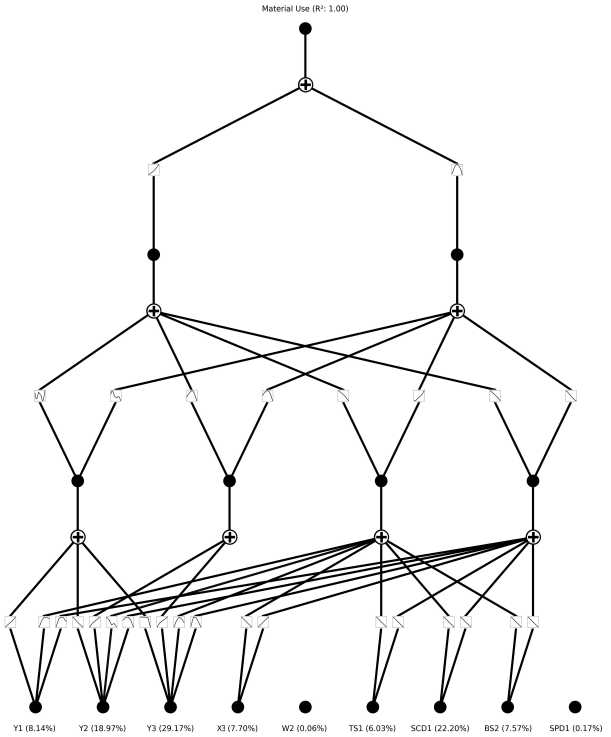


Figure 6.2.7. Pruned interpretability plot KAN for material use performance.

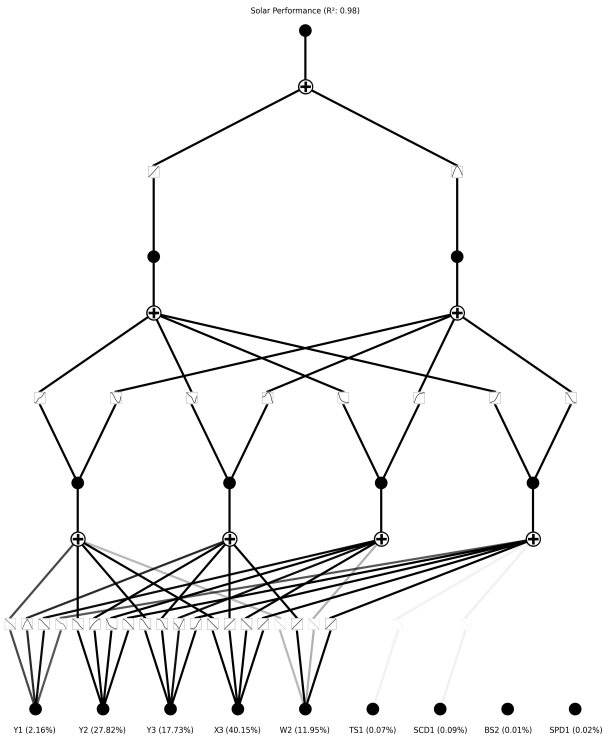


Figure 6.2.8. Pruned interpretability plot KAN for solar heat gains performance.

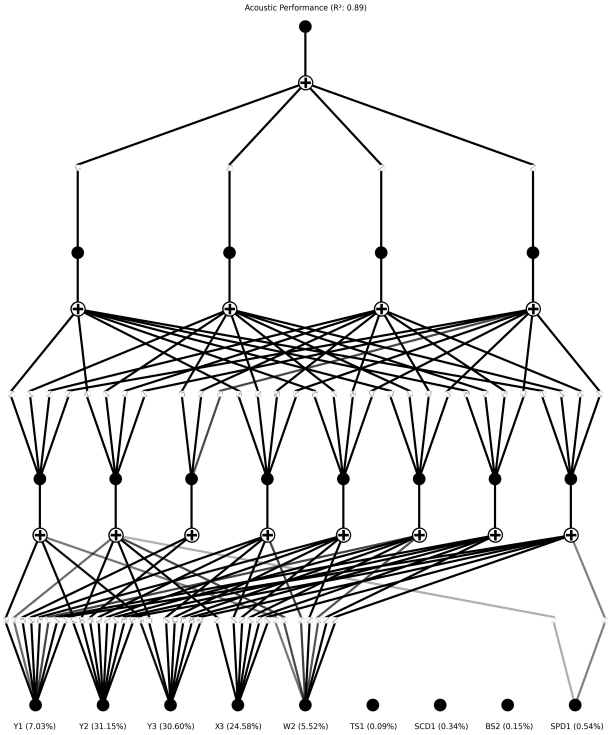


Figure 6.2.9. Pruned interpretability plot KAN for acoustic SPL performance.

6.3 MLP vs KAN

Aiming to demonstrate the potential of Kolmogorov-Arnold Networks (KAN) as a promising surrogate model alternative to traditional Multi-Layer Perceptrons (MLP), the KAN models are systematically compared with identical MLP models, as well as those with four, eight, and sixteen times as many nodes, across varying levels of performance approximation complexity, evaluating R^2-scores at intervals of 300 up to 6,750 samples, offering insights into KAN's performance approximation accuracy as data and complexity increases, relative to MLP.

The first step in training the MLP models, after preparing the training vectors with almost the exact same logic as used for the KAN models - with the only difference being their final transformation into numpy arrays instead of torch tensors - involved initialising the models in Python 3.9.13 using Tensorflow 2.10.0. Aiming to optimise computational efficiency, DirectML was integrated for GPU acceleration, enabling parallel computing. Next in initialising the MLP models, the so-called Adaptive Moment Estimation (Adam) optimiser was chosen to minimise Mean Squared Errors (MSE) between predictions and ground truth during training, by adjusting weights and biases through back-propagation. Additionally, the MLP models were initialised with so-called Rectified Linear Unit (ReLU) activation functions, aiming to introduce non-linearity by setting negative inputs to zero and keeping positive inputs unchanged. As the final step in initialising the MLP models, each model architecture was set up with nine inputs, two hidden dense layers

with two dropout layers randomly deactivating nodes during training to prevent overfitting, and one output, with the number of nodes in the hidden layers carefully selected to allow for thorough comparison of performance approximation efficiency between MLP and KAN. Consequently, a total of six MLP models were trained: two models for material use with 4-2 and 16-8 nodes respectively, two for solar heat gains with 4-2 and 32-16 nodes, and two for sound pressure level with 8-4 and 128-64 nodes. On top of that, each MLP model was initialised with a fixed number of epochs set at 1000, paired with an early stopping mechanism with a patience of 50 epochs, keeping the best-performing state.

Following the initialisation of the MLP models, an extensive hyperparameter tuning process was carried out to optimise each model's performance approximation accuracy. This process involved iterating through various values for learning rate [0.0001, 0.0005, 0.001, 0.005, 0.01], batch size [8, 16, 32, 64], and dropout probability [0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01], with different optimal hyperparameter settings for each model, yielding the lowest Mean Squared Errors assessed on their validation set. After the hyperparameter tuning process, their final performance approximation accuracy was assessed by examining how closely their predictions aligned with the ground truth of the test set, evaluated based on their R^2-values. Additionally, aiming to provide insight into each model's prediction accuracy consistency across different value ranges, scatter plots were created, similar to those created for the KAN models.

As illustrated in Figure 6.3.1, the MLP model trained to approximate total material use - with 4 and 2 nodes in its hidden layers respectively, an optimal learning rate of 0.0005, batch size of 8, and dropout probability of 0.01 - achieved an impressive R^2-value of 0.94, slightly worse than the KAN model with the same architecture, which yielded a near-perfect R^2-value of 1.00. Although the performance approximation accuracy of the MLP model is excellent overall, with relatively consistent and low error predictions for most datapoints, it has significant issues with predictions in the value range 0.50-0.54 m^3, predicting all values as 0.54 m^3. Additionally, it has strong underpredictions in the value range 0.71-0.83 m^3, making the MLP model significantly less reliable. This underscores the importance of the scatter plots, providing insight into prediction accuracy consistency, as numeric assessment alone is not everything - a model with a lower R^2-value but more consistent predictions across its entire value range would be a more effective surrogate model, as consistency is crucial for creating reliable models, as well as the case for life in general.

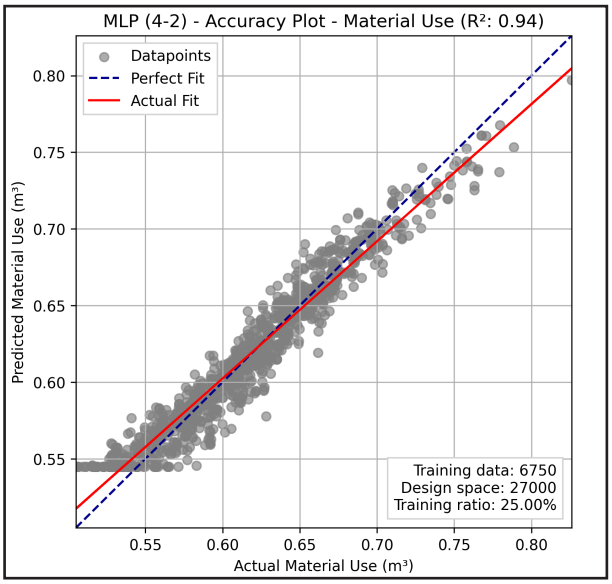


Figure 6.3.1. Prediction accuracy - MLP [9-4-2-1] - material use (R2: 0.94).

Following this, the smallest MLP model architecture that achieved similar results as the KAN model required four times as many neurons in its hidden layers. As illustrated in Figure 6.3.2, this MLP model, trained to approximate total material use - with 16 and 8 neurons in its hidden layers respectively, an optimal learning rate of 0.001, batch size of 8, and dropout probability of 0.005 - yielded a near-perfect R^2-value of 1.00, with highly consistent and low errors across the entire value range, though with slightly more underprediction from 0.71 to 0.83 m^3, thus a bit less reliable than the KAN model.

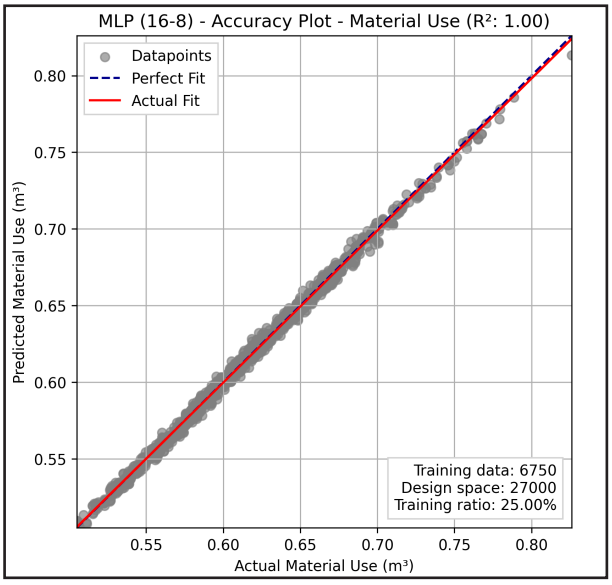


Figure 6.3.2. Prediction accuracy - MLP [9-16-8-1] - material use (R2: 1.00).

As shown in Figure 6.3.3, the MLP model trained to approximate solar heat gains - with 4 and 2 nodes in its hidden layers respectively, an optimal learning rate of 0.005, batch size of 8, and dropout probability of 0.001 - achieved a respectable R^2-value of 0.85, though notably below the R^2-value of 0.98 yielded by the KAN model with the same architec-

ture. As can be seen in the scatter plot, the MLP model has moderately consistent and moderately low error predictions, with strong overpredictions from 33.5 to 34.0 kWh/m², and even stronger underpredictions in the value range 37.2-41.0 kWh/m², indicating a relatively unreliable surrogate model.

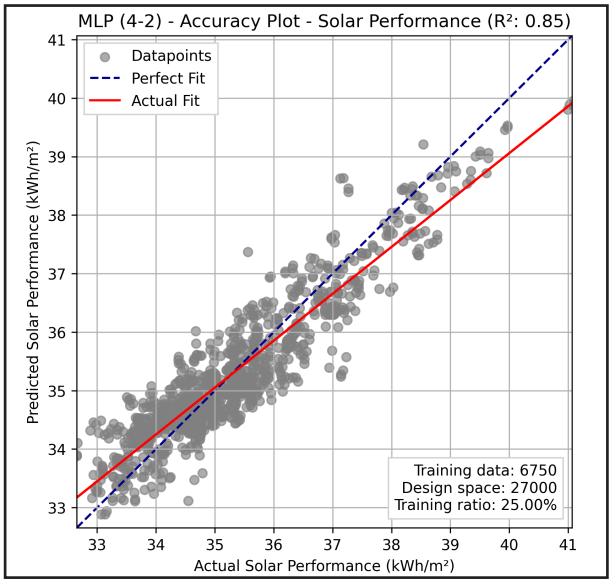


Figure 6.3.3. Prediction accuracy - MLP [9-4-2-1] - solar heat gains (R2: 0.85).

Consequently, the smallest MLP model architecture ture that achieved similar results as the KAN model required eight times as many nodes in its hidden layers. As shown in Figure 6.3.4, this MLP model, trained to approximate solar heat gains - with 32 and 16 nodes in its hidden layers respectively, an optimal learning rate of 0.0005, batch size of 8, and dropout probability of 0.0001 - yielded an R²-value of 0.98, with highly consistent and low error predictions across the entire value range, even slightly more consistent than the KAN model which has a bit of overprediction from 32.6-32.8 kWh/m² and a bit of underprediction from 39.2-39.7 kWh/m².

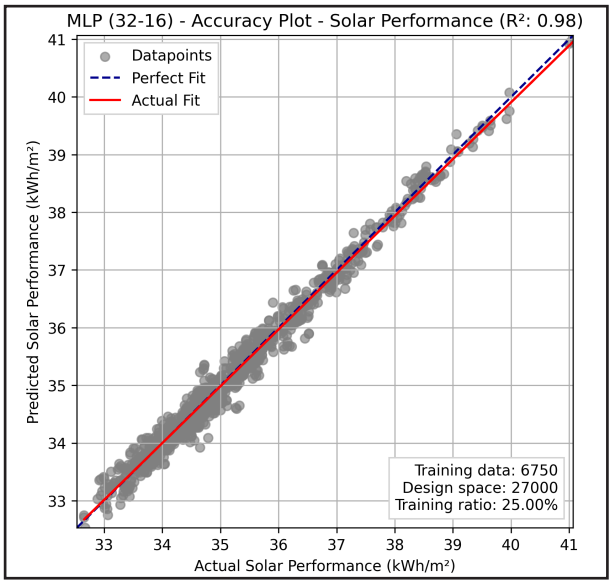


Figure 6.3.4. Prediction accuracy - MLP [9-32-16-1] - solar heat gains (R2: 0.98).

Finally, as illustrated in Figure 6.3.5, the MLP model trained to approximate sound pressure level - with 8 and 4 nodes in its hidden layers respectively, an optimal learning rate of 0.001, batch size of 32, and dropout probability of 0.00005 - achieved a significantly poor R²-value of 0.26, with highly inconsistent and high error predictions across the entire value range, with strong overpredictions from 55.0-55.5 dB and underpredictions from 55.7-56.5 dB.

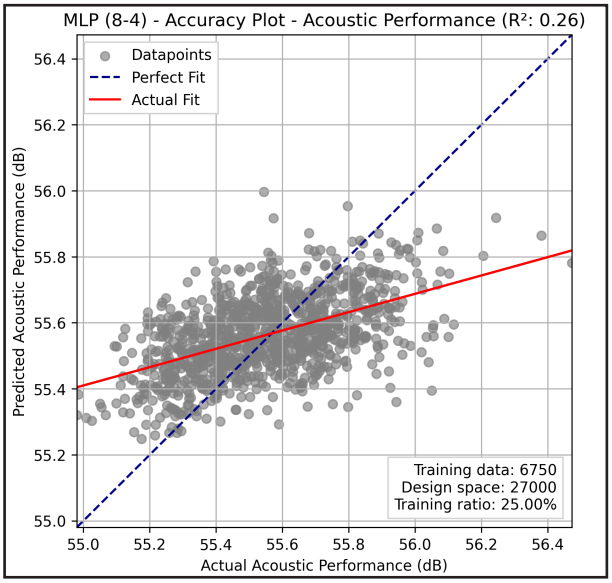


Figure 6.3.5. Prediction accuracy - MLP [9-8-4-1] - acoustic SPL (R2: 0.26).

Aiming to create an MLP model that achieves similar results as the KAN model (R²-value of 0.89), the number of nodes in its hidden layers was increased sixteen times. As shown in Figure 6.3.6, this MLP model - with 128-64 nodes in its hidden layers respectively, an optimal learning rate of 0.0005, batch size of 16, and dropout probability of 0.00005 - still performed significantly poor with an R²-value of 0.42, with moderately inconsistent and moderately high error predictions across the entire value range.

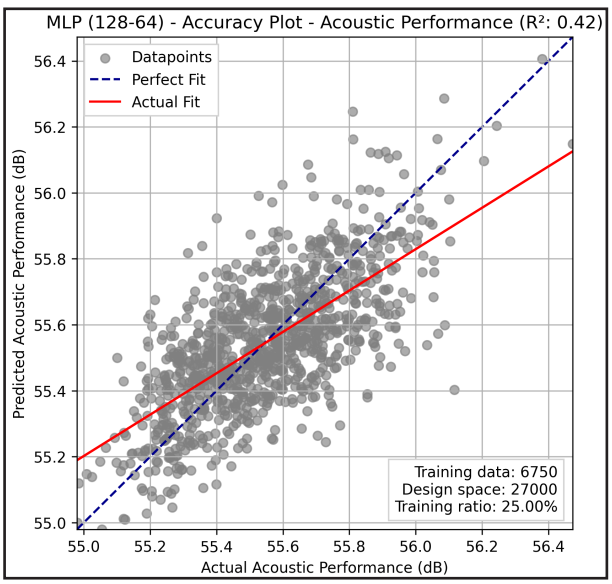


Figure 6.3.6. Prediction accuracy - MLP [9-128-64-1] - acoustic SPL (R2: 0.42).

In a final attempt to create an MLP model achieving similar results as the KAN model, two dense layers were added. As shown in Figure 6.3.7, this MLP model - with 128-64-32-16 nodes in its hidden layers respectively, an optimal learning rate of 0.0005, a batch size of 8, and dropout probability of 0.0001 - yielded a slightly better, but still significantly poor R^2-value of 0.44, with slightly less consistent, but slightly lower errors across the entire value range.

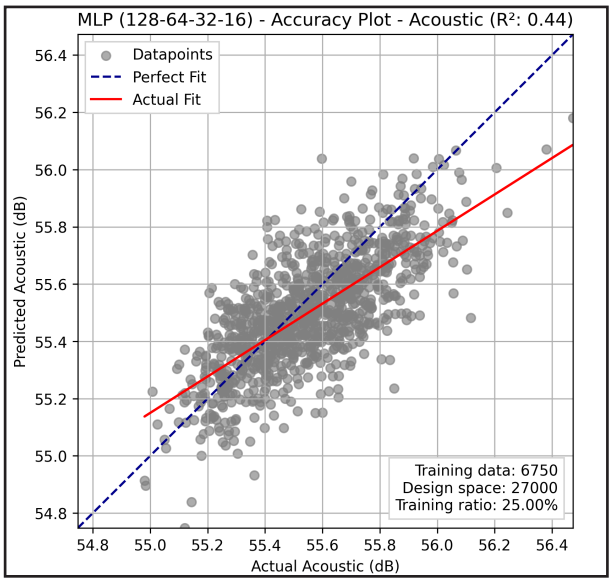


Figure 6.3.7. Prediction accuracy - MLP [9-128-64-32-16-1] - SPL (R2: 0.44).

Aiming to provide insights into KAN's efficiency in approximating non-linear functions as both training data and performance approximation complexity increases, relative to MLP, the three KAN models are systematically compared with the seven MLP models, evaluating their R^2-scores at intervals of 300 up to 6,750 training vectors. Before examining the results, it is essential to note that each surrogate model's optimal hyperparameter settings, derived from training on 6,750 vectors, were kept constant for the smaller datasets. Despite the fact that these models trained on smaller datasets could have different optimal hyperparameter settings, the high number of epochs and early stopping mechanism still allows them to find semi-optimal results.

As shown in Figure 6.3.8, KAN significantly outperform MLP from a performance approximation accuracy and convergence perspective, with much smaller and more interpretable model architectures, which are also faster and more intuitive to train - requiring fewer iterations through hyperparameters and fewer epochs - as well as more reliable due to their more consistent error predictions across value ranges. This not only reduces computational demands significantly by minimising training data required to yield acceptable prediction accuracies - assuming sufficient representativeness of the dataset - and for optimising KAN's stratified sampling logic, but also make KANs able to adapt to highly-complex performance metrics like sound pressure level, based on high-dimensional x-input vectors, where MLPs fail significantly.

Examining the results, it can be seen that the KAN model trained for material use, yields an R^2-value of 0.994 after only 600 training vectors, while the MLP model with four times as many nodes, requires 3.5 times as many training vectors to yield similar results, highlighting KAN's fast convergence. This trend extends to solar heat gains, where the KAN model yields an R^2-value of 0.978 after just 900 training vectors, while the MLP model with eight times as many nodes, requires three times as many training vectors to achieve similar results. The KAN model trained for sound pressure level, achieves an R^2-value of 0.801 after only 1200 training vectors, while the three MLP models fail to adapt to the high-complexity at all, with fluctuating R^2-values and larger model architectures requiring more training vectors to yield R^2-values above zero.

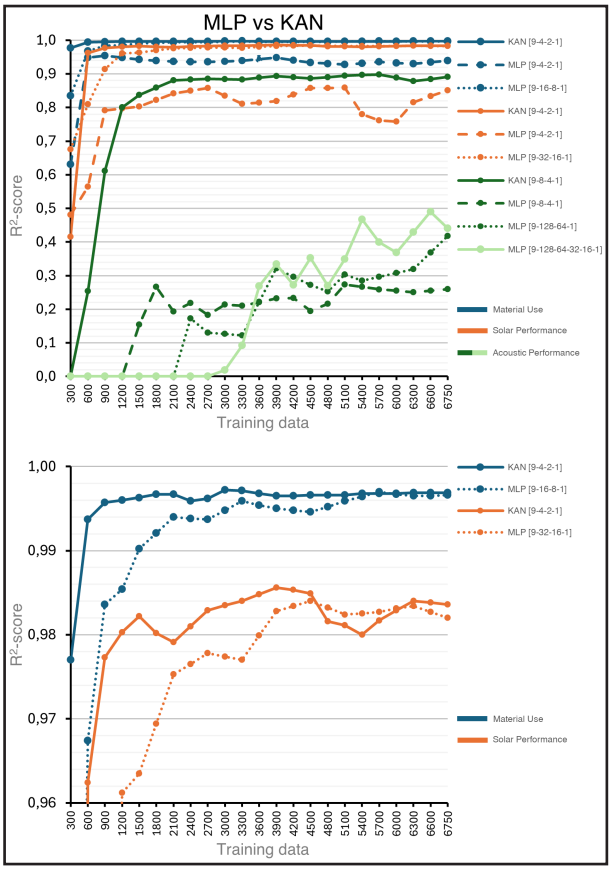


Figure 6.3.8. Prediction accuracy MLP vs KAN as a function of training data.

In conclusion, these findings indicate that Kolmogorov-Arnold Networks (KAN) are highly efficient in approximating non-linear functions across varying levels of performance approximation complexity and dataset sizes, while also reducing computational demands due to their fast convergence, making them a highly promising surrogate model alternative to traditional Multi-Layer Perceptrons (MLP).

7

Design exploration

As aforementioned, the self-organising map (SOM) based design framework, which is the current most advanced performance-driven design exploration framework in facade design, lacks balanced human-AI interaction in its exploration process, consisting of between-cluster and inside-cluster exploration. Particularly, inside-cluster exploration, which involves inspecting hundreds of similar design alternatives, is overly AI-dominant and lacks interactivity with the human designer, making the design framework less effective and intuitive.

Therefore, as the final phase of the proposed AI-driven facade design framework, this chapter delves into the process of creating a novel design exploration framework consisting of an orientation and fine-tuning phase, which aim to balance human-AI interaction more efficiently, making design exploration more interactive, as well as to include less-geometry related design variables, enhancing the framework's optimisation capabilities.

7.1 Orientation

As the foundation of the design exploration process, the orientation phase allows the designer to navigate the Self-Organising Map (SOM) according to geometry typology and approximated performance, facilitating fast design orientation based on design variables with high influence on geometry. As aforementioned, after training the SOM, the final clustering results were integrated back into Grasshopper by extracting the 125 geometry-related node design vectors along with their corresponding coordinates on the SOM grid, followed by creating a hexagonal 15x15 grid in Grasshopper, baking the facade design geometries based on the extracted node design vectors using the parametric model, and positioning them on the SOM grid according to their coordinates, as illustrated in Figure 7.1.1.

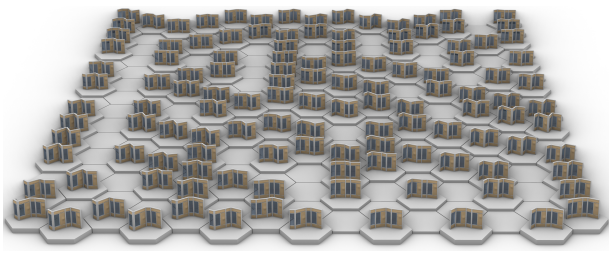


Figure 7.1.1. Clustering facade design geometries on 15x15 hexagonal SOM.

Following this, the performance of these 125 most geometrically influential facade design variants are approximated by the trained Kolmogorov-Arnold Network (KAN) models, based on their nine-dimensional geometry-related design vectors, for material use, solar heat gains, and sound pressure level. As each designer prioritises these performance metrics differently, the design exploration framework allows for weight allocations, with default values of 0.50 for material use, 1.00 for solar heat gains, and 0.25 for sound pressure level. These weighted performance approximations are summed up, remapped to a domain of 0 to 10 meters, and used to generate a performance surface using Grasshopper's delaunay mesh component, which hovers above the SOM as a visual performance indicator, with higher elevations indicating better-performing facade design variants below, as illustrated in Figure 7.1.2.

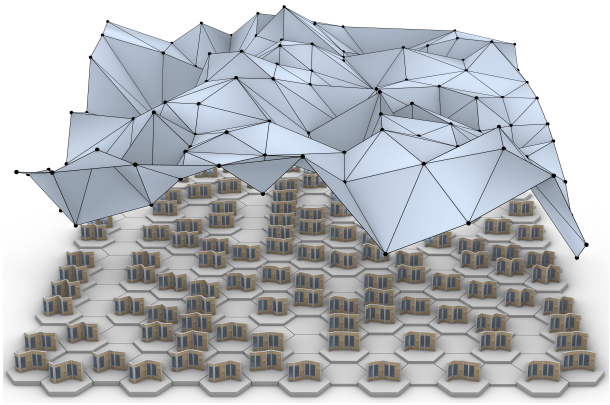


Figure 7.1.2. Weighted performance indication surface hovering above SOM.

Furthermore, a horizontal performance threshold plane and a legend are created to make design exploration more effective, by enabling the designer to filter facade design variants based on a performance score threshold from one to ten - with one corresponding to the lowest and ten to the highest point on the performance surface - or by setting a percentage of facade design variants that surpass the threshold plane. Additionally, to facilitate easier comparison between facade design variants from a qualitative and quantitative perspective, those surpassing the threshold plane are highlighted with vertical lines indicating their relative positions on the SOM, as well as automatically copied above the performance surface, as illustrated in Figure 7.1.3.

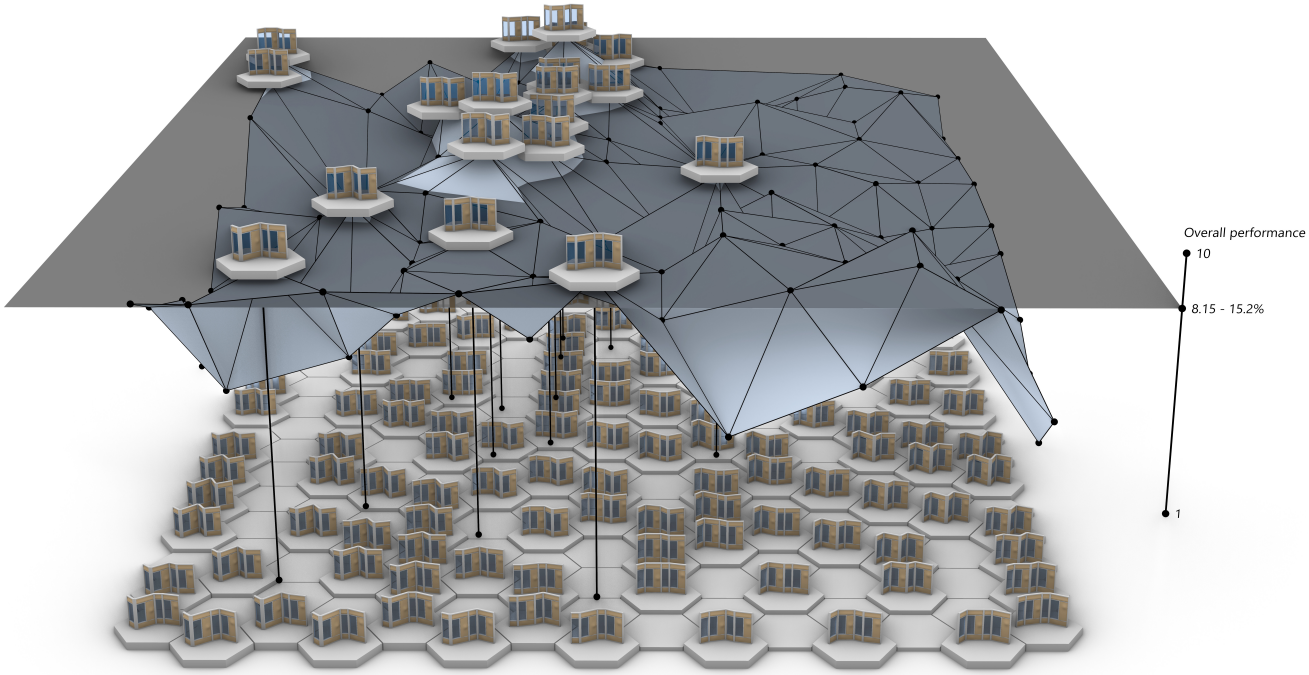


Figure 7.1.3. Final design exploration interface of the orientation phase with the weighted performance surface hovering above the SOM, threshold plane, and legend.

During design exploration, designers can adjust the weights of the performance metrics and the height of the threshold plane to find the best-performing facade design variant, balancing architectural and quantitative performance. As this optimal design serves as the foundation of the fine-tuning phase, the facade shown in Figures 7.1.4-7.1.5 is selected.

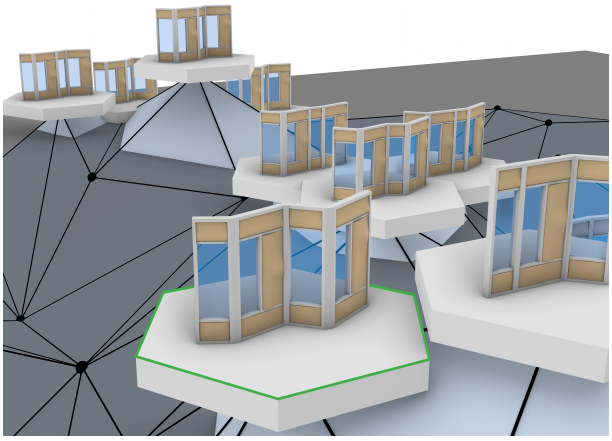


Figure 7.1.4. Perspective view of the selected best-performing facade design.

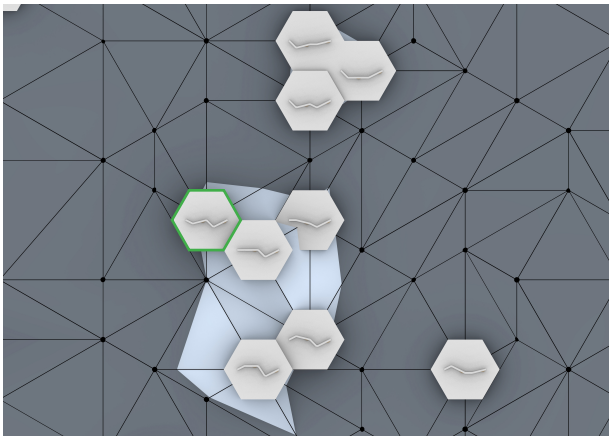


Figure 7.1.5. Overhead view of the selected best-performing facade design.

7.2 Fine-tuning

As the final step of the design exploration process, the fine-tuning phase focuses on smaller geometric changes within the selected best-performing facade design from the orientation phase, allowing designers to adjust design variables strategically, based on real-time performance feedback and actionable insights provided by the KAN models, interactively guiding designers towards optimal performance.

As the first step in creating the fine-tuning design exploration interface, the three-dimensional geometry-related design vector of the best-performing facade design from the orientation phase, is extracted from the SOM and converted into a nine-dimensional vector by adding design variables X3, W2, TS1, SCD1, BS2, and SPD1 - which are significantly less-influential on geometry and therefore guiding the fine-tuning phase - with default values set to the first in their ranges, followed by baking the geometry based on this vector using the parametric model, and placing it on a hexagonal base.

Next, the top part of the interactive information panel is created, displaying the adjustable weights, names, and absolute performance approximations of each metric, enabling the designer to adapt the design exploration framework to their own specific priorities and gain insights into the current facade design's quantitative performance.

Following this, the weighted performance approximations are summed up, remapped to a domain of one to ten, and used to generate a dynamic performance indictor, displayed as a horizontal line with the current facade design's performance score marked by a blue dot, paired with the scores of each individual metric indicated by light grey dots, ena-

bling the designer to quickly assess the current facade design's quantitative performance overall and across different metrics. Additionally, the designer can use the dynamic performance indicator line to inspect various facade designs by setting a specific performance score, indicated with a red cross. This not only shows the specific facade design, but also highlights its design variable values with red lines underneath specific points that indicate the number of values in each design variable's value range, as illustrated in the bottom-right side of the interactive information panel in Figure 7.2.1. The design variable values of the current facade design are also highlighted by blue dots, while the best-performing design variable values are marked by green circles.

As the final step in creating the fine-tuning design exploration interface, the design variable names, their absolute values, and their relative influence on quantitative performance are displayed, offering the designer the ability to adjust design variables strategically, as some might not have as big of an impact on performance as expected and might predominantly affect geometry, therefore able to be freely tweaked within their original domain of values. After extracting the influence percentages for each performance metric from the Kolmogorov-Arnold Network (KAN) models' interpretability plots, the specified metric weights are applied, and the average influence percentage for each design variable is calculated. These values are then remapped to a domain of 0 to 0.7 meters, and used to create a surface using Grasshopper's interpolate curve and ruled surface components, with higher elevations indicating greater influence on performance, as illustrated in figure 7.2.1. Additionally, an influence percentage threshold is created, with the default percentage set at 5.0%, highlighting design variables with influence below this threshold with blue lines underneath their variable names.

During design exploration, designers can tweak design variables based on real-time performance feedback and actionable insights provided by the Kolmogorov-Arnold Network models, adjust weights of the performance metrics to adapt the exploration framework to their own priorities, set specific performance scores to inspect various facade designs and learn from their design variable values, and specify the influence percentage threshold to assist in identifying design variables with minimal impact on quantitative performance, allowing designers to focus on the most important design variables more easily and ultimately find the best-performing facade design variant, balancing architectural and quantitative performance best.

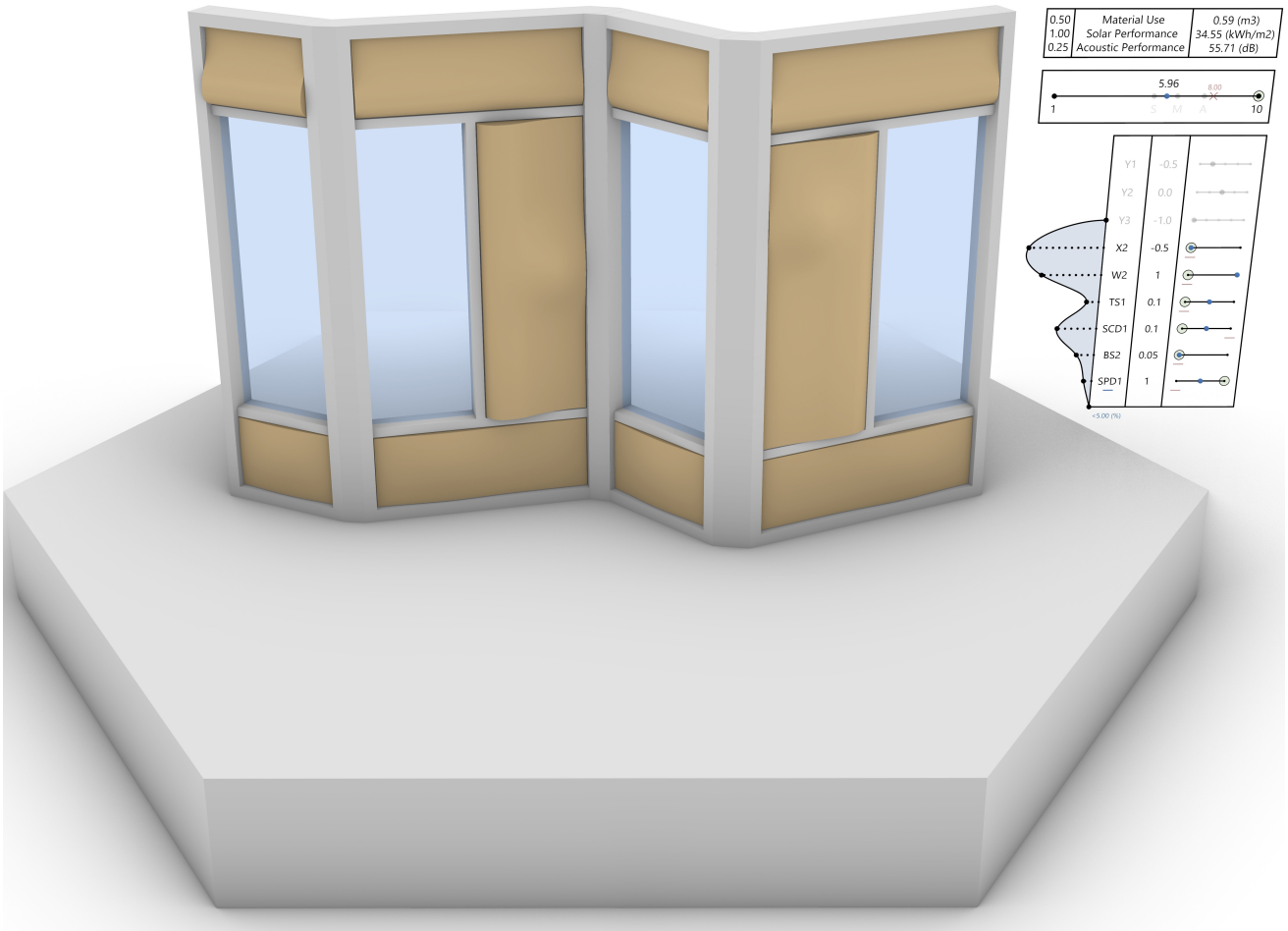


Figure 7.2.1. Final design exploration interface of the fine-tuning phase with the information panel interactively guiding the designer towards optimal performance.

7.3 Validation

As aforementioned, a novel approach to early-stage facade design exploration was developed to balance human-AI interaction more efficiently and integrate less-geometry related design variables, aiming to make design exploration more interactive, thereby more effective and intuitive, while enhancing optimisation capabilities. Where design exploration has proven to be more interactive by the incorporation of performance metric weights reflecting designer priorities in the orientation phase, and by empowering designers with control over the design process with AI assisting in the fine-tuning phase, the integration of less-geometry related design variables has proven successful by operating the fine-tuning phase independently of geometric variables.

However, whether designers perceive this increased interactivity as beneficial for design exploration remains subjective. Therefore, to validate the novel design exploration framework's effectiveness compared to traditional SOM-based between-cluster and inside-cluster design exploration, sixteen Building Technology and Architecture master students from Delft University of Technology have engaged in one-on-one design exploration sessions, with feedback obtained through a questionnaire.

This questionnaire, included in its entirety in Appendix H, consists of 22 multiple-choice questions categorised into five distinct sections. In the first section (Q1-Q3), the participant's background in using AI-driven design tools, parametric modeling, and facade design is evaluated to place their feedback into context by enabling the creation of participant profiles based on experience levels. In the second section (Q5-Q10), the orientation phase is assessed on its overall impression, effectiveness in enabling fast design orientation, intuitiveness, usefulness of its design tools in narrowing design options, and satisfaction with the final design outcome. Building on this, the third section (Q11-Q16), focuses on assessing the fine-tuning phase, examining its overall impression, effectiveness in refining the chosen design, intuitiveness, usefulness of its information panel in adjusting design variables strategically, and satisfaction with the final design outcome. In the fourth section (Q17-Q18), the overall performance of the novel design exploration framework is evaluated, examining its effectiveness in balancing qualitative and quantitative aspects and overall design exploration. Finally, in the fifth section (Q19-Q22), the novel framework is compared to the traditional approach by evaluating the orientation and fine-tuning phase against the between- and inside-cluster phase, respectively, in terms of visualisation, speed, usability, and effectiveness in decision-making, as well as by evaluating the effectiveness of design exploration of both frameworks overall. While most questions (Q1-Q20) are on a five-point scale, comparative questions (Q21-Q22) allow participants to select either the traditional framework, the novel approach, or remain neutral.

Delving into the results of the second, third, and fourth section of questions (Q5-Q18), as shown in Figure 7.3.1 and highlighted in blue, orange, and green, respectively, with the size of the dots reflecting the number of participants, it can be concluded that most participants are highly positive about the novel design exploration framework's effectiveness, assessing each section with average scores of 4.30, 4.33, and 4.34, respectively. In addition to the small score differences between each section, the mean values of each individual question within each section are close to the average score for that section as well, with bounds of [+0.20, -0.30, ±0.50], [+0.17, -0.20, ±0.37], and [+0.22, -0.21, ±0.43], respectively, thereby indicating consistent overall performance.

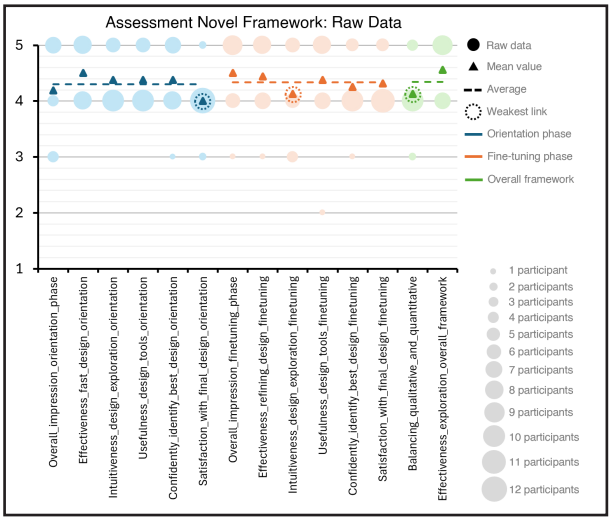


Figure 7.3.1. Assessment Novel Framework: Raw Data.

As aforementioned, to examine potential correlations between experience levels and assessment, participants were grouped into three profiles: low, moderate, and high experience based on the summed total of their responses to Q1-Q3. While two participants were grouped into the low profile, the moderate and high profiles included seven participants. For each question, the average score within each profile was used, as depicted in Figure 7.3.2.

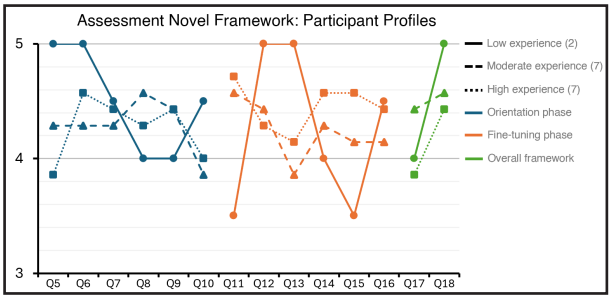


Figure 7.3.2. Assessment Novel Framework: Participant Profiles.

To better understand these results, Figure 7.3.3 was created, showing the averages for each section and their bounds across the three participant profiles. For the orientation phase: low, moderate, and high experience participants scored 4.50 [+0.50, -0.50, ±1.00], 4.29 [+0.28, -0.43, ±0.71], and 4.26 [+0.31, -0.40, ±0.71], respectively, showing higher scores and greater variability for low experience participants. In the fine-tuning phase, averages were 4.25 [+0.75, -0.75, ±1.50], 4.23 [+0.34, -0.37, ±0.71], and 4.45 [+0.26, -0.31, ±0.57], with the highest scores and lowest variability for high experience participants. Lastly, for the overall framework, averages were 4.50 [+0.50, -0.50, ±1.00], 4.47 [+0.07, -0.07, ±0.14], and 4.14 [+0.28, -0.29, ±0.57], with the lowest scores for high experience participants and greatest variability for low experience participants. The consistently high variability in the low experience group is likely due to its underrepresentativeness with only two participants, compared to seven, amplifying the impact of individual scores.

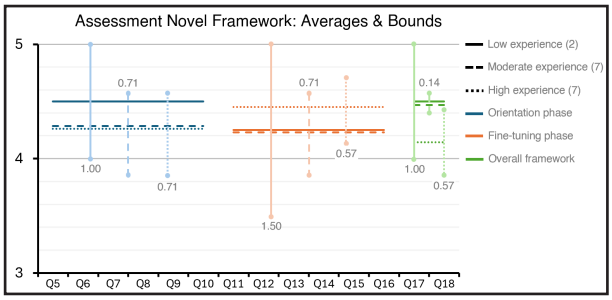


Figure 7.3.3. Assessment Novel Framework: Averages & Bounds.

Building on this, Figure 7.3.4 was created, showing the averages and bounds for all sections combined, based on the summed total divided by three, across the three participant profiles. As shown, low, moderate, and high experience participants scored 4.42 [+0.08, -0.17, ±0.25], 4.33 [+0.14, -0.10, ±0.24], and 4.28 [+0.17, -0.14, ±0.31], respectively. Given these results, a slight trend of decreasing performance scores and increasing variability with increasing experience levels is observed. However, given the small sample size of only sixteen participants, and the underrepresentativeness in the low experience group which could have affected the results significantly, these trends should be interpreted carefully.

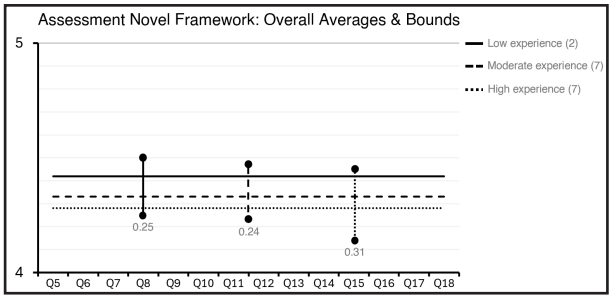


Figure 7.3.4. Assessment Novel Framework: Overall Averages & Bounds.

As aforementioned, in the fifth section of questions (Q19-Q22), the novel framework is compared to the traditional approach. As shown in Figure 7.3.5, between-cluster exploration (left) aligns closely with the orientation phase, with small differences in visualisation, insights, and design exploration tools, while inside-cluster exploration (right) differs significantly from the fine-tuning phase, presenting designs sequentially with quantitative filter options.

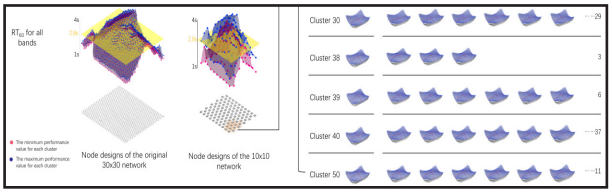


Figure 7.3.5. Traditional exploration. Adapted from "*Design exploration of quantitative performance and geometry typology for indoor arena based on self-organizing map and multi-layered perceptron neural network*", by Turrin et al., 2020 [94].

Delving into the results of Q19-Q22, as shown in Figure 7.3.6, it can be concluded that most participants are strongly favouring the novel framework over the traditional approach in terms of visualisation, speed, usability, and effectiveness in decision-making. As illustrated, low, moderate, and high experience participants scored 5.00, 4.14, and 4.71, as well as 5.00, 4.71, and 4.86 in favour of the orientation and fine-tuning phase, with a score of three equal to remaining neutral. They scored 3.00, 3.00, and 2.86, as well as 3.00, 2.71, and 3.00, assessing confidence in design choices and overall effectiveness, with a score of two equal to remaining neutral.

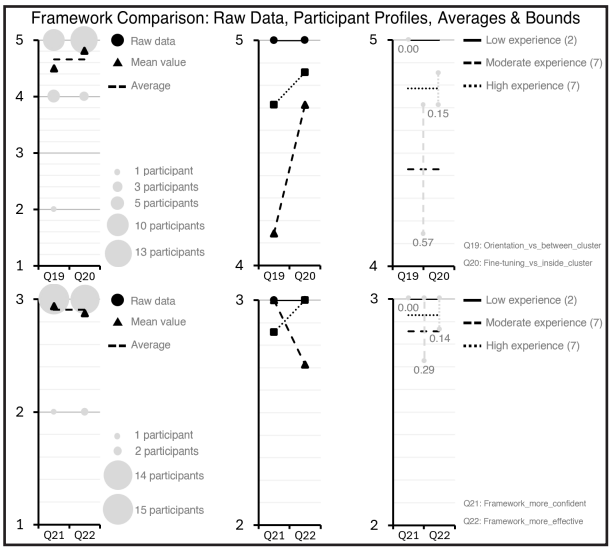


Figure 7.3.6. Framework Comparison: Raw Data, Profiles, Averages & Bounds.

Interpreting the results, no significant correlations between experience levels and assessment were observed, while feedback from the one-on-one design exploration sessions revealed a strong preference for the novel framework due to its interactive nature and balanced consideration of qualitative and quantitative aspects during design exploration.

Conclusion

This thesis aimed to advance the facade design process by developing a novel performance-driven design exploration framework, inspired by the SOM-MLP framework and applied in the design process of an aluminium-based biocomposite curtain wall facade fragment, comprising SOM and KAN.

By integrating KAN as a surrogate model within an SOM-based design framework, which is the current most advanced performance-driven design exploration framework used in facade design, it was aimed to enhance computational efficiency, substituting heavy performance simulations with fast approximations. While initial training on 2,700 simulations yielded an R^2-score of over 0.88 for all KAN models with only 134 hours of computation, a total of 6,750 simulations, covering 25% of the entire dataset, were conducted to ensure proper generalisation, requiring 335 hours of computation and saving over 1,000 hours. This not only allows for the exploration of a larger design space to find better-performing designs, but also for the adaptation to higher-complex design problems.

Furthermore, KAN was systematically compared with various MLP models of different architectures to evaluate its efficiency in approximating non-linear functions as training data and performance approximation complexity increases. The results show that KAN significantly outperforms MLP from a performance approximation accuracy and convergence perspective, achieving these results with much smaller architectures. While KAN yields R^2-values of 0.99 for material use after 600 vectors, 0.98 for solar heat gains after 900 vectors, 0.80 for sound pressure level after 1,200 vectors, and 1.00, 0.98, and 0.89 after 6,750 vectors respectively, the MLP models require around 3.5 times as many vectors to yield similar results for material use and solar heat gains, and even the deepest MLP model, with four hidden layers of 128, 64, 32, and 16 nodes, fails to adapt to the high-complexity of sound pressure level, achieving a poor R^2-value of 0.44. Despite KAN's faster convergence being overshadowed by additional training vectors used to ensure proper generalisation, it still holds the ability to reduce computational demands significantly under different conditions, by minimising training data required to yield acceptable prediction accuracies and for optimising stratified sampling. KAN's superior performance approximation accuracy for sound pressure level shows its potential to adapt to even more complex design problems, while traditional MLP models already fall short. This is crucial given the fact that this thesis focuses on only a small facade fragment. Additionally, KAN has proven to be much faster and more intuitive to train, requiring around fifteen times fewer hyperparameter iterations and ten times less epochs, as well as more consistent in its error predictions across value ranges. This not only makes KAN more computationally efficient and reliable, but also accesible to those with limited expertise in machine learning.

Moreover, introducing KAN as a surrogate model alternative to MLP, sought to enhance the interpretability between geometry and performance, enabling designers to identify and focus on relevant design variables that affect both; crucial as design spaces are often high-dimensional and SOMs face the curse of dimensionality, while also empowering them with actionable insights to adjust design variables strategically toward optimal performance. This study confirmed enhanced interpretability through the creation of plots showing the percentual impact of each design variable on performance predictions, validated by personal assessment as well as sensitivity analysis results, which also highlighted KAN's advantages as an all-in-one solution with transparent decision-making and four times faster results. Thus, KAN not only makes the design process more efficient, but also enhances usability and reliability, and reduces computational demands by enabling stratified sampling and through faster processing.

Finally, a novel approach to design exploration was developed to balance human-AI interaction more efficiently than traditional methods, and integrate less-geometry related design variables, aiming to make design exploration more interactive, thereby more effective and intuitive, while enhancing optimisation capabilities. Where one-on-one design exploration sessions with experts in the field and feedback obtained through a questionnaire has validated enhanced effectiveness and intuitiveness, less-geometry related design variables have been successfully integrated by operating part of design exploration independently of geometric variables.

To conclude, the SOM-KAN framework has proven to advance the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, usability, and accessibility, facilitating more efficient decision-making in early design stages, leading to superior architectural and sustainable solutions.

Further research could focus on extending the SOM-KAN framework to other design practises within the AEC sector, assessing its effectiveness on larger and more complex design problems, and investigating the limits of dimensionality in SOMs.

Reflection

What is the relation between this project and the Building Technology master programme?

As aforementioned, this thesis introduces a novel performance-driven design exploration framework, inspired by the SOM-MLP framework and applied in the design process of an aluminium-based biocomposite curtain wall facade fragment, integrating Self-Organising Maps (SOM) and Komogorov-Arnold Networks (KAN), which has proven to advance the facade design process by enhancing computational efficiency, performance approximation accuracy, interpretability, reliability, usability, and accessibility, facilitating more efficient decision-making in the early design stage, leading to better architectural and sustainable solutions. At its core, this thesis focuses on leveraging computational and AI-driven tools to advance the approach to early-stage facade design. This thesis aligns well with the Building Technology master programme, which focuses on sustainable and innovative design of buildings and their components, with the role of AI becoming increasingly prevalent, as well as the graduation themes: Design Informatics and Facade & Product Design, and the courses offered throughout the programme. It also aligns with Climate Design through the integration of sound pressure level and solar heat gains performance simulations, and less with Structural Design, although mechanical properties of biocomposites were examined. Finally, this thesis integrates biocomposites as a case study material to address sustainability, which is a foundational aspect within the master programme, and uses performance metrics that address sustainability, energy demands, and urban noise pollution.

How did research and design affect each other?

The research on Self-Organising Maps (SOM) and Kolmogorov-Arnold Networks (KAN) directly influenced the design process by providing the theoretical foundation for integrating these AI-driven tools, while the research on biocomposites enabled realistic performance simulations by using material properties derived from the best-performing compound identified by the design tool. In turn, the design process influenced the research on KAN by demonstrating its efficiency in approximating non-linear functions as training data and performance approximation complexity increases, relative to traditional MLP models, while the biocomposite facade design showcases a potential application of wood dust crafted biocomposite facade panels within an aluminium-based curtain wall system.

How do you assess the value of transferability?

The transferability of this thesis is significant due to its use of Python for coding the SOM, KAN, MLP, and sensitivity analysis, and its use of Rhino 3D and Grasshopper for parametric modeling, performance simulations, and design exploration; ensuring ease of implementation and accesibility, as Python is a free open-source programming language with extensive libraries and active support, and Rhino 3D and Grasshopper are already widely implemented into architectural design. Furthermore, the SOM-KAN framework is fully scalable to higher-complex design problems and different datasets, making it adaptable to various applications within the AEC sector. Finally, the biocomposite design tool, set up in Excel, is able to adapt to more research as well.

How do you reflect on your way of working?

At the start of this thesis, my strategy was to choose a topic that would not reinvent the wheel entirely, as I know I tend to take on too much, but instead would allow me to explore a specific area in depth, which I believed would be more valuable to the research community as well as more manageable for myself. Looking back, I believe my thesis balances innovation while maintaining a focused approach quite effectively, though I did face some challenges in narrowing my thesis. Up until P2, I was torn between focusing on advancing the approach to early-stage facade design utilising computational and AI-driven tools and optimising the properties of wood dust crafted biocomposites at a material level, with a focus on robotic fabrication and its implementation into building facades. Initially, I thought these topics would be more closely related and could be combined, but I eventually decided to focus on advancing the facade design process, using biocomposites as a relevant and sustainable case study material, aiming to maintain a more focused approach.

How do you reflect on your mentor feedback?

As aforementioned, I had some difficulties narrowing down my thesis up until P2. With regards to overcoming this issue, a significant role was played by my mentors, who guided me towards a focused and relevant direction. Following P2, several feedback sessions took place which helped me fine-tune the SOM and maintain focus on the overall objective throughout the process, as well as enhance the applicability and validity of this thesis, by advising me to add a design exploration phase and validate its effectiveness in practise through a questionnaire.

References

[1] Abdallah, H. A., Abu-Jdayil, B., & Iqbal, M. Z. (2022). Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment. Journal of Cleaner Production, 346, 131242. https://doi.org/10.1016/j.jclepro.2022.131242

[2] Adelaja, O. A., Babaniyi, B. R., & Udorah, D. (2024). Improvement of Polypropylene (PP)-Chitosan Nanoparticles (CNP) for Advanced Bio-Composite. Advanced Energy Conversion Materials, 5(1), 117–132. https://doi.org/10.37256/aecm.5120244544

[3] Agbakoba, V. C., Hlangothi, P., Andrew, J., & John, M. J. (2023). Preparation of cellulose nanocrystal (CNCs) reinforced polylactic acid (PLA) bionanocomposites filaments using biobased additives for 3D printing applications. Nanoscale Advances, 5(17), 4447–4463. https://doi.org/10.1039/d3na00281k

[4] Al-Maharma, A. Y., & Al-Huniti, N. (2019). Critical review of the parameters affecting the effectiveness of moisture absorption treatments used for natural composites. Journal of Composites Science, 3(1), 27. https://doi.org/10.3390/jcs3010027

[5] Ali, A. (2019, May 26). Self Organizing Map(SOM) with Practical Implementation. Medium. https://miro.medium.com/v2/resize:fit:640/format:webp/1*0PdY0c_2FFZ1-BY-_j0yRA.png

[6] Altun, Y., Doğan, M., & Bayramlı, E. (2012). Effect of Alkaline Treatment and Pre-impregnation on Mechanical and Water Absorbtion Properties of Pine Wood Flour Containing Poly (Lactic Acid) Based Green-Composites. Journal of Polymers and the Environment, 21(3), 850–856. https://doi.org/10.1007/s10924-012-0563-x

[7] Aravind, T., Ashraf, M. S., S, R. A., Ahalya, N., Rawat, M. S., Uma, B., Sharma, R., Subbiah, R., & Sida, S. (2022). Study of Progress on Nanocrystalline Cellulose and Natural Fiber Reinforcement Biocomposites. Journal of Nanomaterials, 2022, 1–16. https://doi.org/10.1155/2022/6519480

[8] Ayrilmis, N., Yurttas, E., Durmus, A., Ozdemir, F., Nagarajan, R., Kalimuthu, M., & Kuzman, M. K. (2021). Properties of Biocomposite Films From PLA and Thermally Treated Wood Modified with Silver Nanoparticles Using Leaf Extracts of Oriental Sweetgum. Journal of Polymers and the Environment, 29, 2409–2420. https://doi.org/10.1007/s10924-021-02065-x

[9] Azka, M. A., Sapuan, S., Abral, H., Zainudin, E., & Aziz, F. A. (2024). An examination of recent research of water absorption behavior of natural fiber reinforced polylactic acid (PLA) composites: A review. International Journal of Biological Macromolecules, 131845. https://doi.org/10.1016/j.ijbiomac.2024.131845

[10] Bahar, A., Hamami, A. E. A., Benmahiddine, F., Belhabib, S., Belarbi, R., & Guessasma, S. (2023). The Thermal and Mechanical Behaviour of Wood-PLA Composites Processed by Additive Manufacturing for Building Insulation. Polymers, 15(14), 3056. https://doi.org/10.3390/polym15143056

[11] Bahrami, M., Abenojar, J., & Martínez, M. Á. (2020). Recent progress in hybrid biocomposites: mechanical properties, water absorption, and flame retardancy. Materials, 13(22), 5145. https://doi.org/10.3390/ma13225145

[12] Bahl, S., Bagha, A. K., & Sehgal, S. (2021). Experimental investigations into sound transmission loss by different materials at aircraft noise. Materials Today: Proceedings, 44, 2048–2053. https://doi.org/10.1016/j.matpr.2020.12.153

[13] Balmer, V., Kuhn, S. V., Bischof, R., Salamanca, L., Kaufmann, W., Perez-Cruz, F., & Kraus, M. A. (2024). Design space exploration and explanation via conditional variational autoencoders in Meta-Model-Based conceptual design of pedestrian bridges. Automation in Construction, 163, 105411. https://doi.org/10.1016/j.autcon.2024.105411

[14] Barkhad, M. S., Abu-Jdayil, B., Mourad, A. H. I., & Iqbal, M. Z. (2020). Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions. Polymers, 12(9), 2091. https://doi.org/10.3390/polym12092091

[15] Baştanlar, Y., & Özuysal, M. (2013). Introduction to machine learning. In Methods in molecular biology (pp. 105–128). https://doi.org/10.1007/978-1-62703-748-8_7

[16] Berisha, V., Krantsevich, C., Hahn, P. R., Hahn, S., Dasarathy, G., Turaga, P., & Liss, J. (2021). Digital medicine and the curse of dimensionality. Npj Digital Medicine, 4(1). https://doi.org/10.1038/s41746-021-00521-5

[17] Bertagna, F., D'Acunto, P., Ohlbrock, P. O., & Moosavi, V. (2021). Holistic design explorations of building envelopes supported by machine learning. Journal of Facade Design and Engineering, 9(1), 31–46. https://doi.org/10.7480/jfde.2021.1.5423

[18] Bethell, D. (2024, May 13). Demystifying Kolmogorov-Arnold Networks: A Beginner-Friendly Guide with Code. Daniel-Bethell. https://i.imgur.com/v8D90ml.gif

[19] Boarino, A., Schreier, A., Leterrier, Y., & Klok, H. (2022). Uniformly Dispersed Poly(lactic acid)-Grafted Lignin Nanoparticles Enhance Antioxidant Activity and UV-Barrier Properties of Poly(lactic acid) Packaging Films. ACS Applied Polymer Materials, 4(7), 4808–4817. https://doi.org/10.1021/acsapm.2c00420

[20] Boruvka, M., & Prusek, J. (2016). Green nanocomposites based on lignin coated cellulose nanocrystals and poly(lactic acid): crystallization, mechanical and thermal properties. NANOCON. https://www.confer.cz/nanocon/2016/437-green-nanocomposites-based-on-lignin-coated-cellulose-nanocrystals-and-poly-lactic-acid-crystallization-mechanical-and-thermal-properties

[21] Bozorgasl, Z., & Chen, H. (2024). Wav-KAN: Wavelet Kolmogorov-Arnold Networks. In arXivLabs. https://arxiv.org/abs/2405.12832

[22] Brown, N., De Oliveira, J. I. F., Ochsendorf, J., Mueller, C., Massachusetts Institute of Technology, & Universidade Federal Rural do Semi-Árido. (2016). Early-stage integration of architectural and structural performance in a parametric multi-objective design tool. In Massachusetts Institute of Technology [Journal-article]. http://digitalstructures.mit.edu/files/2016-10/icsa-ncb-final-2-2.pdf

[23] Business Waste. (2024, February 14). Wood Recycling Facts | Wood Waste Statistics | Business Waste. https://www.businesswaste.co.uk/your-waste/waste-wood-collection/wood-recycling-facts-and-statistics/

[24] Cavallo, E., He, X., Luzi, F., Dominici, F., Cerrutti, P., Bernal, C., Foresti, M. L., Torre, L., & Puglia, D. (2020). UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles. Molecules/Molecules Online/Molecules Annual, 26(1), 126. https://doi.org/10.3390/molecules26010126

[25] Chi, K., & Catchmark, J. M. (2017). Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant adsorption. Cellulose, 24, 4845–4860. https://doi.org/10.1007/s10570-017-1479-3

[26] Chong, W. J., Simunec, D. P., Trinchi, A., Kyratzis, I., Li, Y., Wright, P., Shen, S., Sola, A., & Wen, C. (2023). Advancing the additive manufacturing of PLA-ZnO nanocomposites by fused filament fabrication. Virtual and Physical Prototyping, 19(1). https://doi.org/10.1080/17452759.2023.2285418

[27] Choo, T.-S., & Janssen, P. (2015). PERFORMANCE-BASED PARAMETRIC DESIGN. In Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (pp. 000–000) [Conference-proceeding]. CAADRIA. https://papers.cumincad.org/data/works/att/caadria2015_172.content.pdf

[28] Chow, A. (2020, December 26). Artificial Neural Network. Medium. https://miro.medium.com/v2/resize:fit:720/format:webp/1*lnhJA-8vzxdniOTqygcdX2Q.png

[29] Csizmadia, R., Faludi, G., Renner, K., Móczó, J., & Pukánszky, B. (2013). PLA/wood biocomposites: Improving composite strength by chemical treatment of the fibers. Composites. Part a, Applied Science and Manufacturing, 53, 46–53. https://doi.org/10.1016/j.compositesa.2013.06.003

[30] Daassi, R., Durand, K., Rodrigue, D., & Stevanovic, T. (2023). Optimization of the Electrospray Process to Produce Lignin Nanoparticles for PLA-Based Food Packaging. Polymers, 15(13), 2973. https://doi.org/10.3390/polym15132973

[31] Dai, D., & Fan, M. (2014). Wood fibres as reinforcements in natural fibre composites: structure, properties, processing and applications. In Elsevier eBooks (pp. 3–65). https://doi.org/10.1533/9780857099228.1.3

[32] Danhaive, R., & Mueller, C. T. (2021). Design subspace learning: Structural design space exploration using performance-conditioned generative modeling. Automation in Construction, 127, 103664. https://doi.org/10.1016/j.autcon.2021.103664

[33] Dantuma, E., & Van Sante, M. (2018). De circulaire corporatie naar volledig duurzame huisvesting. https://fd-binary-external-prod.imgix.net/dOoJIsbH3U5PB3dPRM2JgYKKkDE.pdf?dl=De+circulaire+corporatie.+Naar+volledig+duurzame+huisvesting.pdf

[34] Darko, A., Chan, A. P., Adabre, M. A., Edwards, D. J., Hosseini, M. R., & Ameyaw, E. E. (2020). Artificial intelligence in the AEC industry: Scientometric analysis and visualization of research activities. Automation in Construction, 112, 103081. https://doi.org/10.1016/j.autcon.2020.103081

[35] Dhal, M. K., Madhu, K., Banerjee, A., Prasannavenkadesan, V., Kumar, A., & Katiyar, V. (2023). Polylactic acid/polycaprolactone/sawdust based biocomposites trays with enhanced compostability. International Journal of Biological Macromolecules, 253, 126977. https://doi.org/10.1016/j.ijbiomac.2023.126977

[36] Ellen MacArthur Foundation. (2019, February). Circular economy systems diagram. https://images.ctfassets.net/isq5xwjfoz2m/3v3SXaPbb5iKAFF4sSMClk/920fc7076d12cf6c2b4478c2d91dfbdc/Butterfly_Diagram.webp

[37] Ellen MacArthur Foundation. (n.d.). The circular economy in detail. https://www.ellenmacarthurfoundation.org/the-circular-economy-in-detail-deep-dive

[38] European Circular Economy Stakeholder Platform (ECESP). (2021). Circular buildings and infrastructure. In European Circular Economy Stakeholder Platform. https://circulareconomy.europa.eu/platform/sites/default/files/circular_buildings_and_infrastructure_brochure.pdf

[39] Fortunati, E., Luzi, F., Puglia, D., Petrucci, R., Kenny, J., & Torre, L. (2015). Processing of PLA nanocomposites with cellulose nanocrystals extracted from Posidonia oceanica waste: Innovative reuse of coastal plant. Industrial Crops and Products, 67, 439–447. https://doi.org/10.1016/j.indcrop.2015.01.075

[40] Fortunati, E., Peltzer, M., Armentano, I., Jiménez, A., & Kenny, J. (2013). Combined effects of cellulose nanocrystals and silver nanoparticles on the barrier and migration properties of PLA nano-biocomposites. Journal of Food Engineering, 118(1), 117–124. https://doi.org/10.1016/j.jfoodeng.2013.03.025

[41] Fortunati, E., Peltzer, M., Armentano, I., Torre, L., Jiménez, A., & Kenny, J. (2012). Effects of modified cellulose nanocrystals on the barrier and migration properties of PLA nano-biocomposites. Carbohydrate Polymers, 90(2), 948–956. https://doi.org/10.1016/j.carbpol.2012.06.025

[42] Gandini, A., & Belgacem, M. (2011). Modifying cellulose fiber surfaces in the manufacture of natural fiber composites. In Elsevier eBooks (pp. 3–42). https://doi.org/10.1533/9780857092281.1.3

[43] Gervasio, H., & Dimova, S. (2018). Environmental benchmarks for buildings. In JRC Publications Repository. https://doi.org/10.2760/073513

[44] González-López, M., Del Campo, A., Robledo-Ortíz, J, Arellano, M., & Pérez-Fonseca, A. (2020). Accelerated weathering of poly(lactic acid) and its biocomposites: A review. Polymer Degradation and Stability, 179, 109290. https://doi.org/10.1016/j.polymdegradstab.2020.109290

[45] Government of the Netherlands. (2023). Circular Economy Programme 2023-2030. In Government of the Netherlands. https://www.rijksoverheid.nl/onderwerpen/circulaire-economie/documenten/beleidsnotas/2023/02/03/nationaal-programma-circulaire-economie-2023-2030

[46] Gregorova, A., Hrabalova, M., Kovalcik, R., & Wimmer, R. (2010). Surface modification of spruce wood flour and effects on the dynamic fragility of PLA/wood composites. SPE Transactions/Polymer Engineering and Science, 51(1), 143–150. https://doi.org/10.1002/pen.21799

[47] Guercia, G., & Cannon, K. (2018). Classification + Reference Standards for UHPC in Architectural Applications. In ResearchGate. https://www.researchgate.net/publication/327262901_Classification_Reference_Standards_for_UHPC_in_Architectural_Applications

[48] Hamdan, S., Islam, S., & Rusop, M. (2012). Dimensional stability and water repellent efficiency measurement of chemically modified tropical light hardwood. BioResources, 7(1), 1221–1231. https://doi.org/10.15376/biores.7.1.1221-1231

[49] Islam, S., Bhat, G., & Sikdar, P. (2023). Thermal and acoustic performance evaluation of 3D-Printable PLA materials. Journal of Building Engineering, 67, 105979. https://doi.org/10.1016/j.jobe.2023.105979

[50] Jamnongkan, T., Jaroensuk, O., Khankhuean, A., Laobuthee, A., Srisawat, N., Pangon, A., Mongkholrattanasit, R., Phuengphai, P., Wattanakornsiri, A., & Huang, C. (2022). A Comprehensive Evaluation of Mechanical, Thermal, and Antibacterial Properties of PLA/ZnO Nanoflower Biocomposite Filaments for 3D Printing Application. Polymers, 14(3), 600. https://doi.org/10.3390/polym14030600

[51] Jasiński, W., & Szymanowski, K. (2023). Selected flexural and hygroscopic properties of waste wood dust - polylactic acid biocomposite for 3D printing. Annals of Warsaw University of Life Sciences SGGW. Forestry and Wood Technology/Annals of Warsaw University of Life Sciences. Forestry and Wood Technology, 121, 11–20. https://doi.org/10.5604/01.3001.0053.8562

[52] Jawaid, M., & Khalil, H. A. (2011). Cellulosic/synthetic fibre reinforced polymer hybrid composites: A review. Carbohydrate Polymers, 86(1), 1–18. https://doi.org/10.1016/j.carbpol.2011.04.043

[53] Jayamani, E., Hamdan, S., Rahman, M. R., Bakri, M. K. B., & Kakar, A. (2015). An investigation of sound absorption coefficient on sisal fiber poly lactic acid bio-composites. Journal of Applied Polymer Science, 132(34). https://doi.org/10.1002/app.42470

[54] Karkhanis, S. S., Stark, N. M., Sabo, R. C., & Matuana, L. M. (2018). Water vapor and oxygen barrier properties of extrusion-blown poly(lactic acid)/cellulose nanocrystals nanocomposite films. Composites. Part a, Applied Science and Manufacturing, 114, 204–211. https://doi.org/10.1016/j.compositesa.2018.08.025

[55] Kelleci, O., Aydemir, D., Altuntas, E., Kurt, R., Oztel, A., Yorur, H., & Istek, A. (2022). Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods. Journal of Polymers and the Environment, 30, 3091–3106. https://doi.org/10.1007/s10924-022-02415-3

[56] Kelnar, I., Kaprálková, L., Krejčíková, S., Dybal, J., Vyroubalová, M., & Abdel-Mohsen, A. M. (2023). Effect of Polydopamine Coating of Cellulose Nanocrystals on Performance of PCL/PLA Bio-Nanocomposites. Materials, 16(3), 1087. https://doi.org/10.3390/ma16031087

[57] Kim, I., Viswanathan, K., Kasi, G., Sadeghi, K., Thanakkasaranee, S., & Seo, J. (2019). Poly(Lactic Acid)/ZnO Bionanocomposite Films with Positively Charged ZnO as Potential Antimicrobial Food Packaging Materials. Polymers, 11(9), 1427. https://doi.org/10.3390/polym11091427

[58] Kohonen, T. (2013). Essentials of the self-organizing map. Neural Networks, 37, 52–65. https://doi.org/10.1016/j.neunet.2012.09.018

[59] Kurpińska, M., Pawelska-Mazur, M., Gu, Y., & Kurpiński, F. (2022). The impact of natural fibers' characteristics on mechanical properties of the cement composites. Scientific Reports, 12(1). https://doi.org/10.1038/s41598-022-25085-6

[60] Leising, E., Quist, J., & Bocken, N. (2018). Circular Economy in the building sector: Three cases and a collaboration tool. Journal of Cleaner Production, 176, 976–989. https://doi.org/10.1016/j.jclepro.2017.12.010

[61] Lendvai, L., Omastova, M., Patnaik, A., Dogossy, G., & Singh, T. (2022). Valorization of Waste Wood Flour and Rice Husk in Poly(Lactic Acid)-Based Hybrid Biocomposites. Journal of Polymers and the Environment, 31(2), 541–551. https://doi.org/10.1007/s10924-022-02633-9

[62] Lertwattanaruk, P., & Suntijitto, A. (2015). Properties of natural fiber cement materials containing coconut coir and oil palm fibers for residential building applications. Construction & Building Materials, 94, 664–669. https://doi.org/10.1016/j.conbuildmat.2015.07.154

[63] Li, M., Zhang, Y., Li, L., Xie, Y., Yao, W., Chen, Q., & Ju, J. (2023). Characterization of an eco-friendly active packaging film for food with ultraviolet light blocking ability. Food Science and Technology, 43. https://doi.org/10.5327/fst.16723

[64] Li, Z., Guo, T., Chen, Y., Yang, W., Wang, J., & Jin, L. (2024). Preparation and properties of pretreated jute fiber cement-based composites. Industrial Crops and Products, 210, 118090. https://doi.org/10.1016/j.indcrop.2024.118090

[65] Liang, J., & Jiang, X. (2012). Soundproofing effect of polypropylene/inorganic particle composites. Composites. Part B, Engineering, 43(4), 1995–1998. https://doi.org/10.1016/j.compositesb.2012.02.020

[66] Licen, S., Astel, A., & Tsakovski, S. (2023). Self-organizing map algorithm for assessing spatial and temporal patterns of pollutants in environmental compartments: A review. Science of the Total Environment, 878, 163084. https://doi.org/10.1016/j.scitotenv.2023.163084

[67] Ling, Z., Chen, J., Wang, X., Shao, L., Wang, C., Chen, S., Guo, J., & Yong, Q. (2022). Nature-inspired construction of iridescent CNC/Nano-lignin films for UV resistance and ultra-fast humidity response. Carbohydrate Polymers, 296, 119920. https://doi.org/10.1016/j.carbpol.2022.119920

[68] Liu, N., Li, Y., Xie, D., Zeng, Q., Xu, H., Ge, M., Zhang, W., Zhang, Y., Liu, R., Dai, J., Zhang, H., & Li, S. (2023). UV stabilizer intercalated layered double hydroxide to enhance the thermal and UV degradation resistance of polypropylene fiber. Polymer Testing, 121, 107979. https://doi.org/10.1016/j.polymertesting.2023.107979

[69] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljacic, M., Hou, T. Y., & Tegmark, M. (2024). KAN: Kolmogorov–Arnold Networks. In arXivLabs. https://arxiv.org/abs/2404.19756

[70] Luzi, F., Fortunati, E., Jiménez, A., Puglia, D., Chiralt, A., & Torre, L. (2016). PLA Nanocomposites Reinforced with Cellulose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application. Journal of Renewable Materials, 5(2), 103–115. https://doi.org/10.7569/jrm.2016.634135

[71] Maadi, M., Khorshidi, H. A., & Aickelin, U. (2021). A review on Human–AI interaction in Machine Learning and insights for medical applications. International Journal of Environmental Research and Public Health/International Journal of Environmental Research and Public Health, 18(4), 2121. https://doi.org/10.3390/ijerph18042121

[72] Mármol, G., Gauss, C., & Fangueiro, R. (2020). Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement. Molecules/Molecules Online/Molecules Annual, 25(20), 4653. https://doi.org/10.3390/molecules25204653

[73] Mathew, A. P., Oksman, K., & Sain, M. (2005). Mechanical properties of biodegradable composites from poly lactic acid (PLA) and microcrystalline cellulose (MCC). Journal of Applied Polymer Science, 97(5), 2014–2025. https://doi.org/10.1002/app.21779

[74] Miljkovic, D. (Ed.). (2017). Brief Review of Self-Organizing Maps [MIPRO]. https://doi.org/10.23919/MIPRO.2017.7973581

[75] Minaei, M., & Aksamija, A. (2020). Performance-Based Facade Framework Automated and Multi-Objective Simulation and Optimization. In SimAUD 2020. https://simaud.org/2020/proceedings/95.pdf

[76] Mohammed, M., Rahman, R., Mohammed, A. M., Adam, T., Betar, B. O., Osman, A. F., & Dahham, O. S. (2022). Surface treatment to improve water repellence and compatibility of natural fiber with polymer matrix: Recent advancement. Polymer Testing, 115, 107707. https://doi.org/10.1016/j.polymertesting.2022.107707

[77] Mohammadsalih, Z. G., Muawwidzah, M., Siddiqui, V. U., & Sapuan, S. M. (2023). Mechanical Properties of Wood Fibre Filled Polylactic Acid (PLA) Composites Using Additive Manufacturing Techniques. Journal of Natural Fibre Polymer Composites (JNFPC), 2(2), 2821–3289. https://www.researchgate.net/publication/376885815_Mechanical_Properties_of_Wood_Fibre_Filled_Polylactic_Acid_PLA_Composites_Using_Additive_Manufacturing_Techniques

[78] Muzata, T. S., Gebrekrstos, A., Orasugh, J. T., & Ray, S. S. (2023). An overview of recent advances in polymer composites with improved UV-shielding properties. Journal of Applied Polymer Science, 140(14). https://doi.org/10.1002/app.53693

[79] Narlioglu, N., Salan, T., & Mehmet, A. (2021). Properties of 3D-Printed Wood Sawdust-Reinforced PLA Composites. Bioresources, 16(3), 5467–5480. https://doi.org/10.15376/biores.16.3.5467-5480

[80] Overend, M., Hartwell, R., & Macmillan, S. (2021). Circular economy of façades: Real-world challenges and opportunities. Resources, Conservation and Recycling, 175, 105827. https://doi.org/10.1016/j.resconrec.2021.105827

[81] Rafsanjani, H. N., & Nabizadeh, A. H. (2023). Towards human-centered artificial intelligence (AI) in architecture, engineering, and construction (AEC) industry. Computers in Human Behavior Reports, 11, 100319. https://doi.org/10.1016/j.chbr.2023.100319

[82] Samadi, M. E., Muller, Y., & Schuppert, A. (2024). Smooth Kolmogorov Arnold networks enabling structural knowledge representation. In arXivLabs. https://arxiv.org/abs/2405.11318

[83] Samek, W., Montavon, G., Lapuschkin, S., Anders, C., & Muller, K.-R. (2021). Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. Proceedings of the IEEE, 109(3), 247–278. https://doi.org/10.1109/JPROC.2021.3060483

[84] Shawia, N., Jabber, M., & Mamouri, A. (2014). Mechanical and physical properties of natural fiber cement board for building partitions. Physical Sciences Research International, 2(3), 49–53. https://www.netjournals.org/z_PSRI_14_021.html

[85] Shojaeiarani, J., Bajwa, D. S., Ryan, C., & Kane, S. (2022). Enhancing UV-shielding and mechanical properties of polylactic acid nanocomposites by adding lignin coated cellulose nanocrystals. Industrial Crops and Products, 183, 114904. https://doi.org/10.1016/j.indcrop.2022.114904

[86] Silva, G., D'Almeida, J. R. M., & Cardoso, D. C. T. (2024). Investigation on moisture absorption behavior on GFRP and neat epoxy systems in hygrothermal salt fog aging. Composites. Part B, Engineering, 272, 111214. https://doi.org/10.1016/j.compositesb.2024.111214

[87] Singh, T., Pattnaik, P., Aherwar, A., Ranakoti, L., Dogossy, G., & Lendvai, L. (2022). Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC–MABAC-Based Decision-Making Algorithm. Polymers, 14(13), 2603. https://doi.org/10.3390/polym14132603

[88] Sringam, S., Pornbencha, K., Piyanirund, S., Seubsai, A., Prapainainar, P., Niumnuy, C., Roddecha, S., & Dittanet, P. (2023). Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid). RSC Advances, 13(22), 15311–15321. https://doi.org/10.1039/d3ra02693k

[89] Sun, Z., Luo, C., Wang, Q., Wang, X., & Mu, J. (2023). Preparation of ODA-TA-L-CNC and its effect on multifunctional PLA composite films. China Plastics, 37(12), 14–22. https://doi.org/10.19491/j.issn.1001-9278.2023.12.003

[90] Tan, M. A., Yeoh, C. K., Teh, P. L., Rahim, N. a. A., Song, C. C., & Voon, C. H. (2023). Effect of zinc oxide suspension on the overall filler content of the PLA/ZnO composites and cPLA/ZnO composites. E-Polymers, 23(1). https://doi.org/10.1515/epoly-2022-8113

[91] Thakur, V. K. (2014). Processing and characterization of natural cellulose fibers/thermoset polymer composites. Carbohydrate Polymers, 109, 102–117. https://doi.org/10.1016/j.carbpol.2014.03.039

[92] Thepruttana, S., Patthanavarit, J., Hankoy, M., Kitiwan, M., Keawprak, N., & Tunthawiroon, P. (2024). Enhancement of Flexural Strength in Fiber–Cement Composites through Modification of Sisal Fiber with Natural Rubber Latex and Expanded Perlite. Buildings, 14(4), 1067. https://doi.org/10.3390/buildings14041067

[93] Turrin, M., Pan, W., Sun, Y., Louter, C., & Sariyildiz, S. (2020). Design exploration of quantitative performance and geometry typology for indoor arena based on self-organizing map and multi-layered perceptron neural network. Automation in Construction, 114, 103163. https://doi.org/10.1016/j.autcon.2020.103163

[94] Turrin, M., Pan, W., Sun, Y., Louter, C., & Sariyildiz, S. (2020, March 5). Design exploration of quantitative performance and geometry typology for indoor arena based on self-organizing map and multi-layered perceptron neural network. ScienceDirect. https://ars.els-cdn.com/content/image/1-s2.0-S0926580519309008-gr14_lrg.jpg

[95] Vazquez, A. N., & Walker, M. (2021). Balancing Design and Performance: Multi-objective optimization for the generation of high-performance facade panels for retrofitting. ResearchGate. https://www.researchgate.net/publication/360658591_Balancing_Design_and_Performance_Multi-objective_optimization_for_the_generation_of_high-performance_facade_panels_for_retrofitting

[96] Viscovery. (n.d.). Self-Organizing Map (SOM) feature map topographical surface plot. https://www.viscovery.net/bilder/somne/UnfoldedMap.png

[97] Wang, Y. Y., Yu, H., Yang, L., Abdalkarim, S. Y. H., & Chen, W. (2019). Enhancing long-term biodegradability and UV-shielding performances of transparent polylactic acid nanocomposite films by adding cellulose nanocrystal-zinc oxide hybrids. International Journal of Biological Macromolecules, 141, 893–905. https://doi.org/10.1016/j.ijbiomac.2019.09.062

[98] Wei, J., & Meyer, C. (2015). Degradation mechanisms of natural fiber in the matrix of cement composites. Cement and Concrete Research, 73, 1–16. https://doi.org/10.1016/j.cemconres.2015.02.019

[99] Wei, L., Agarwal, U. P., Matuana, L., Sabo, R. C., & Stark, N. M. (2017). Performance of high lignin content cellulose nanocrystals in poly(lactic acid). Polymer, 135, 305–313. https://doi.org/10.1016/j.polymer.2017.12.039

[100] Wei, L., Stark, N. M., Sabo, R. C., & Matuana, L. (2016). Modification of cellulose nanocrystals (CNCs) for use in poly(lactic acid) (PLA)-CNC composite packaging products. US Forest Service Research and Development. https://www.fs.usda.gov/research/treesearch/52859

[101] Xu, Y., Li, F., & Asgari, A. (2022). Prediction and optimization of heating and cooling loads in a residential building based on multi-layer perceptron neural network and different optimization algorithms. Energy, 240, 122692. https://doi.org/10.1016/j.energy.2021.122692

[102] Yang, S., Qin, L., & Yu, X. (2024). Endowing Interpretability for Neural Cognitive Diagnosis by Efficient Kolmogorov-Arnold Networks. In arXivLabs. https://arxiv.org/abs/2405.14399

[103] Yang, W., Fortunati, E., Dominici, F., Kenny, J., & Puglia, D. (2015). Effect of processing conditions and lignin content on thermal, mechanical and degradative behavior of lignin nanoparticles/polylactic (acid) bionanocomposites prepared by melt extrusion and solvent casting. European Polymer Journal/European Polymer Journal, 71, 126–139. https://doi.org/10.1016/j.eurpolymj.2015.07.051

[104] Yang, X., Yin, G., Amorós, O. Z., Hernández, M. A., De La Vega, J., & Torralba, J. M. (2023). Polydopamine surface functionalized submicron ZnO for broadening the processing window of 3D printable PLA composites. Journal of Polymer Research, 30(5). https://doi.org/10.1007/s10965-023-03540-w

[105] Zhao, J., Zong, Z., Cen, H., & Jiang, P. (2024). Analysis of Mechanical Properties of Fiber-Reinforced Soil Cement Based on Kaolin. Materials, 17(9), 2153. https://doi.org/10.3390/ma17092153

[106] Zwawi, M. (2021). A review on Natural Fiber Bio-Composites, Surface Modifications and applications. Molecules/Molecules Online/Molecules Annual, 26(2), 404. https://doi.org/10.3390/molecules26020404

A

Biocomposites

This appendix is optional to read and covers the results of the research into biocomposites. With regards to these results, Figures A.1 to A.40, A.41 to A.119, A.120 to A.125, and A.126 to A.130 cover the effect of (un)treated wood dust content, nanoparticle content, annealing time, and fabrication method on neat PLA's performance respectively, giving insights into exact values. Figures A.131 to A.182 and A.183 to A.226 cover the effect of (un)-treated wood dust content and nanoparticle content on the performance of neat PLA respectively, giving insights into the relative change of performance. Figures A.227 to A.236 and A.237 to A.242 cover the combined effect of various PLA fillers and nanoparticles respectively, giving insights into the relative change of performance. Figures A.243 to A.268 cover all the performance overview and performance score graphs, as part of the design tool.

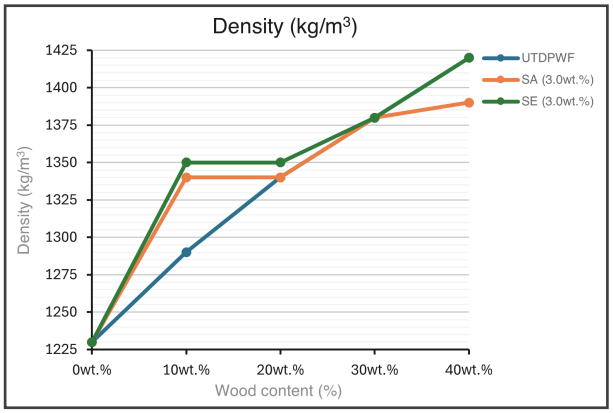


Figure A.1. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the density. Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

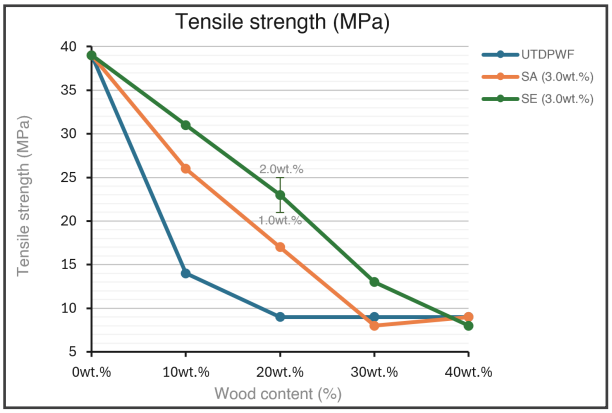


Figure A.3. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the tensile strength. Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

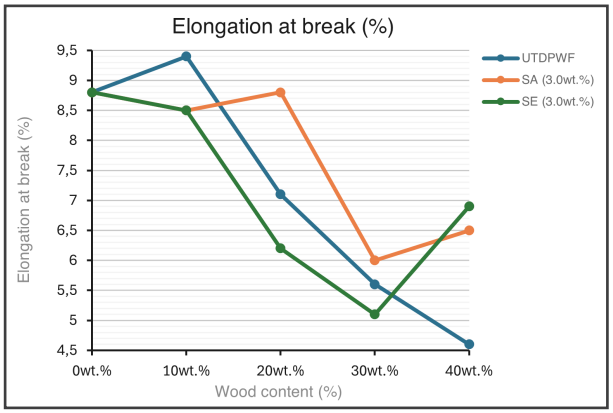


Figure A.2. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the elongation at break. Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

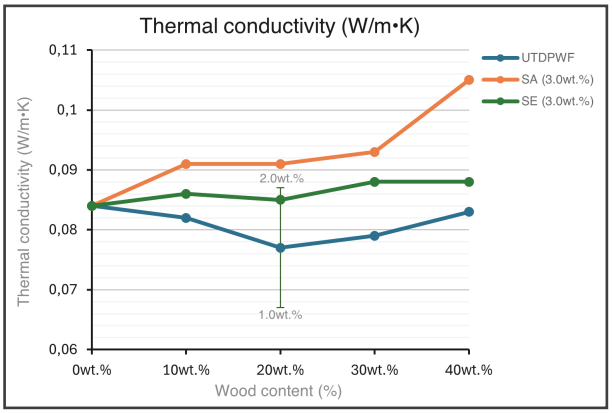


Figure A.4. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the thermal conductivity. Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

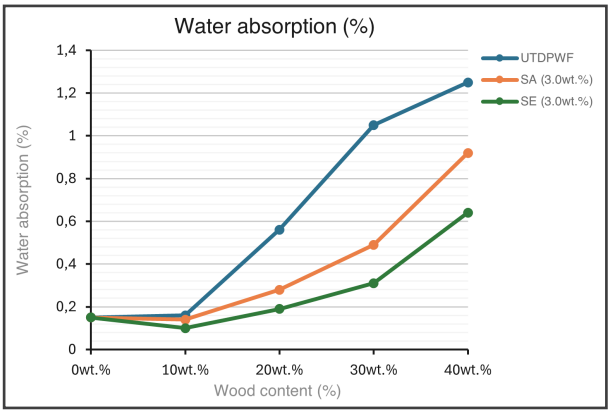


Figure A.5. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the water absorption (25°C/24h). Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

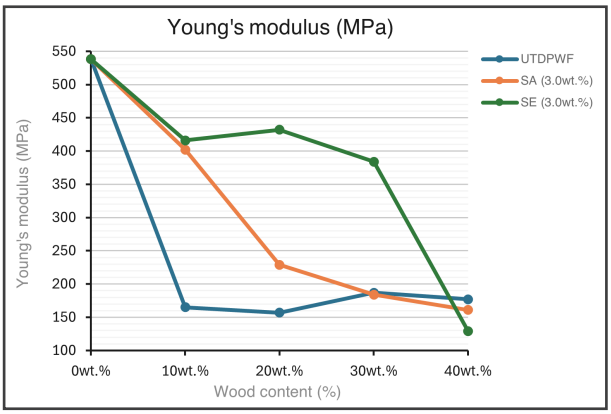


Figure A.6. Effect of wood content (blue: untreated date palm wood fibers {UTDPWF}, orange: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, and green: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio) in PLA matrix on the Young's modulus. Adapted from "*Improvement of mechanical properties and water resistance of bio-based thermal insulation material via silane treatment*", by Abdallah et al., 2022 [1].

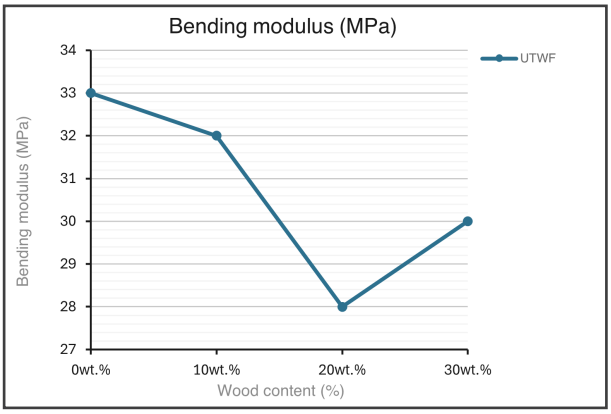


Figure A.7. Effect of untreated waste wood flour (UTWF) content in PLA matrix on the bending modulus. Adapted from "*Selected flexural and hygroscopic properties of waste wood dust - polyactic acid biocomposite for 3D printing*", by Jasinski & Szymanowski, 2023 [51].

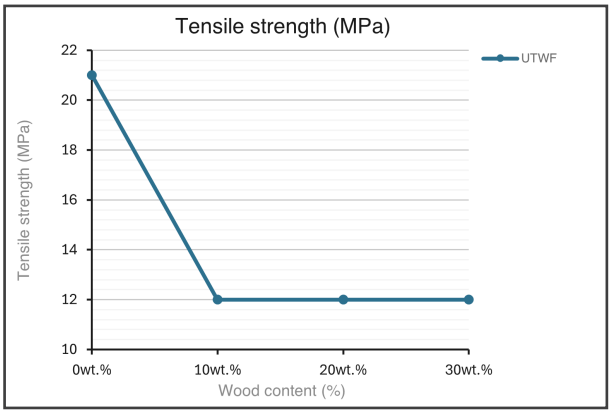


Figure A.8. Effect of untreated waste wood flour (UTWF) content in PLA matrix on the tensile strength. Adapted from "*Selected flexural and hygroscopic properties of waste wood dust - polyactic acid biocomposite for 3D printing*", by Jasinski & Szymanowski, 2023 [51].

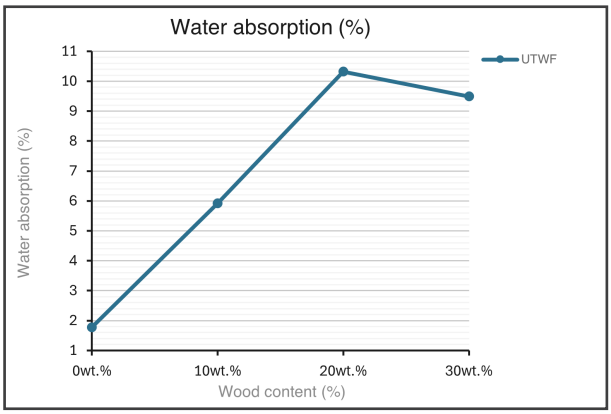


Figure A.9. Effect of untreated waste wood flour (UTWF) content in PLA matrix on the water absorption (25°C / 24h). Adapted from "*Selected flexural and hygroscopic properties of waste wood dust - polyactic acid biocomposite for 3D printing*", by Jasinski & Szymanowski, 2023 [51].

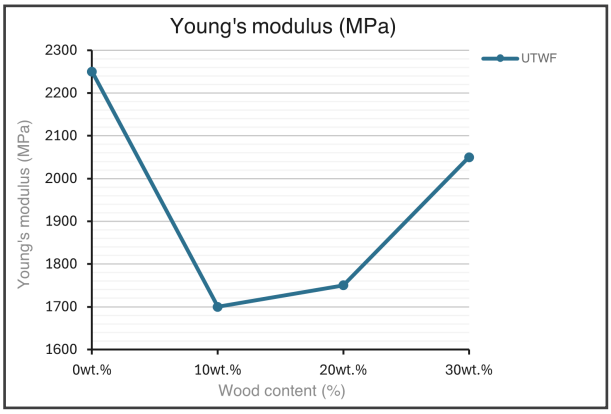


Figure A.10. Effect of untreated waste wood flour (UTWF) content in PLA matrix on the Young's modulus. Adapted from "*Selected flexural and hygroscopic properties of waste wood dust - polyactic acid biocomposite for 3D printing*", by Jasinski & Szymanowski, 2023 [51].

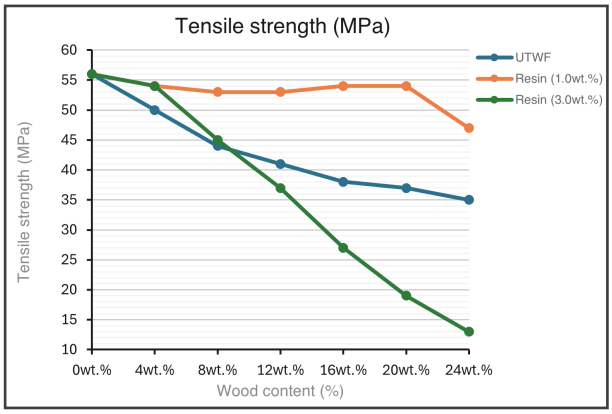


Figure A.11. Effect of wood content (blue: untreated wood flour {UTWF}, orange: 1wt.% phenolic resin-treated wood flour, and green: 3wt.% phenolic resin-treated wood flour) in PLA matrix on the tensile strength. Adapted from "*PLA/wood biocomposites: Improving composite strength by chemical treatment of the fibers*", by Csizmadia et al., 2013 [29].

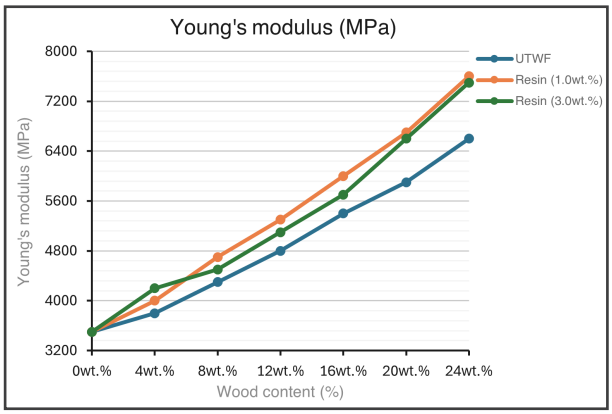


Figure A.12. Effect of wood content (blue: untreated wood flour {UTWF}, orange: 1wt.% phenolic resin-treated wood flour, and green: 3wt.% phenolic resin-treated wood flour) in PLA matrix on the Young's modulus. Adapted from "*PLA/wood biocomposites: Improving composite strength by chemical treatment of the fibers*", by Csizmadia et al., 2013 [29].

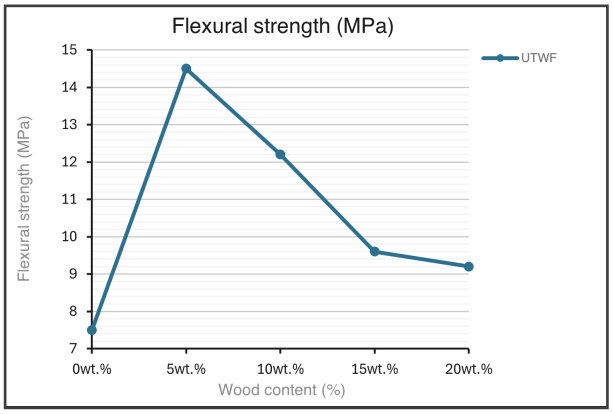


Figure A.13. Effect of untreated waste pine sawdust (UTWF) content in PLA matrix on the flexural strength. Adapted from "*Properties of 3D-Printed Wood Sawdust-Reinforced PLA Composites*", by Narlioglu et al., 2021 [79].

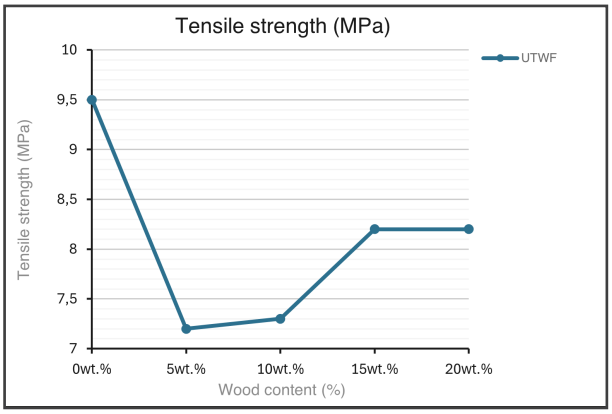


Figure A.14. Effect of untreated waste pine sawdust (UTWF) content in PLA matrix on the tensile strength. Adapted from "*Properties of 3D-Printed Wood Sawdust-Reinforced PLA Composites*", by Narlioglu et al., 2021 [79].

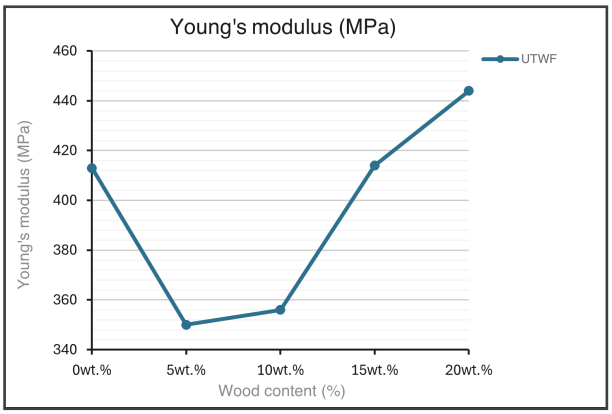


Figure A.15. Effect of untreated waste pine sawdust (UTWF) content in PLA matrix on the Young's modulus. Adapted from "*Properties of 3D-Printed Wood Sawdust-Reinforced PLA Composites*", by Narlioglu et al., 2021 [79].

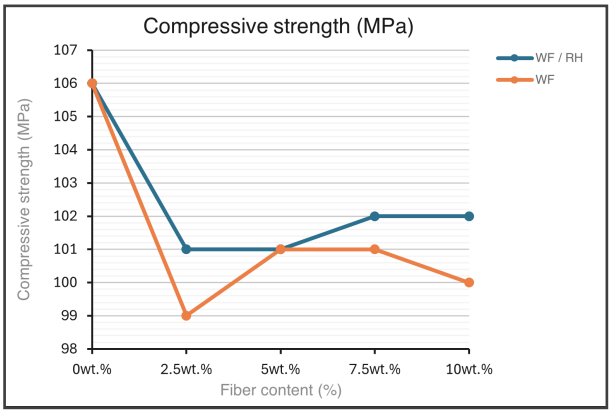


Figure A.16. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the compressive strength. Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

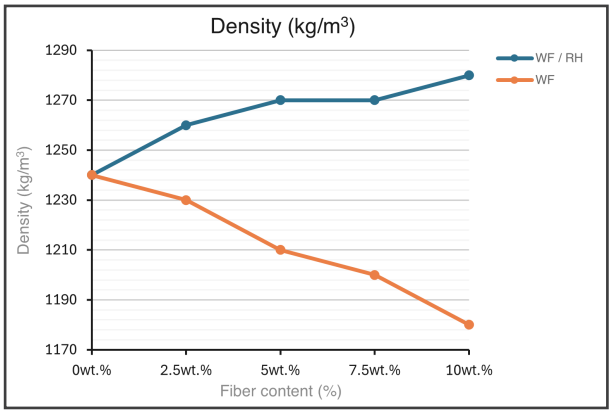


Figure A.17. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the density. Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

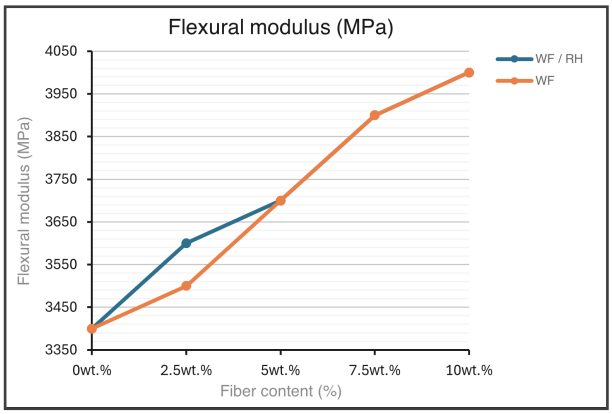


Figure A.18. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the flexural modulus. Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

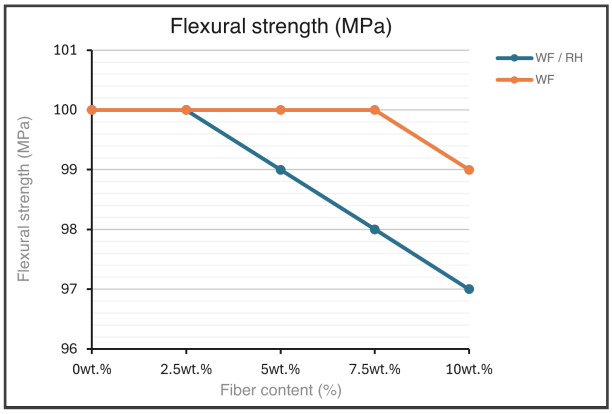


Figure A.19. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the flexural strength. Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

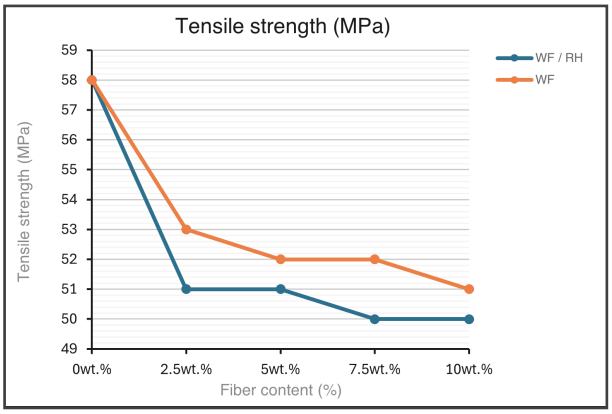


Figure A.20. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the tensile strength. Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

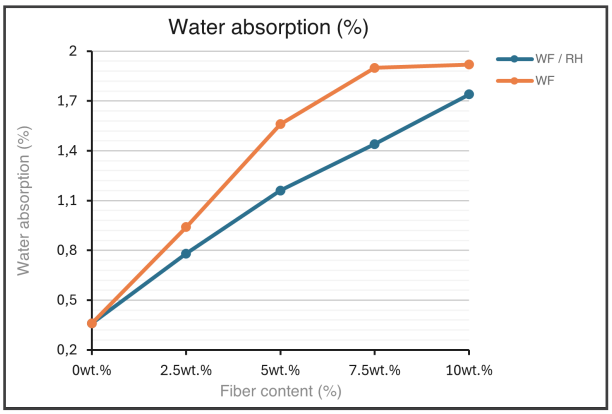


Figure A.21. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the water absorption (23°C / 120h). Adapted from "*Optimal Design of Wood/Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

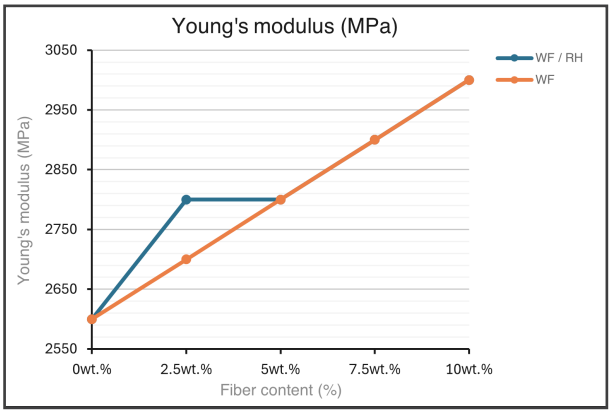


Figure A.22. Effect of fiber content (blue: 2wt.% Alkaline-treated wood flour/ rice husk in 50:50wt.% ratio, and orange: 2wt.% Alkaline-treated wood flour) in PLA matrix on the Young's modulus. Adapted from "*Optimal Design of Wood/ Rice Husk-Waste-Filled PLA Biocomposites Using Integrated CRITIC-MABAC-Based Decision-Making Algorithm*", by Singh et al., 2022 [87].

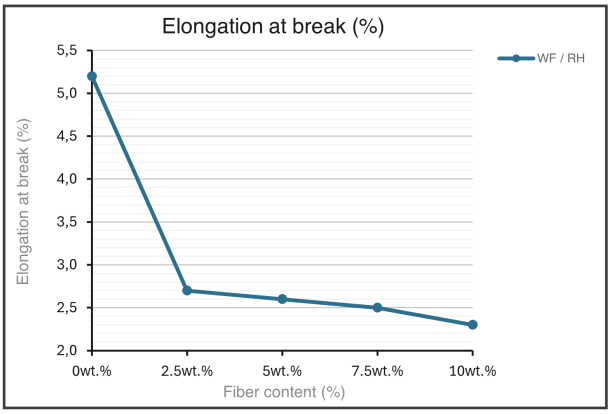


Figure A.23. Effect of 2wt.% Alkaline-treated wood flour/rice husk (in 50:50wt.% ratio) content in PLA matrix on the elongation at break. Adapted from "*Valorization of Waste Wood Flour and Rice Husk in Poly(Lactic Acid)-Based Hybrid Biocomposites*", by Lendvai et al., 2022 [61].

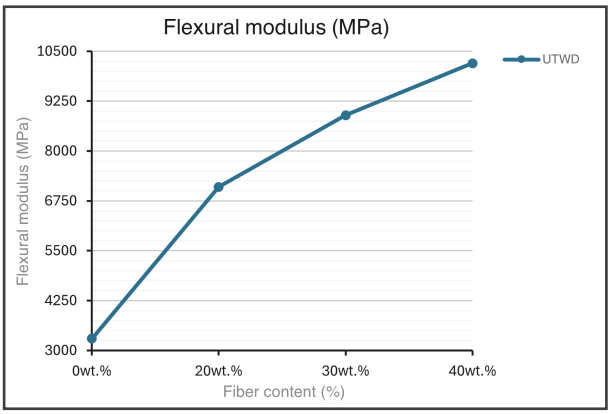


Figure A.24. Effect of untreated olive wood waste dust (UTWF) content in PLA matrix on the flexural modulus. Adapted from "*Mechanical Properties of Wood Fibre Filled Polylactic Acid (PLA) Composites Using Additive Manufacturing Techniques*", by Mohammadsalih et al., 2023 [77].

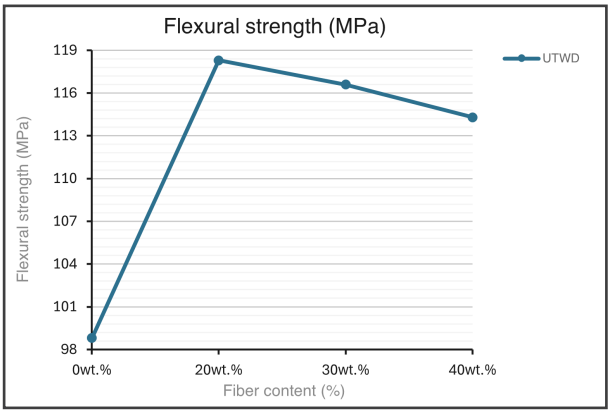


Figure A.25. Effect of untreated olive wood waste dust (UTWF) content in PLA matrix on the flexural strength. Adapted from "*Mechanical Properties of Wood Fibre Filled Polylactic Acid (PLA) Composites Using Additive Manufacturing Techniques*", by Mohammadsalih et al., 2023 [77].

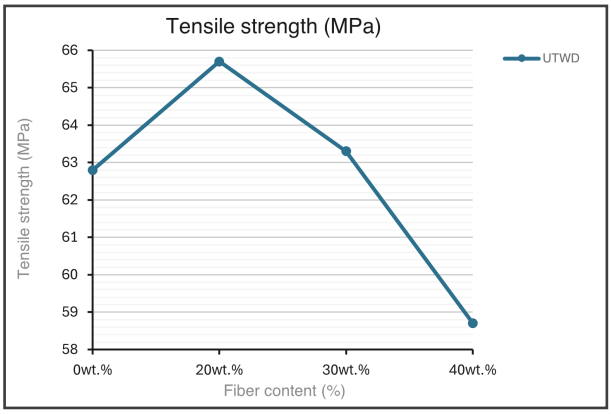


Figure A.26. Effect of untreated olive wood waste dust (UTWF) content in PLA matrix on the tensile strength. Adapted from "*Mechanical Properties of Wood Fibre Filled Polylactic Acid (PLA) Composites Using Additive Manufacturing Techniques*", by Mohammadsalih et al., 2023 [77].

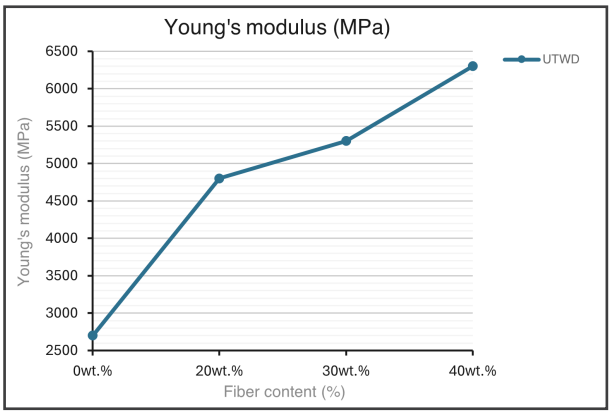


Figure A.27. Effect of untreated olive wood waste dust (UTWF) content in PLA matrix on the Young's modulus. Adapted from "*Mechanical Properties of Wood Fibre Filled Polylactic Acid (PLA) Composites Using Additive Manufacturing Techniques*", by Mohammadsalih et al., 2023 [77].

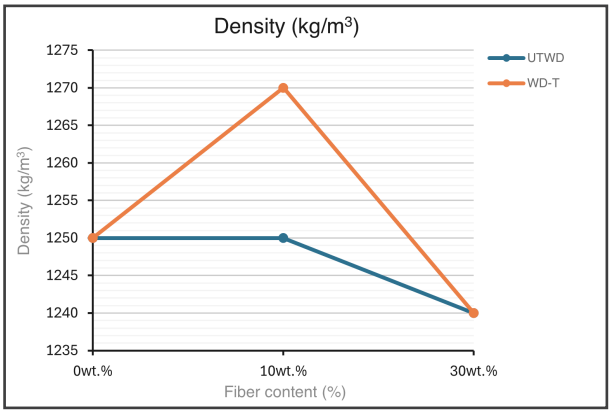


Figure A.28. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the density. Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

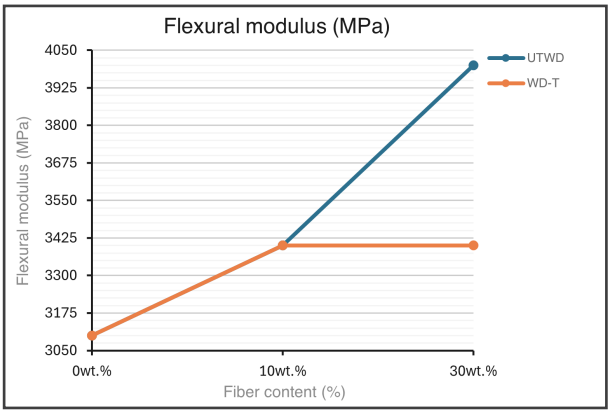


Figure A.29. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the flexural modulus. Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

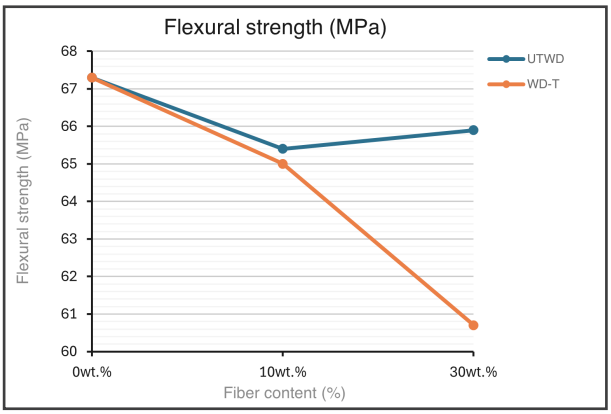


Figure A.30. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the flexural strength. Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

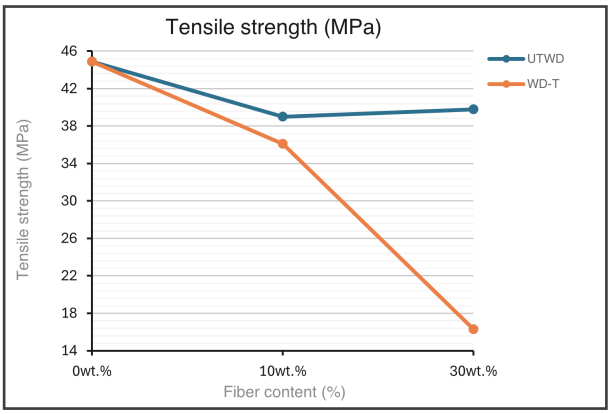


Figure A.31. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the tensile strength. Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

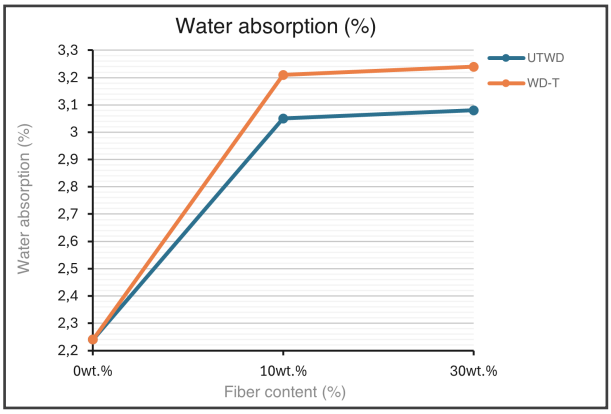


Figure A.32. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the water absorption (20°C / 480h). Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

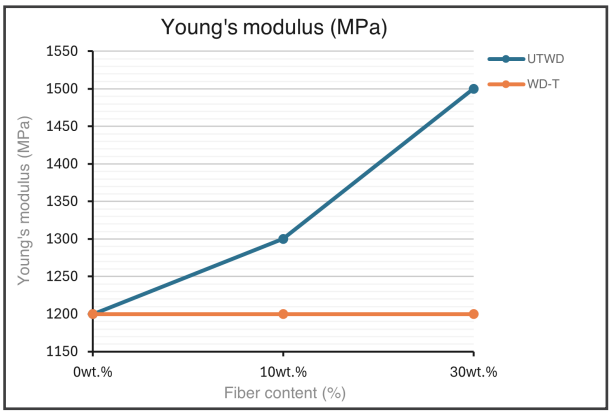


Figure A.33. Effect of fiber content (blue: untreated pine wood flour {UTWD}, and orange: thermally-treated pine wood flour {WD-T}) in PLA matrix on the Young's modulus. Adapted from "*Wood Flour-Reinforced Green Composites: Parameter Optimization via Multi-criteria Decision-Making Methods*", by Kelleci et al., 2022 [55].

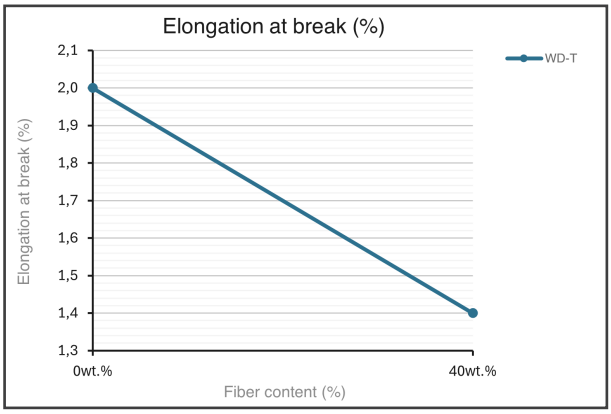


Figure A.34. Effect of thermally-treated spruce wood flour (WD-T) content in PLA matrix on the elongation at break. Adapted from "*Surface Modification of Spruce Wood Flour and Effects on the Dynamic Fragility of PLA/Wood Composites*", by Gregorova et al., 2011 [46].

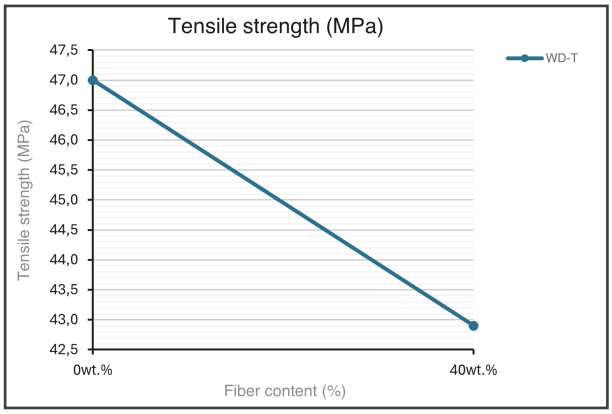


Figure A.35. Effect of thermally-treated spruce wood flour (WD-T) content in PLA matrix on the tensile strength. Adapted from "*Surface Modification of Spruce Wood Flour and Effects on the Dynamic Fragility of PLA/Wood Composites*", by Gregorova et al., 2011 [46].

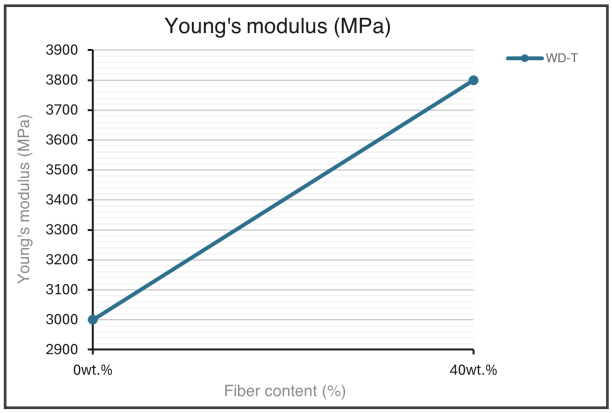


Figure A.36. Effect of thermally-treated spruce wood flour (WD-T) content in PLA matrix on the Young's modulus. Adapted from "*Surface Modification of Spruce Wood Flour and Effects on the Dynamic Fragility of PLA/Wood Composites*", by Gregorova et al., 2011 [46].

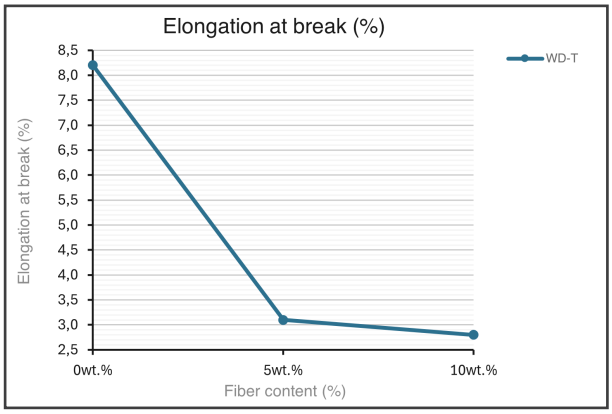


Figure A.37. Effect of thermally-treated wood flour (WD-T) content in PLA matrix on the elongation at break. Adapted from "*Properties of Biocomposite Films From PLA and Thermally Treated Wood Modified with Silver Nanoparticles Using Leaf Extracts of Oriental Sweetgum*", by Ayrilmis et al., 2021 [8].

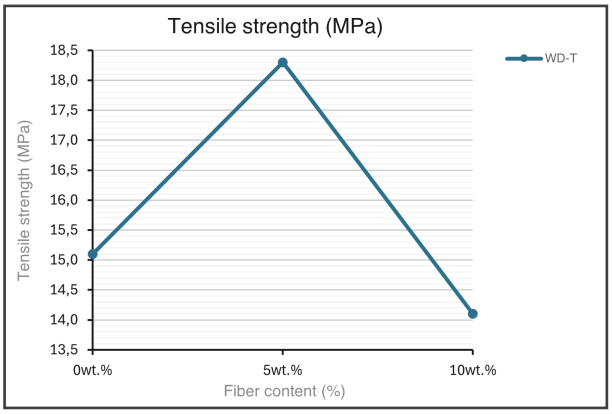


Figure A.38. Effect of thermally-treated wood flour (WD-T) content in PLA matrix on the tensile strength. Adapted from "*Properties of Biocomposite Films From PLA and Thermally Treated Wood Modified with Silver Nanoparticles Using Leaf Extracts of Oriental Sweetgum*", by Ayrilmis et al., 2021 [8].

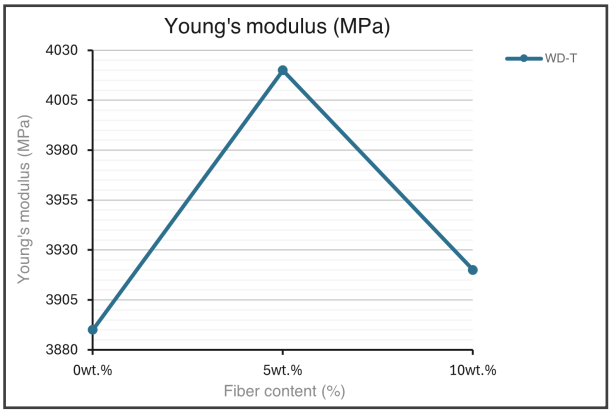


Figure A.39. Effect of thermally-treated wood flour (WD-T) content in PLA matrix on the Young's modulus. Adapted from "*Properties of Biocomposite Films From PLA and Thermally Treated Wood Modified with Silver Nanoparticles Using Leaf Extracts of Oriental Sweetgum*", by Ayrilmis et al., 2021 [8].

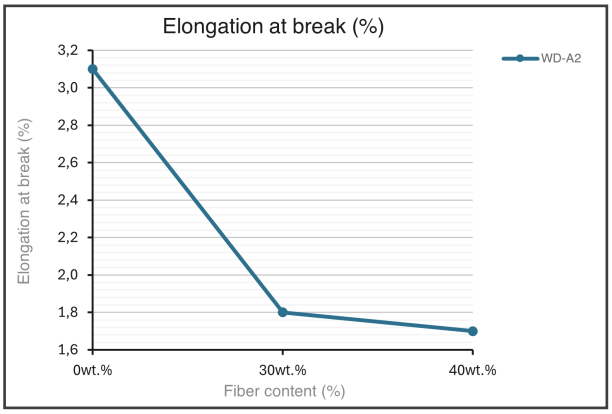


Figure A.40. Effect of 2wt.% Alkaline-treated wood flour (WD-A2) content in PLA matrix on the elongation at break. Adapted from "*Effect of Alkaline Treatment and Pre-impregnation on Mechanical and Water Absorbtion Properties of Pine Wood Flour Containing Poly (Lactic Acid) Based Green-Composites*", by Altun et al., 2012 [6].

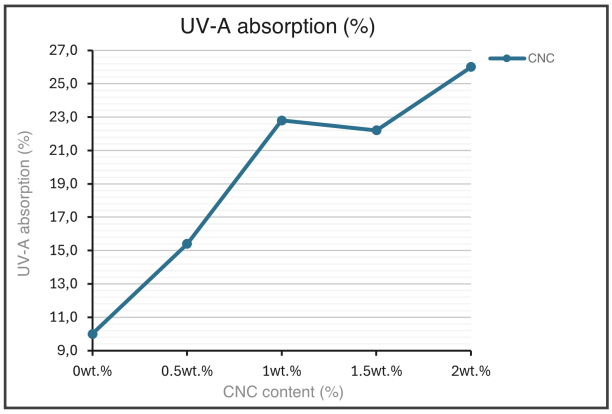


Figure A.41. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from wood pulp) content in PLA matrix on the UV-A absorption. Adapted from "*Water vapor and oxygen barrier properties of extrusion-blown poly(lactic acid)/ cellulose nanocrystals nanocomposite films*", by Karkhanis et al., 2018 [54].

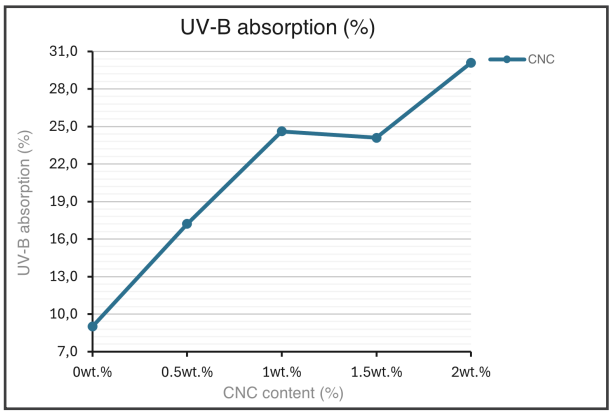


Figure A.42. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from wood pulp) content in PLA matrix on the UV-B absorption. Adapted from "*Water vapor and oxygen barrier properties of extrusion-blown poly(lactic acid)/ cellulose nanocrystals nanocomposite films*", by Karkhanis et al., 2018 [54].

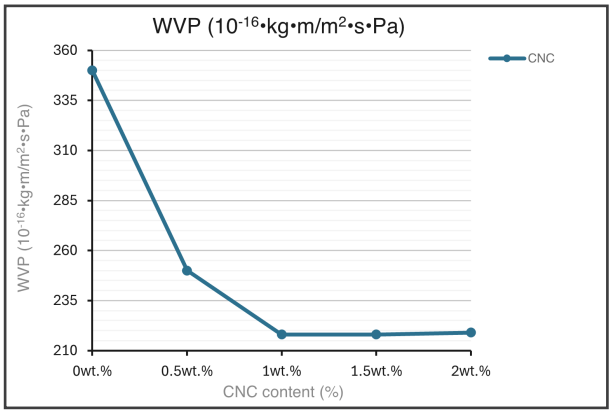


Figure A.43. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from wood pulp) content in PLA matrix on the water vapour permeability. Adapted from "*Water vapor and oxygen barrier properties of extrusion-blown poly(lactic acid)/cellulose nanocrystals nanocomposite films*", by Karkhanis et al., 2018 [54].

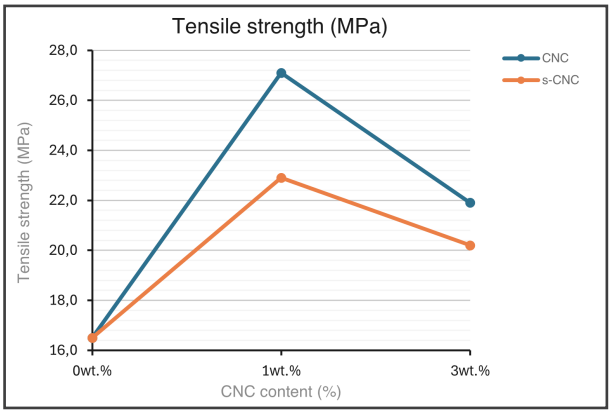


Figure A.44. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from Posidonia Oceanica Alga, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the tensile strength. Adapted from "*Processing of PLA nanocomposites with cellulose nanocrystals extracted from Posidonia oceanica waste: Innovative reuse of coastal plant*", by Fortunati et al., 2015 [39].

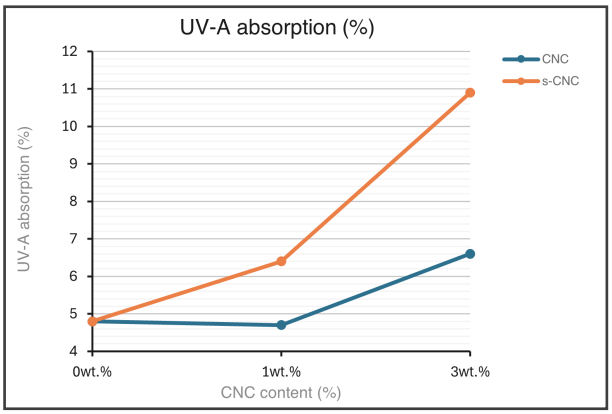


Figure A.45. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from Posidonia Oceanica Alga, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the UV-A absorption. Adapted from "*Processing of PLA nanocomposites with cellulose nanocrystals extracted from Posidonia oceanica waste: Innovative reuse of coastal plant*", by Fortunati et al., 2015 [39].

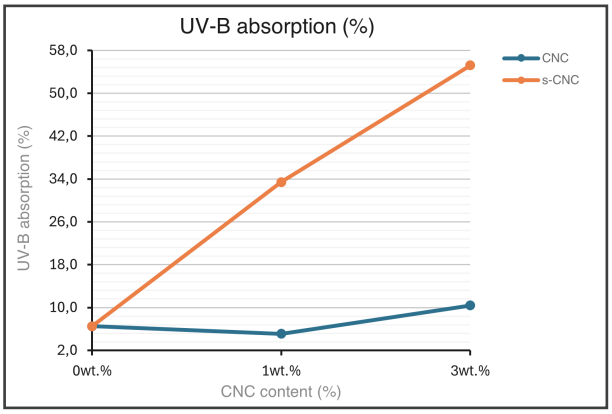


Figure A.46. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from Posidonia Oceanica Alga, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the UV-B absorption. Adapted from "*Processing of PLA nanocomposites with cellulose nanocrystals extracted from Posidonia oceanica waste: Innovative reuse of coastal plant*", by Fortunati et al., 2015 [39].

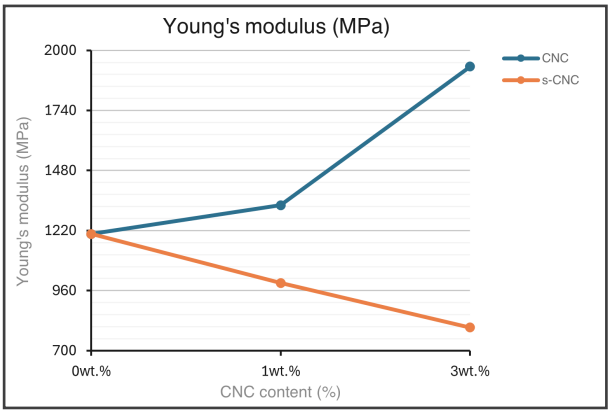


Figure A.47. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from Posidonia Oceanica Alga, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the Young's modulus. Adapted from "*Processing of PLA nanocomposites with cellulose nanocrystals extracted from Posidonia oceanica waste: Innovative reuse of coastal plant*", by Fortunati et al., 2015 [39].

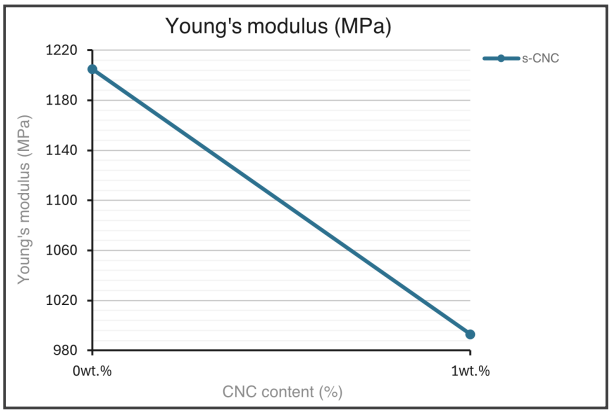


Figure A.50. Effect of surfactant-modified cellulose nanocrystals (CNC extracted with acid hydrolysis from Posidonia Oceanica Alga and modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) content in PLA matrix on the Young's modulus. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

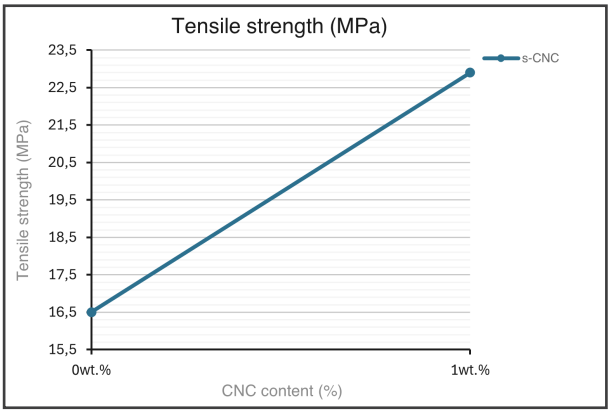


Figure A.48. Effect of surfactant-modified cellulose nanocrystals (CNC extracted with acid hydrolysis from Posidonia Oceanica Alga and modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) content in PLA matrix on the tensile strength. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

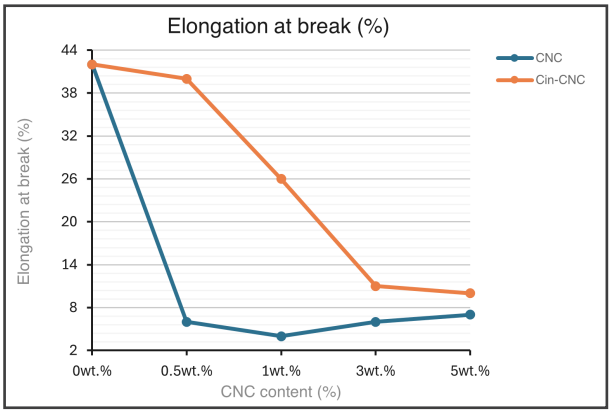


Figure A.51. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the elongation at break. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

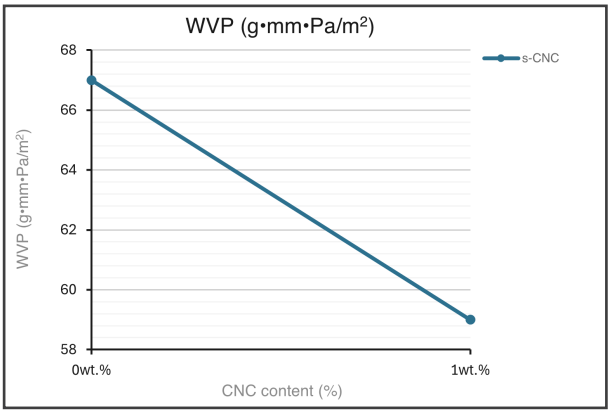


Figure A.49. Effect of surfactant-modified cellulose nanocrystals (CNC extracted with acid hydrolysis from Posidonia Oceanica Alga and modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) content in PLA matrix on the water vapour permeability. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

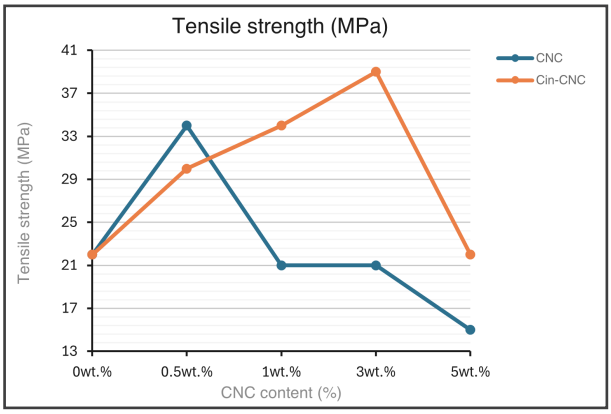


Figure A.52. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the tensile strength. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

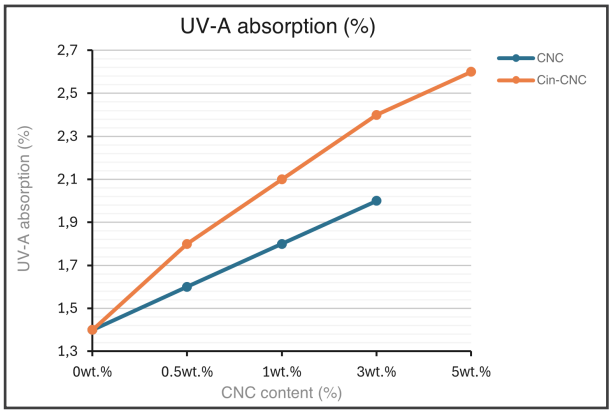


Figure A.53. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the UV-A absorption. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

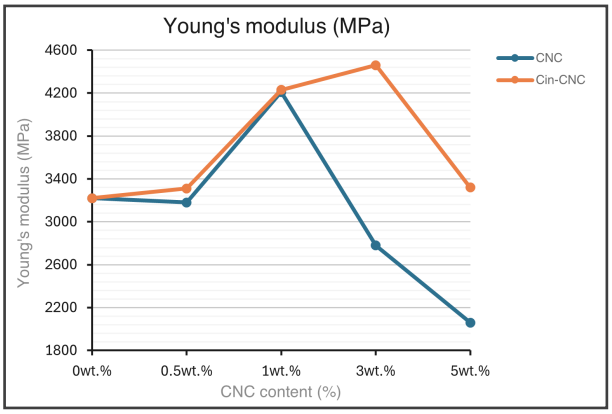


Figure A.56. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the Young's modulus. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

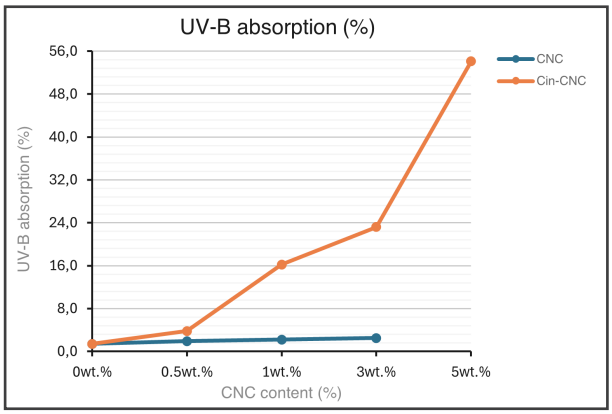


Figure A.54. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the UV-B absorption. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

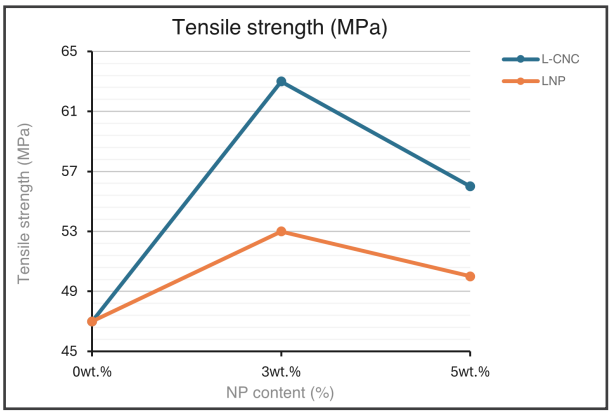


Figure A.57. Effect of nanoparticle content (blue: lignin-coated cellulose nanocrystals extracted with acid hydrolysis from wood-based cellulose {L-CNC}, and orange: lignin nanoparticles {LNP}) in PLA matrix on the tensile strength. Adapted from "*Enhancing UV-shielding and mechanical properties of polylactic acid nanocomposites by adding lignin coated cellulose nanocrystals*", by Shojaeiarani et al., 2022 [85].

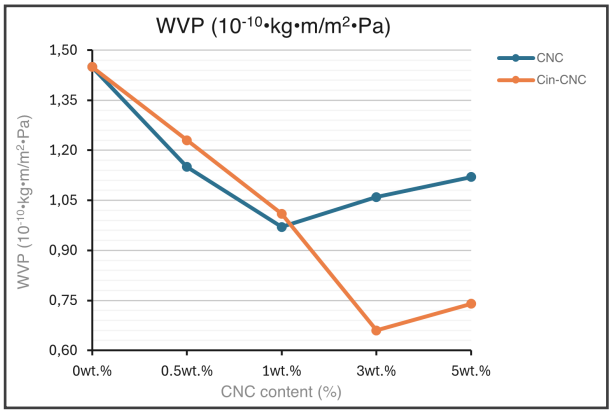


Figure A.55. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from pineapple leaves, and orange: cinnamate-grafted CNC through acylation {Cin-CNC}) in PLA matrix on the water vapour permeability. Adapted from "*Functionalization of cellulose nanocrystals extracted from pineapple leaves as a UV-absorbing agent in poly(lactic acid)*", by Sringam et al., 2023 [88].

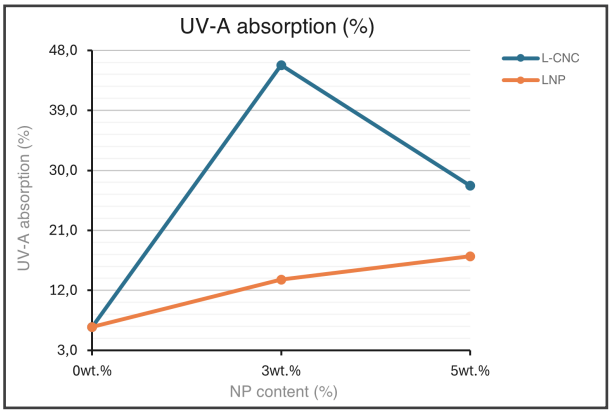


Figure A.58. Effect of nanoparticle content (blue: lignin-coated cellulose nanocrystals extracted with acid hydrolysis from wood-based cellulose {L-CNC}, and orange: lignin nanoparticles {LNP}) in PLA matrix on the UV-A absorption. Adapted from "*Enhancing UV-shielding and mechanical properties of polylactic acid nanocomposites by adding lignin coated cellulose nanocrystals*", by Shojaeiarani et al., 2022 [85].

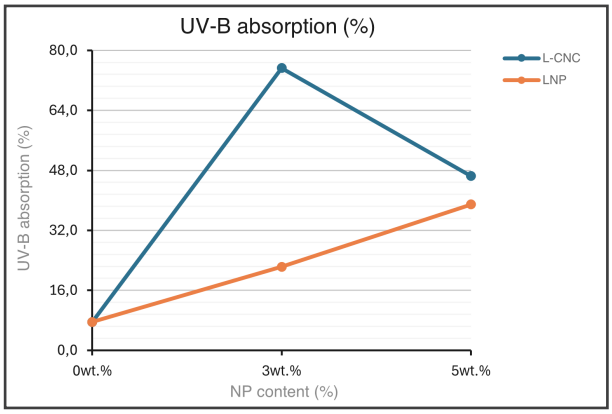


Figure A.59. Effect of nanoparticle content (blue: lignin-coated cellulose nanocrystals extracted with acid hydrolysis from wood-based cellulose {L-CNC}, and orange: lignin nanoparticles {LNP}) in PLA matrix on the UV-B absorption. Adapted from "*Enhancing UV-shielding and mechanical properties of polylactic acid nanocomposites by adding lignin coated cellulose nanocrystals*", by Shojaeiarani et al., 2022 [85].

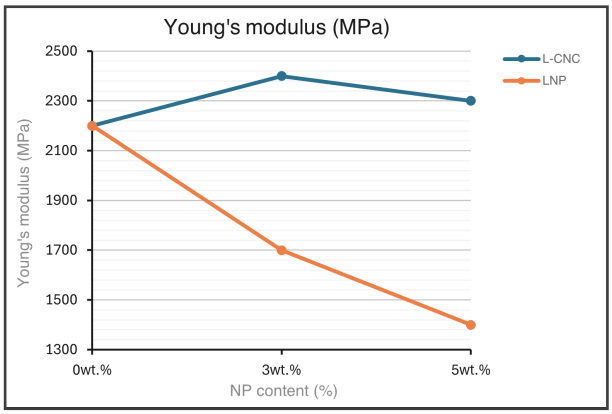


Figure A.60. Effect of nanoparticle content (blue: lignin-coated cellulose nanocrystals extracted with acid hydrolysis from wood-based cellulose {L-CNC}, and orange: lignin nanoparticles {LNP}) in PLA matrix on the Young's modulus. Adapted from "*Enhancing UV-shielding and mechanical properties of polylactic acid nanocomposites by adding lignin coated cellulose nanocrystals*", by Shojaeiarani et al., 2022 [85].

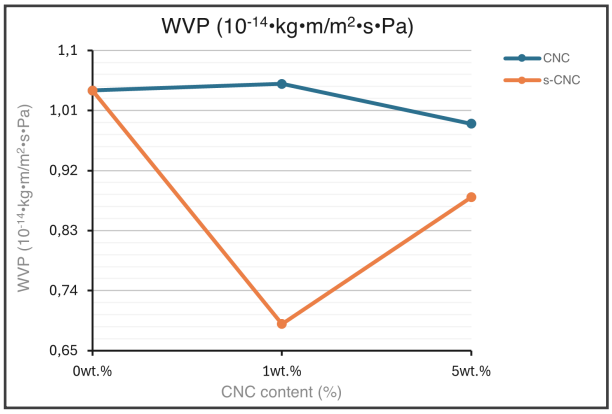


Figure A.61. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the water vapour permeability. Adapted from "*Combined effects of cellulose nanocrystals and silver nanoparticles on the barrier and migration properties of PLA nano-biocomposites*", by Fortunati et al., 2013 [40].

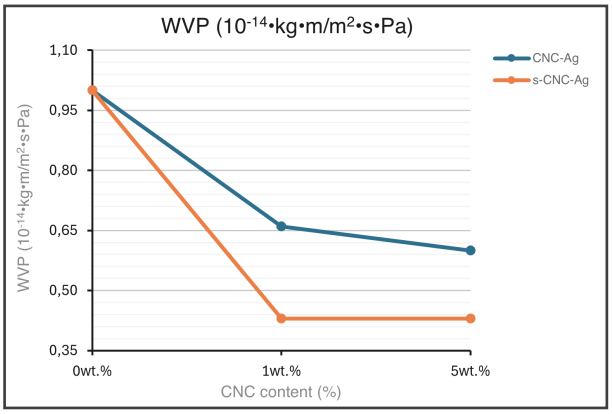


Figure A.62. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose + 1wt.% silver nanopowder {CNC-Ag}, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio + 1wt.% silver nanopowder {s-CNC-Ag}) in PLA matrix on the water vapour permeability. Adapted from "*Combined effects of cellulose nanocrystals and silver nanoparticles on the barrier and migration properties of PLA nano-biocomposites*", by Fortunati et al., 2013 [40].

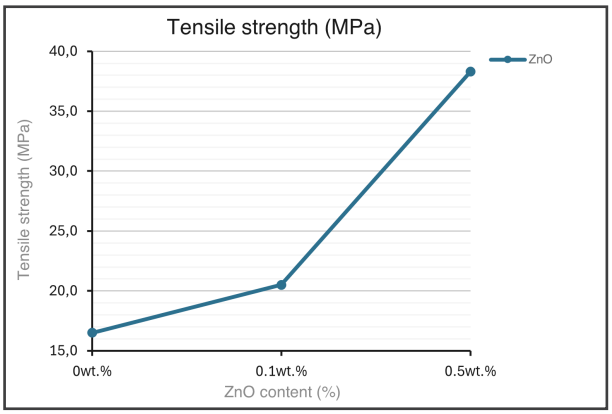


Figure A.63. Effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

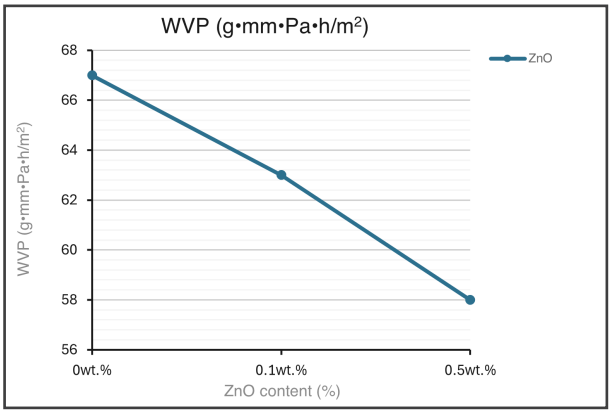


Figure A.64. Effect of Zinc oxide (ZnO) content in PLA matrix on the water vapour permeability. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

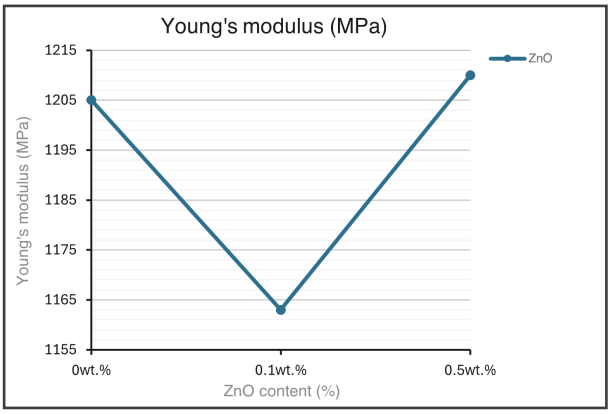


Figure A.65. Effect of Zinc oxide (ZnO) content in PLA matrix on the Young's modulus. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

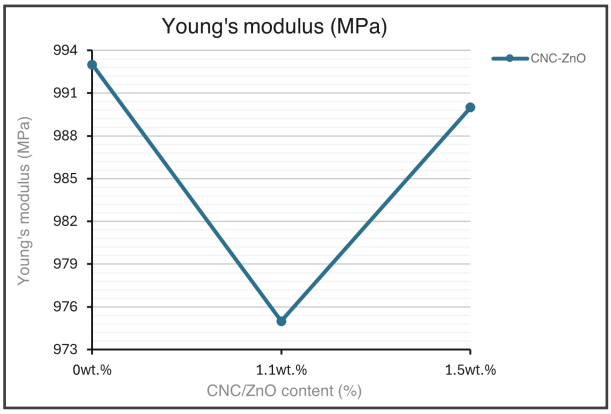


Figure A.68. Effect of cellulose nanocrystals-Zinc oxide nanoparticles (1wt.% CNC extracted with acid hydrolysis from Posidonia Oceanica Alga) content in PLA matrix on the Young's modulus. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

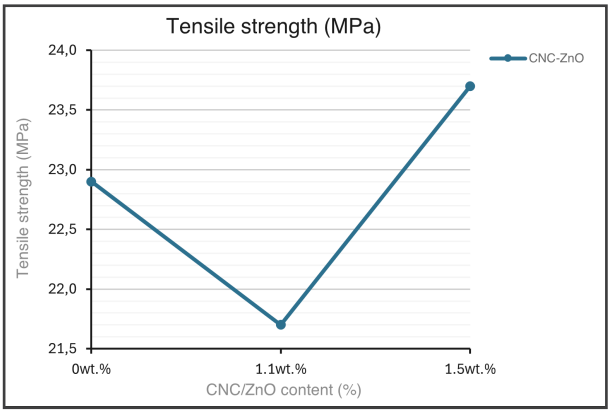


Figure A.66. Effect of cellulose nanocrystals-Zinc oxide nanoparticles (1wt.% CNC extracted with acid hydrolysis from Posidonia Oceanica Alga) content in PLA matrix on the tensile strength. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

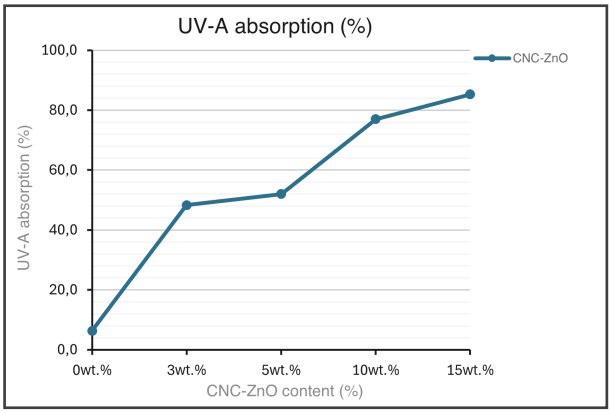


Figure A.69. Effect of cellulose nanocrystals-Zinc oxide nanoparticles (CNC extracted with acid hydrolysis from microcrystalline cellulose and 50:50vo.% ratio of CNC:ZnO) content in PLA matrix on the UV-A absorption. Adapted from "*Enhancing long-term biodegradability and UV-shielding performances of transparent polylactic acid nanocomposite films by adding cellulose nanocrystal-zinc oxide hybrids*", by Wang et al., 2019 [97].

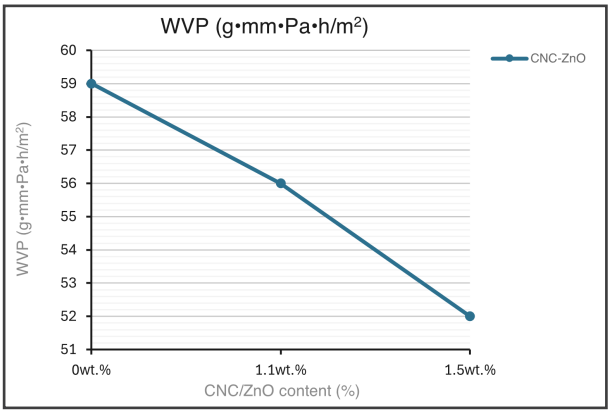


Figure A.67. Effect of cellulose nanocrystals-Zinc oxide nanoparticles (1wt.% CNC extracted with acid hydrolysis from Posidonia Oceanica Alga) content in PLA matrix on the water vapour permeability. Adapted from "*PLA Nanocomposites Reinforced with Celluose Nanocrystals from Posidonia oceanica and ZnO Nanoparticles for Packaging Application*", by Luzi et al., 2016 [70].

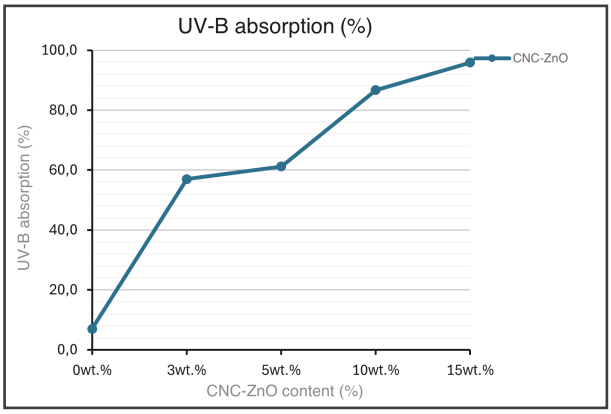


Figure A.70. Effect of cellulose nanocrystals-Zinc oxide nanoparticles (CNC extracted with acid hydrolysis from microcrystalline cellulose and 50:50vo.% ratio of CNC:ZnO) content in PLA matrix on the UV-B absorption. Adapted from "*Enhancing long-term biodegradability and UV-shielding performances of transparent polylactic acid nanocomposite films by adding cellulose nanocrystal-zinc oxide hybrids*", by Wang et al., 2019 [97].

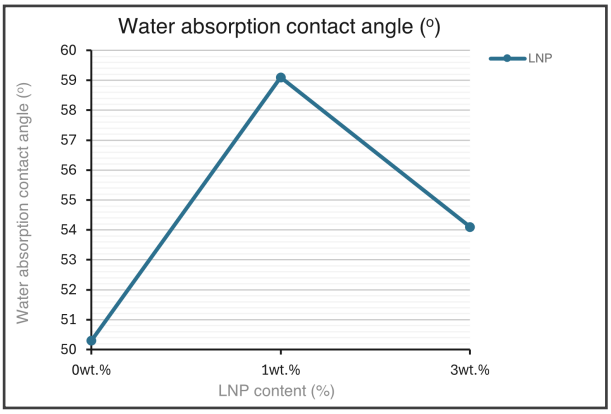


Figure A.71. Effect of lignin nanoparticles (LNP) content in PLA matrix on the water absorption contact angle. Adapted from "*Effect of processing conditions and lignin content on thermal, mechanical and degradative behavior of lignin nanoparticles/polylactic(acid) bionanocomposites prepared by melt extrusion and solvent casting*", by Yang et al., 2015 [103].

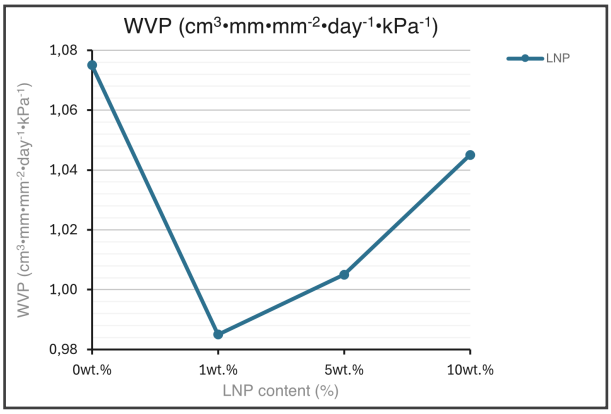


Figure A.74. Effect of lignin nanoparticles (LNP) content in PLA matrix on the water vapour permeability. Adapted from "*Uniformly Dispersed Poly(lactic acid)-Grafted Lignin Nanoparticles Enhance Antioxidant Activity and UV-Barrier Properties of Poly(lactic acid) Packaging Films*", by Boarino et al., 2022 [19].

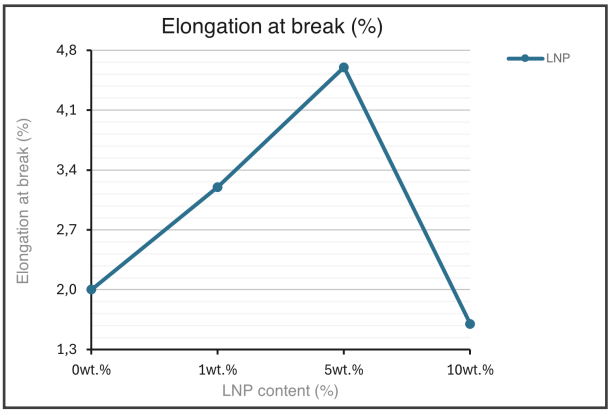


Figure A.72. Effect of lignin nanoparticles (LNP) content in PLA matrix on the elongation at break. Adapted from "*Uniformly Dispersed Poly(lactic acid)-Grafted Lignin Nanoparticles Enhance Antioxidant Activity and UV-Barrier Properties of Poly(lactic acid) Packaging Films*", by Boarino et al., 2022 [19].

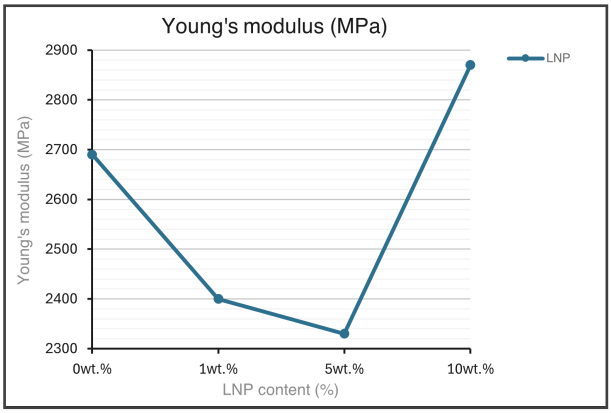


Figure A.75. Effect of lignin nanoparticles (LNP) content in PLA matrix on the Young's modulus. Adapted from "*Uniformly Dispersed Poly(lactic acid)-Grafted Lignin Nanoparticles Enhance Antioxidant Activity and UV-Barrier Properties of Poly(lactic acid) Packaging Films*", by Boarino et al., 2022 [19].

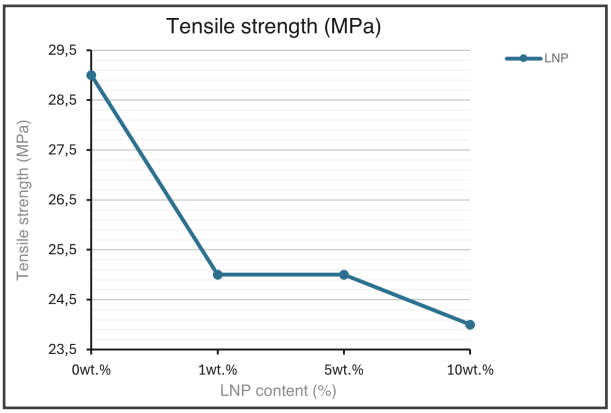


Figure A.73. Effect of lignin nanoparticles (LNP) content in PLA matrix on the tensile strength. Adapted from "*Uniformly Dispersed Poly(lactic acid)-Grafted Lignin Nanoparticles Enhance Antioxidant Activity and UV-Barrier Properties of Poly(lactic acid) Packaging Films*", by Boarino et al., 2022 [19].

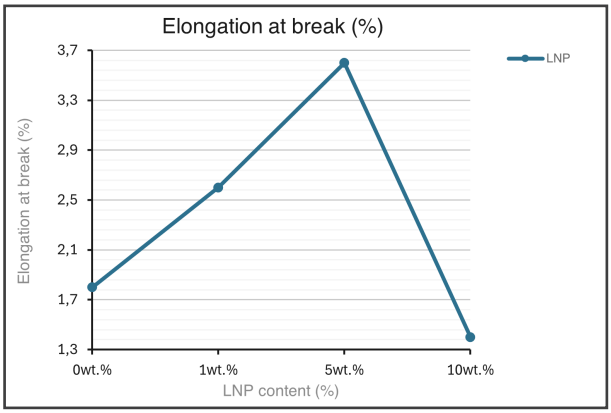


Figure A.76. Effect of lignin nanoparticles (LNP) content in PLA matrix on the elongation at break. Adapted from "*Optimization of the Electrospray Process to Produce Lignin Nanoparticles for PLA-based Food Packaging*", by Daassi et al., 2023 [30].

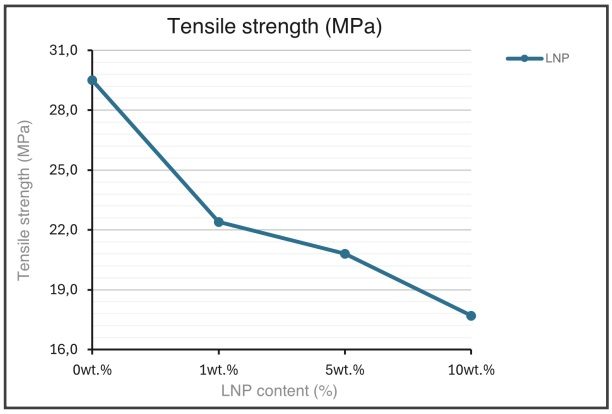


Figure A.77. Effect of lignin nanoparticles (LNP) content in PLA matrix on the tensile strength. Adapted from "*Optimization of the Electrospray Process to Produce Lignin Nanoparticles for PLA-based Food Packaaging*", by Daassi et al., 2023 [30].

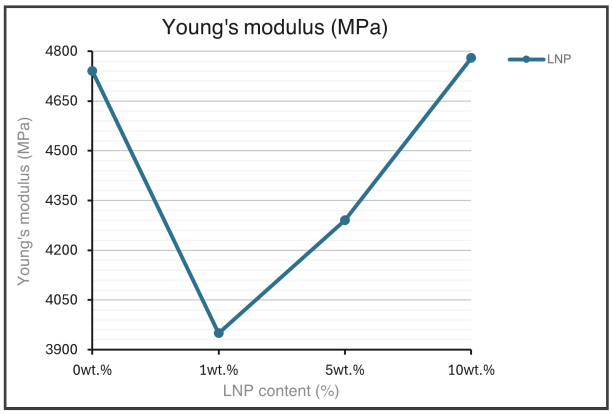


Figure A.78. Effect of lignin nanoparticles (LNP) content in PLA matrix on the Young's modulus. Adapted from "*Optimization of the Electrospray Process to Produce Lignin Nanoparticles for PLA-based Food Packaging*", by Daassi et al., 2023 [30].

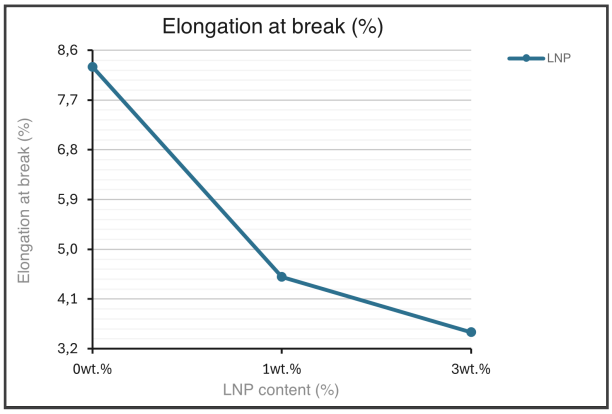


Figure A.79. Effect of lignin nanoparticles (LNP) content in PLA matrix on the elongation at break. Adapted from "*UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles*", by Cavallo et al., 2021 [24].

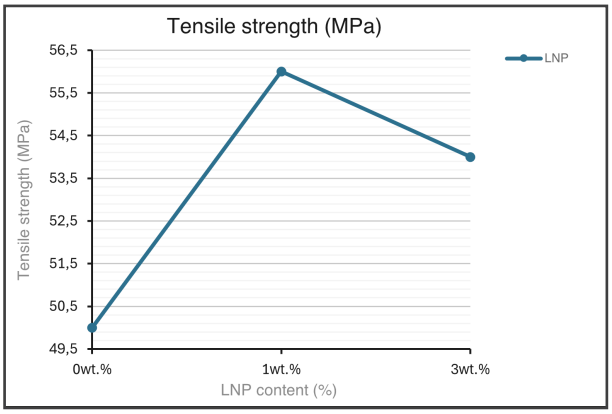


Figure A.80. Effect of lignin nanoparticles (LNP) content in PLA matrix on the tensile strength. Adapted from "*UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles*", by Cavallo et al., 2021 [24].

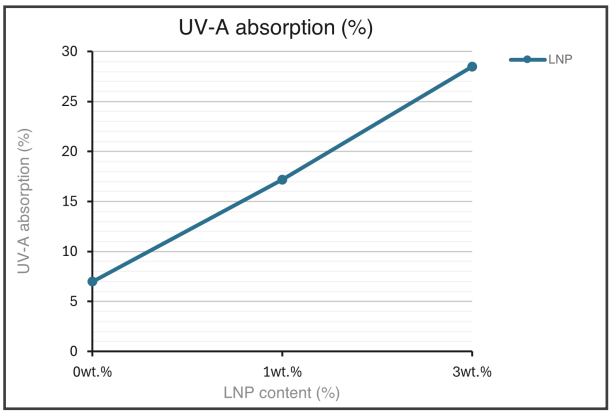


Figure A.81. Effect of lignin nanoparticles (LNP) content in PLA matrix on the UV-A absorption. Adapted from "*UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles*", by Cavallo et al., 2021 [24].

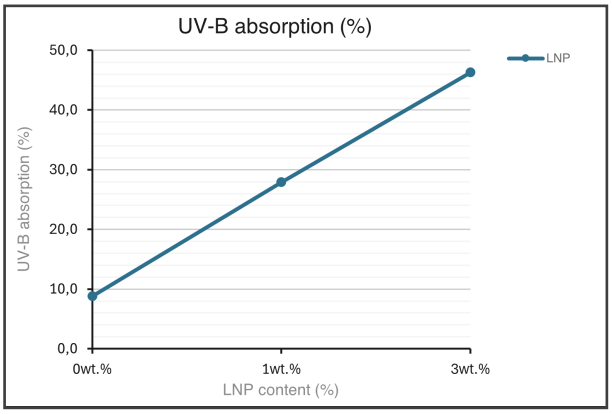


Figure A.82. Effect of lignin nanoparticles (LNP) content in PLA matrix on the UV-B absorption. Adapted from "*UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles*", by Cavallo et al., 2021 [24].

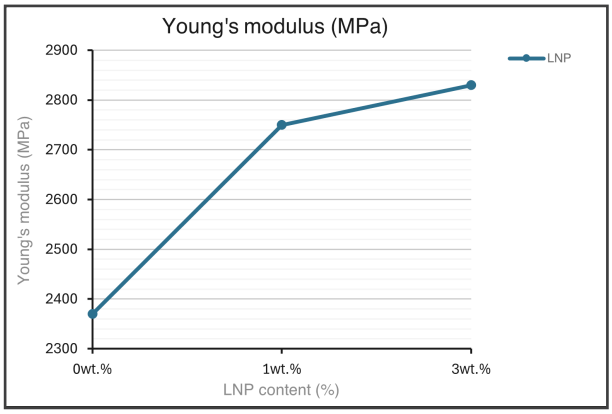


Figure A.83. Effect of lignin nanoparticles (LNP) content in PLA matrix on the Young's modulus. Adapted from "*UV Protective, Antioxidant, Antibacterial and Compostable Polylactic Acid Composites Containing Pristine and Chemically Modified Lignin Nanoparticles*", by Cavallo et al., 2021 [24].

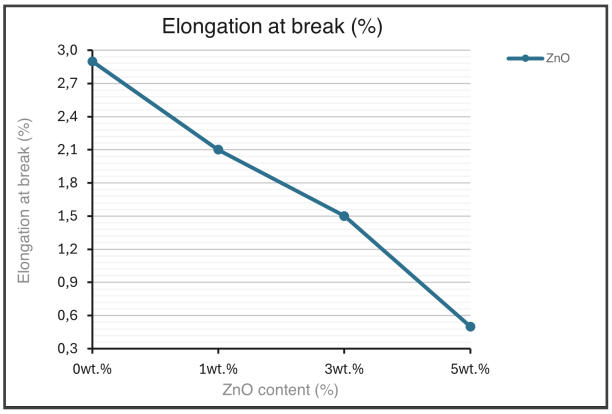


Figure A.86. Effect of Zinc oxide (ZnO) content in PLA matrix on the elongation at break. Adapted from "*Advancing the additive manufacturing of PLA-ZnO nanocomposites by fused filament frabrication*", by Chong et al., 2023 [26].

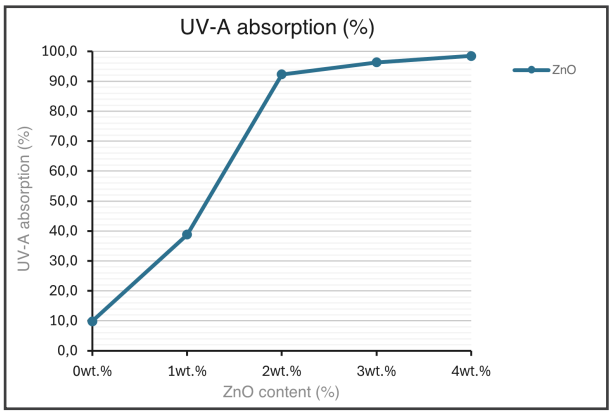


Figure A.84. Effect of Zinc oxide (ZnO) content in PLA matrix on the UV-A absorption. Adapted from "*Characterization of an eco-friendly active packaging film for food with ultraviolet light blocking ability*", by Li et al., 2023 [63].

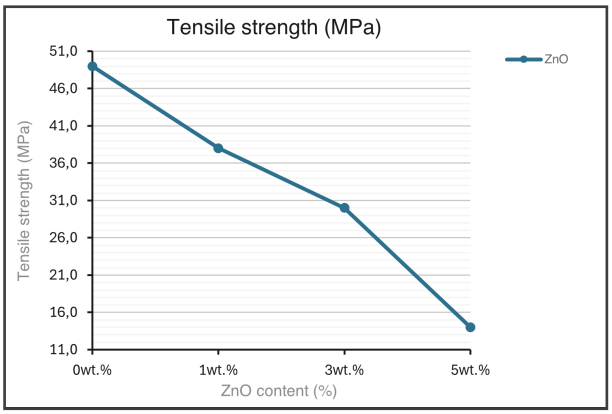


Figure A.87. Effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength. Adapted from "*Advancing the additive manufacturing of PLA-ZnO nanocomposites by fused filament frabrication*", by Chong et al., 2023 [26].

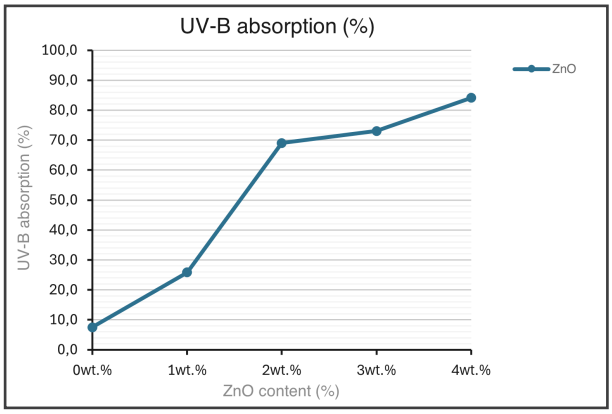


Figure A.85. Effect of Zinc oxide (ZnO) content in PLA matrix on the UV-B absorption. Adapted from "*Characterization of an eco-friendly active packaging film for food with ultraviolet light blocking ability*", by Li et al., 2023 [63].

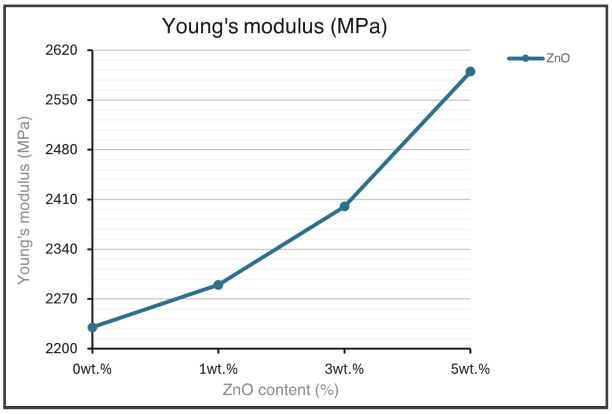


Figure A.88. Effect of Zinc oxide (ZnO) content in PLA matrix on the Young's modulus. Adapted from "*Advancing the additive manufacturing of PLA-ZnO nanocomposites by fused filament frabrication*", by Chong et al., 2023 [26].

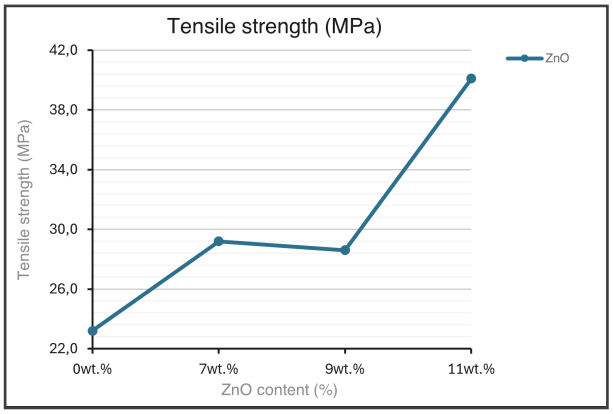


Figure A.89. Effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength. Adapted from "*Effect of zinc oxide suspension on the overall filler content of the PLA/ZnO composites and cPLA/ZnO composites*", by Tan et al., 2023 [90].

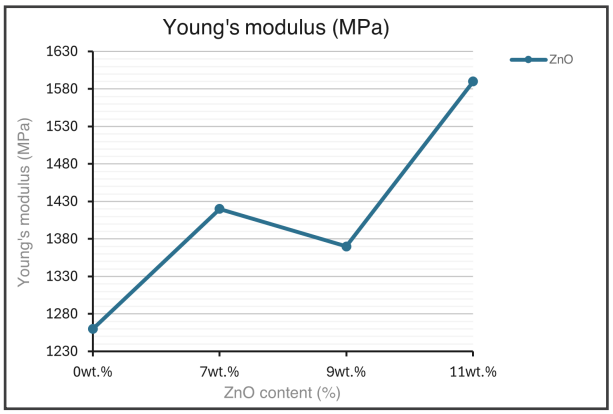


Figure A.90. Effect of Zinc oxide (ZnO) content in PLA matrix on the Young"s modulus. Adapted from "*Effect of zinc oxide suspension on the overall filler content of the PLA/ZnO composites and cPLA/ZnO composites*", by Tan et al., 2023 [90].

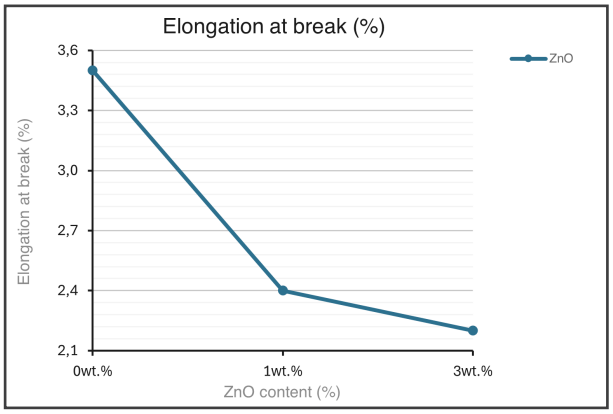


Figure A.91. Effect of Zinc oxide (ZnO) content in PLA matrix on the elongation at break. Adapted from "*Polydopamine surface functionalized submicron ZnO for broadening the processing window of 3D printable PLA composites*", by Yang et al., 2023 [104].

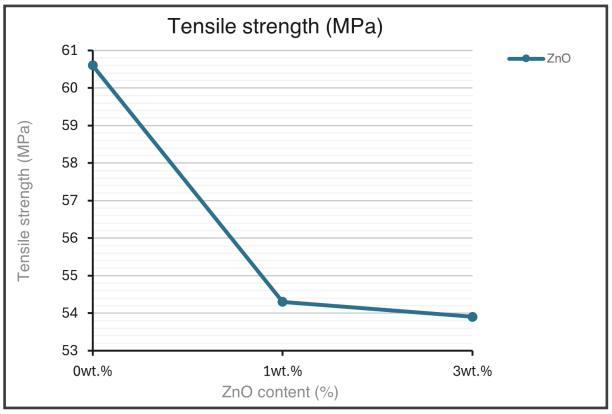


Figure A.92. Effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength. Adapted from "*Polydopamine surface functionalized submicron ZnO for broadening the processing window of 3D printable PLA composites*", by Yang et al., 2023 [104].

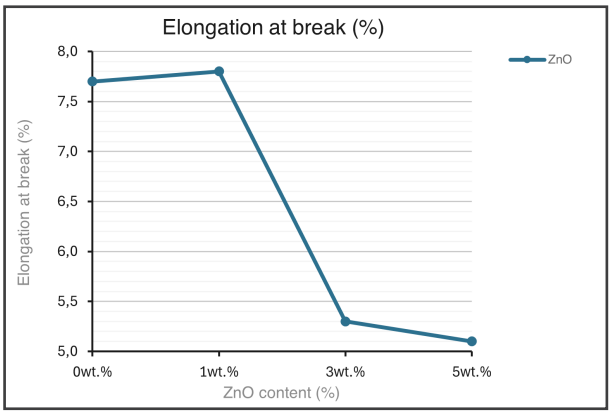


Figure A.93. Effect of Zinc oxide (ZnO) content in PLA matrix on the elongation at break. Adapted from "*A Comprehensive Evaluation of Mechanical, Thermal, and Antibacterial Properties of PLA/ZnO Nanoflower Biocomposite Filaments for 3D Printing Application*", by Jamnongkan et al., 2022 [50].

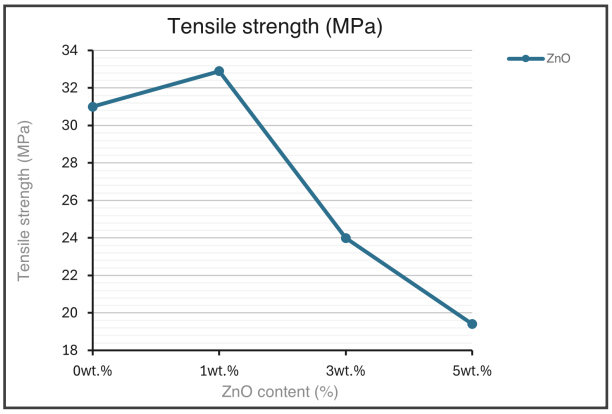


Figure A.94. Effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength. Adapted from "*A Comprehensive Evaluation of Mechanical, Thermal, and Antibacterial Properties of PLA/ZnO Nanoflower Biocomposite Filaments for 3D Printing Application*", by Jamnongkan et al., 2022 [50].

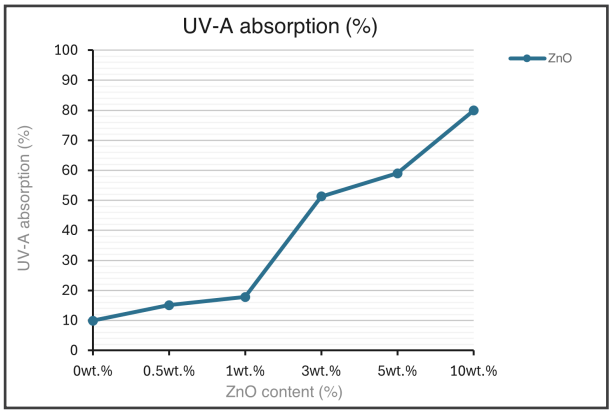


Figure A.95. Effect of Zinc oxide (ZnO) content in PLA matrix on the UV-A absorption. Adapted from "*Poly(Lactic Acid)/Zno Bionanocomposite Films with Positively Charged Zno as Potential Antimicrobial Food Packaging materials*", by Kim et al., 2019 [57].

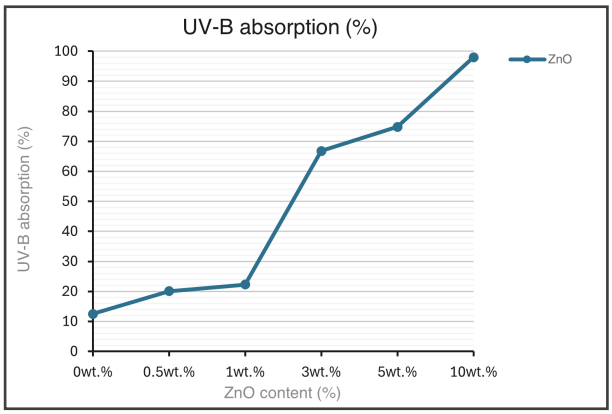


Figure A.96. Effect of Zinc oxide (ZnO) content in PLA matrix on the UV-B absorption. Adapted from "*Poly(Lactic Acid)/Zno Bionanocomposite Films with Positively Charged Zno as Potential Antimicrobial Food Packaging materials*", by Kim et al., 2019 [57].

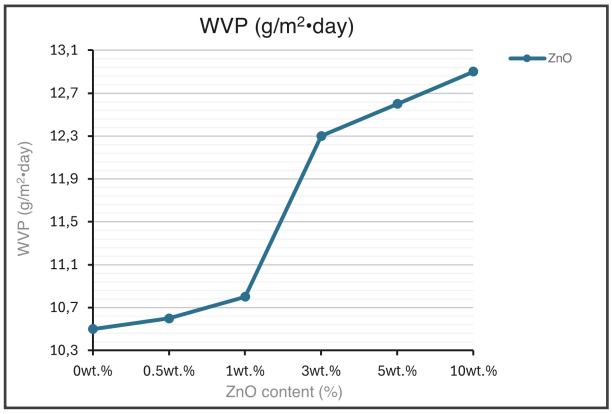


Figure A.97. Effect of Zinc oxide (ZnO) content in PLA matrix on the water vapour permeability. Adapted from "*Poly(Lactic Acid)/Zno Bionanocomposite Films with Positively Charged Zno as Potential Antimicrobial Food Packaging materials*", by Kim et al., 2019 [57].

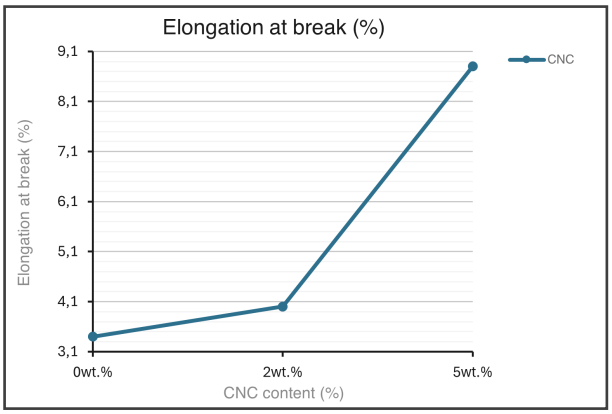


Figure A.98. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from bleached dry lap eucalypyus wood pulp) content in PLA matrix on the elongation at break. Adapted from "*Modification of Cellulose Nanocrystals (CNCs) for use in Poly(lactic acid) (PLA)-CNC Composite Packaging Products*", by Wei et al., 2016 [100].

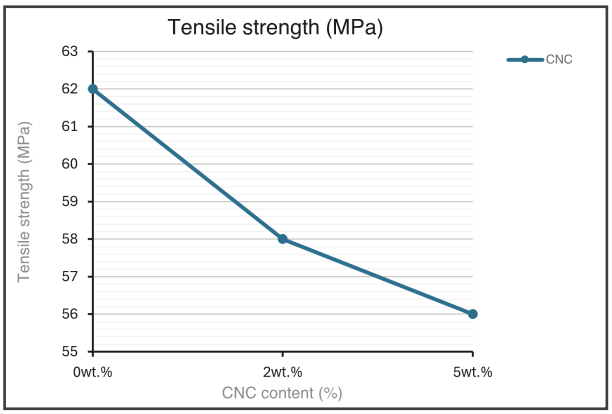


Figure A.99. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from bleached dry lap eucalypyus wood pulp) content in PLA matrix on the tensile strength. Adapted from "*Modification of Cellulose Nanocrystals (CNCs) for use in Poly(lactic acid) (PLA)-CNC Composite Packaging Products*", by Wei et al., 2016 [100].

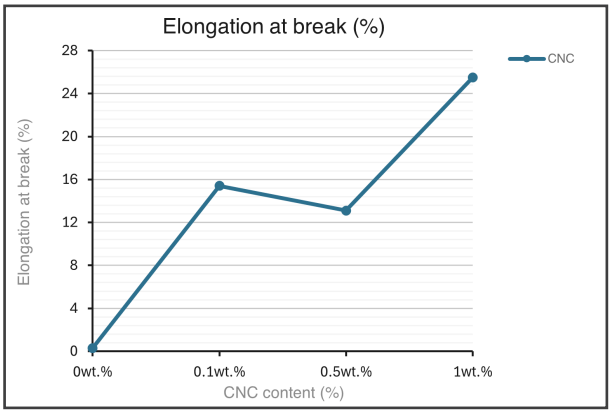


Figure A.100. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from Eucalyptus grandis sawdust) content in PLA matrix on the elongation at break. Adapted from "*Preparation of cellulose nanocrystal (CNCs) reinforced polylactic acid (PLA) bionanocomposites filaments using biobased additives for 3D printing applications*", by Agbakoba et al., 2023 [3].

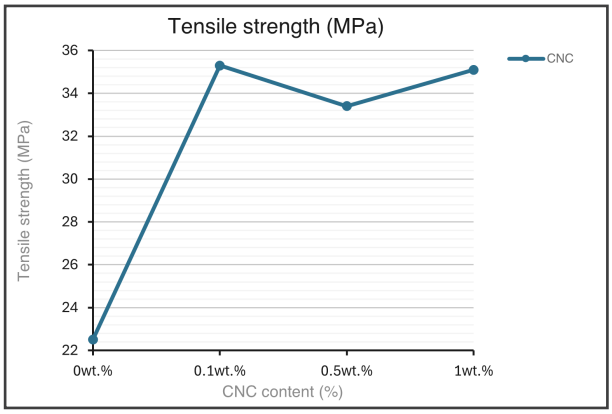


Figure A.101. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from Eucalyptus grandis sawdust) content in PLA matrix on the tensile strength. Adapted from "*Preparation of cellulose nanocrystal (CNCs) reinforced polylactic acid (PLA) bionanocomposites filaments using biobased additives for 3D printing applications*", by Agbakoba et al., 2023 [3].

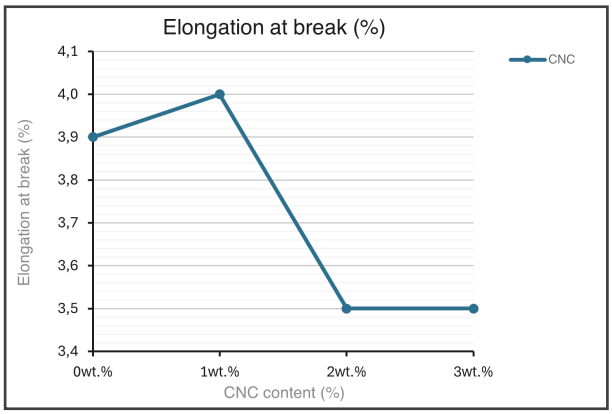


Figure A.102. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the elongation at break. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

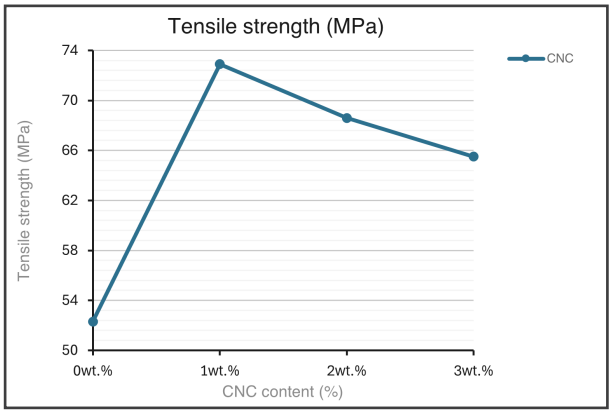


Figure A.103. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the tensile strength. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

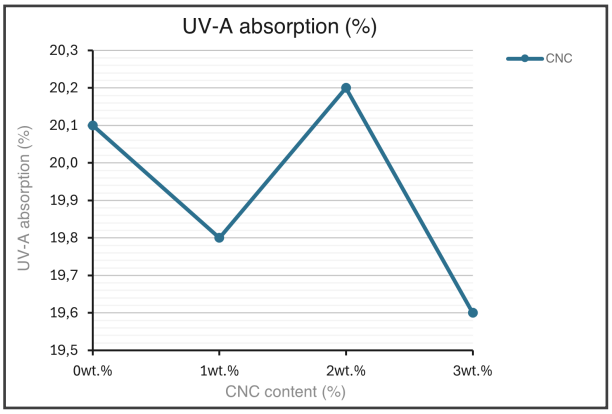


Figure A.104. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the UV-A absorption. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

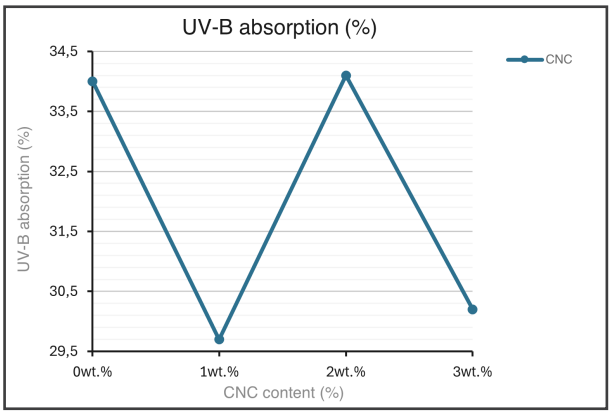


Figure A.105. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the UV-B absorption. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

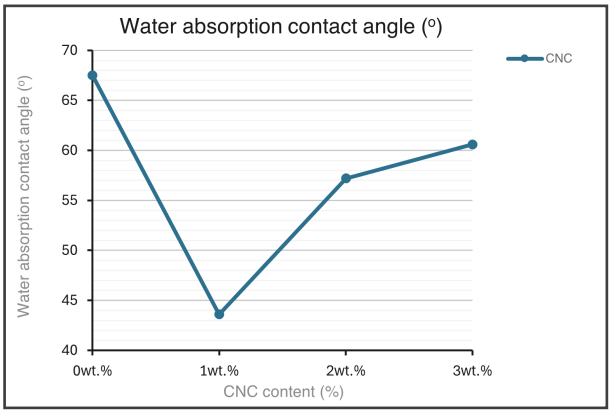


Figure A.106. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the water absorption contact angle. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

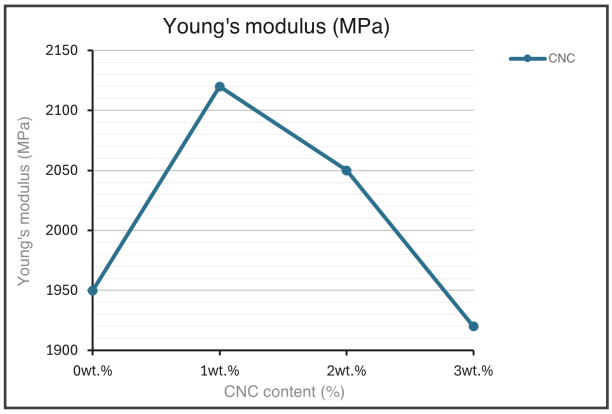


Figure A.107. Effect of cellulose nanocrystals (CNC extracted with acid hydrolysis from kraft paper) content in PLA matrix on the Young's modulus. Adapted from "*Potential of Cellulose Microfibers for PHA and PLA Biopolymers Reinforcement*", by Marmol et al., 2020 [72].

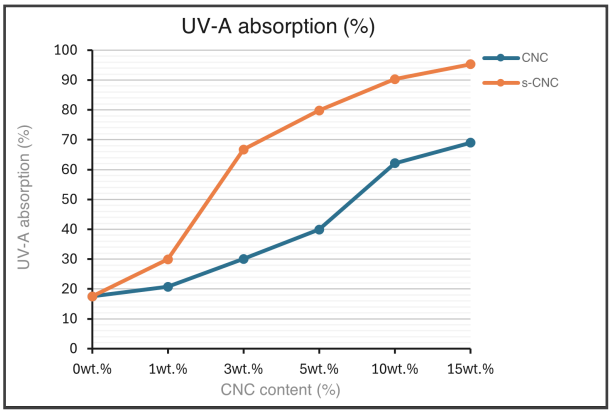


Figure A.110. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the UV-A absorption. Adapted from "*Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant absorption*", by Chi & Catchmark, 2017 [25].

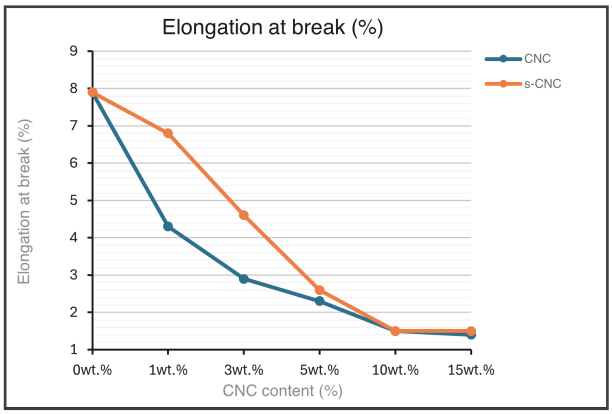


Figure A.108. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the elongation at break. Adapted from "*Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant absorption*", by Chi & Catchmark, 2017 [25].

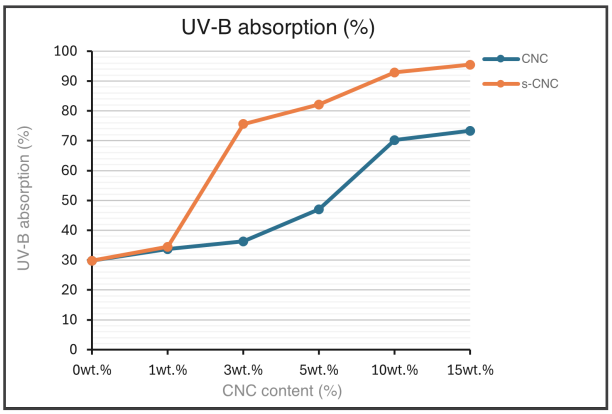


Figure A.111. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the UV-B absorption. Adapted from "*Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant absorption*", by Chi & Catchmark, 2017 [25].

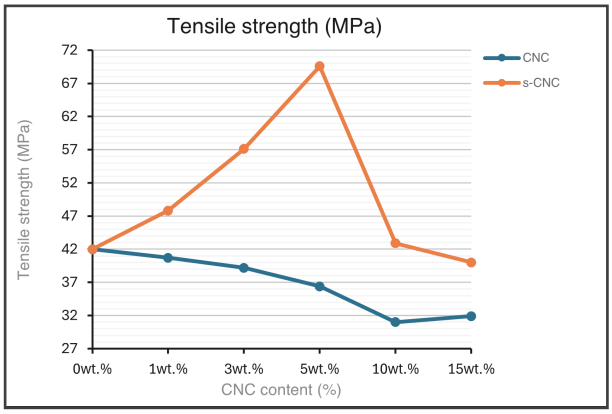


Figure A.109. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the tensile strength. Adapted from "*Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant absorption*", by Chi & Catchmark, 2017 [25].

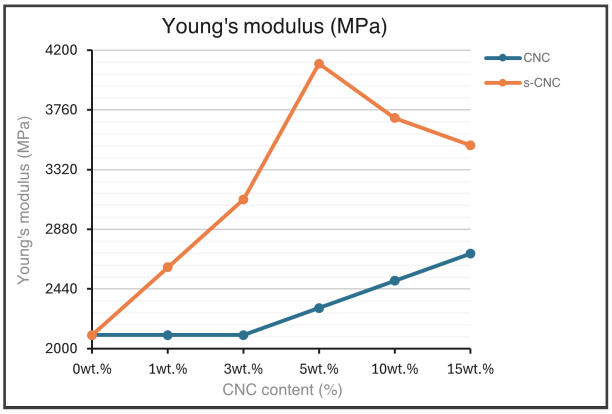


Figure A.112. Effect of cellulose nanocrystals content (blue: CNC extracted with acid hydrolysis from microcrystalline cellulose, and orange: surfactant-modified CNC with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {s-CNC}) in PLA matrix on the Young's modulus. Adapted from "*Enhanced dispersion and interface compatibilization of crystalline nanocellulose in polylactide by surfactant absorption*", by Chi & Catchmark, 2017 [25].

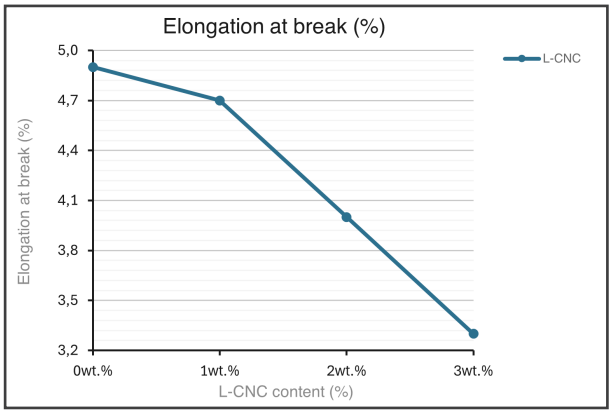


Figure A.113. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from wood pulp) content in PLA matrix on the elongation at break. Adapted from "*Green nanocomposites based on lignin coated cellulose nanocrystals and poly(lactic acid): crystallization, mechanical and thermal properties*", by Boruvka & Prusek, 2016 [20].

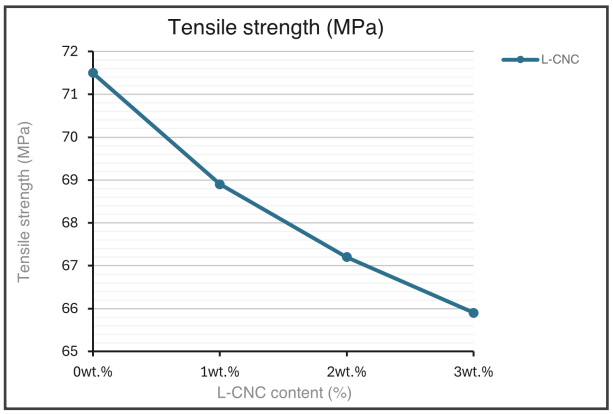


Figure A.114. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from wood pulp) content in PLA matrix on the tensile strength. Adapted from "*Green nanocomposites based on lignin coated cellulose nanocrystals and poly(lactic acid): crystallization, mechanical and thermal properties*", by Boruvka & Prusek, 2016 [20].

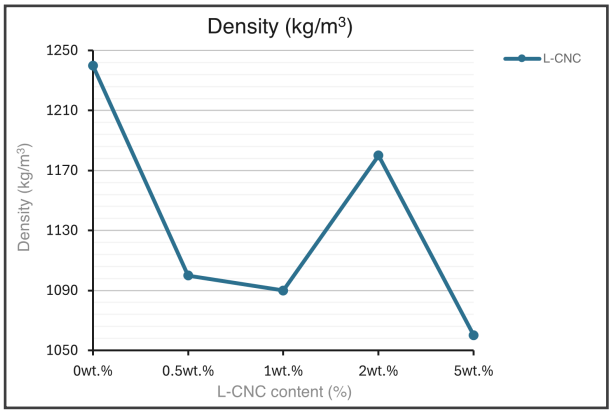


Figure A.115. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from aspen wood fibers) content in PLA matrix on the density. Adapted from "*Performance of high lignin content cellulose nanocrystals in poly(lactic acid)*", by Wei et al., 2017 [99].

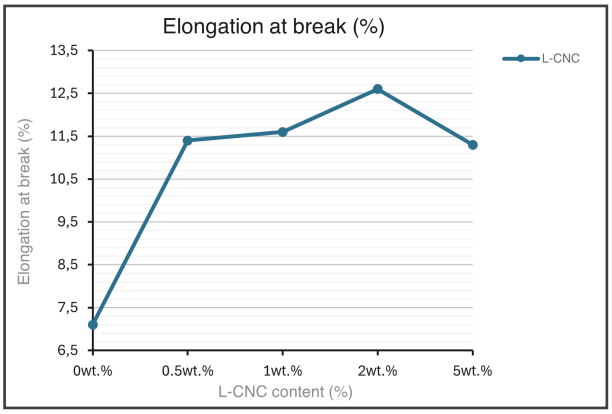


Figure A.116. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from aspen wood fibers) content in PLA matrix on the elongation at break. Adapted from "*Performance of high lignin content cellulose nanocrystals in poly(lactic acid)*", by Wei et al., 2017 [99].

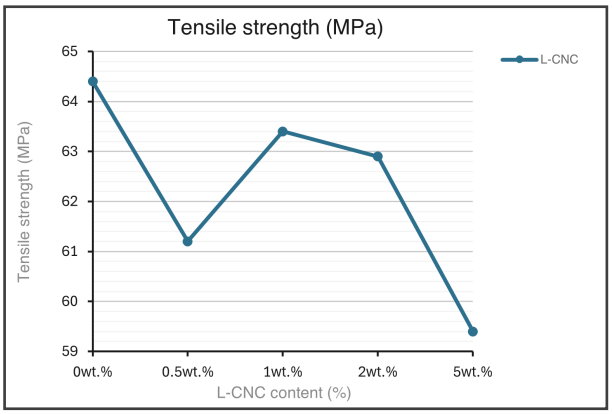


Figure A.117. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from aspen wood fibers) content in PLA matrix on the tensile strength. Adapted from "*Performance of high lignin content cellulose nanocrystals in poly(lactic acid)*", by Wei et al., 2017 [99].

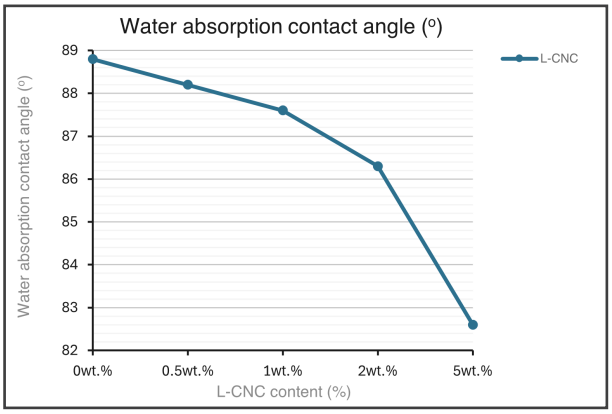


Figure A.118. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from aspen wood fibers) content in PLA matrix on the water absorption contact angle. Adapted from "*Performance of high lignin content cellulose nanocrystals in poly(lactic acid)*", by Wei et al., 2017 [99].

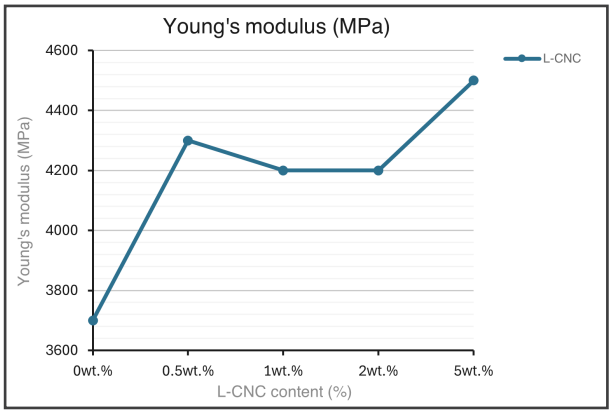


Figure A.119. Effect of lignin-coated cellulose nanocrystals (CNC extracted with acid hydrolysis from aspen wood fibers) content in PLA matrix on the Young's modulus. Adapted from "*Performance of high lignin content cellulose nanocrystals in poly(lactic acid)*", by Wei et al., 2017 [99].

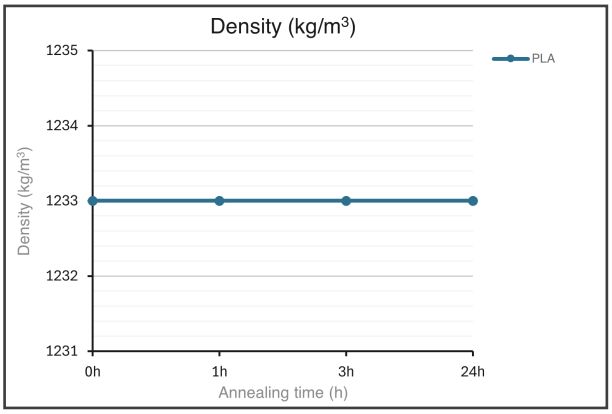


Figure A.122. Effect of PLA annealing time (h) on the density. Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

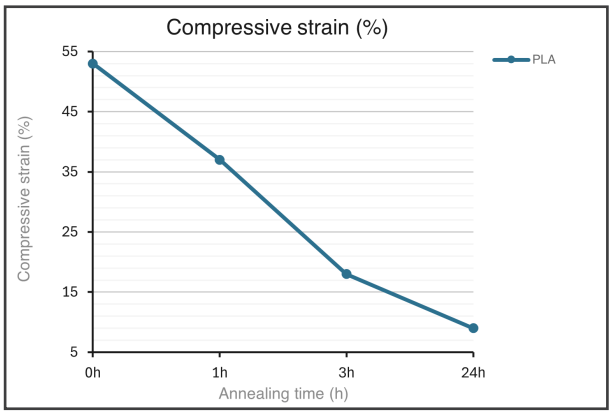


Figure A.120. Effect of PLA annealing time (h) on the compressive strain. Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

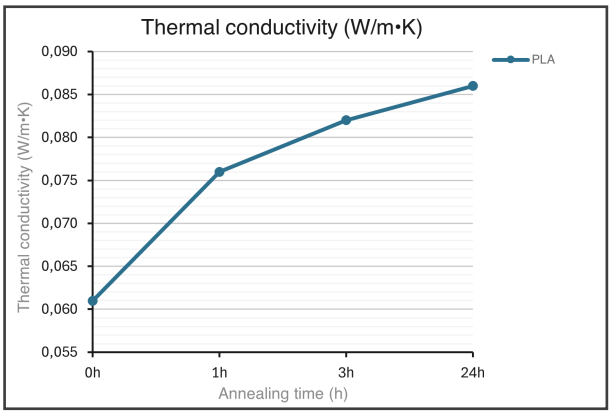


Figure A.123. Effect of PLA annealing time (h) on the thermal conductivity. Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

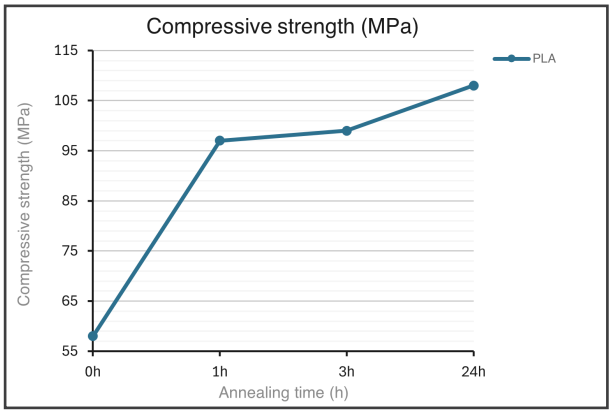


Figure A.121. Effect of PLA annealing time (h) on the compressive strength. Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

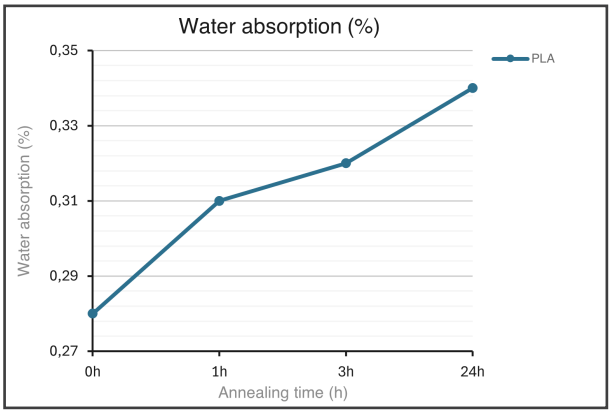


Figure A.124. Effect of PLA annealing time (h) on the water absorption (25°C / 200h). Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

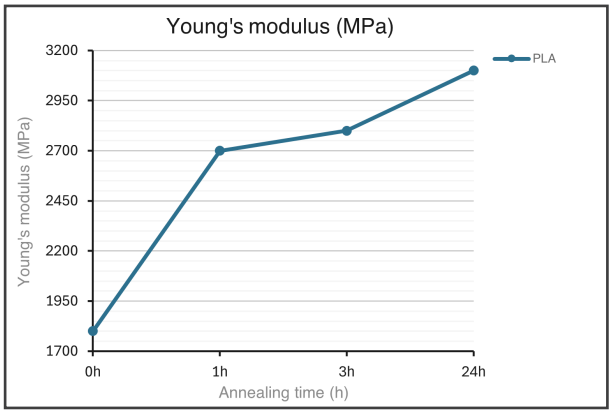


Figure A.125. Effect of PLA annealing time (h) on the compressive strain. Adapted from "*Thermal Insulation and Mechanical Properties of Polylactic Acid (PLA) at Different Processing Conditions*", by Barkhad et al., 2020 [14].

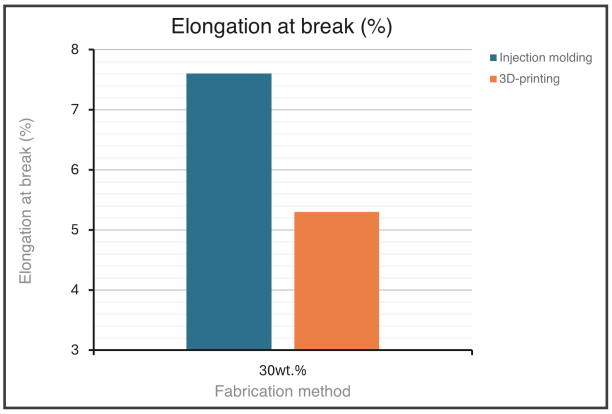


Figure A.128. Effect of fabrication method of PLA/30wt.% recycled pinewood particles on the elongation at break. Adapted from "*The Thermal and Mechanical Behavior of Wood-PLA Composites Processed by Additive Manufacturing for Building Insulation*", by Bahar et al., 2023 [10].

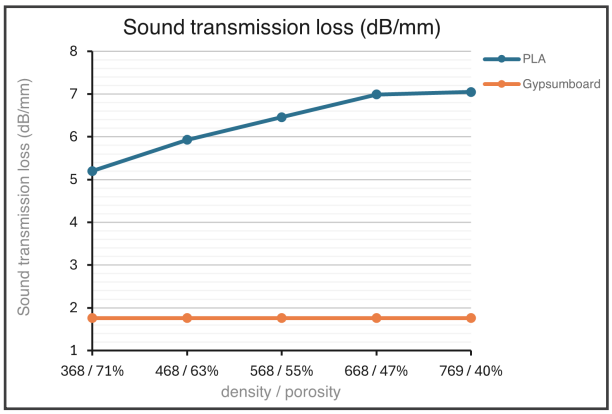


Figure A.126. Effect of PLA porosity on the sound transmission loss. Adapted from "*Thermal and acoustic performance evalation of 3D-Printable PLA materials*", by Islam et al., 2023 [49].

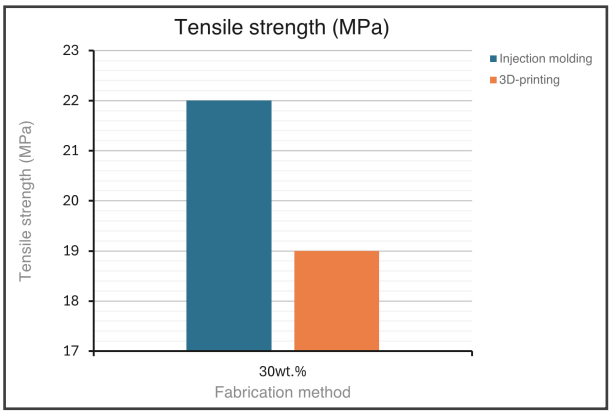


Figure A.129. Effect of fabrication method of PLA/30wt.% recycled pinewood particles on the tensile strength. Adapted from "*The Thermal and Mechanical Behavior of Wood-PLA Composites Processed by Additive Manufacturing for Building Insulation*", by Bahar et al., 2023 [10].

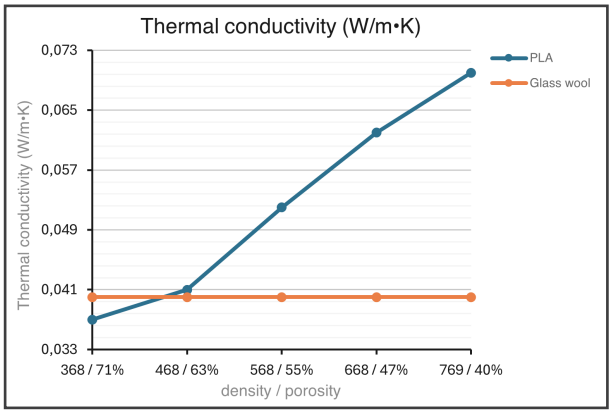


Figure A.127. Effect of PLA porosity on the thermal conductivity. Adapted from "*Thermal and acoustic performance evalation of 3D-Printable PLA materials*", by Islam et al., 2023 [49].

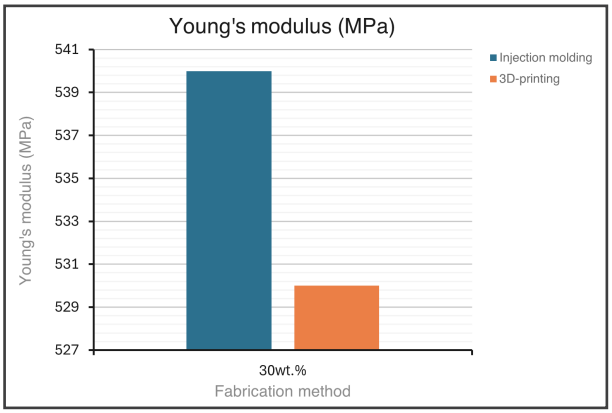


Figure A.130. Effect of fabrication method of PLA/30wt.% recycled pinewood particles on the Young's modulus. Adapted from "*The Thermal and Mechanical Behavior of Wood-PLA Composites Processed by Additive Manufacturing for Building Insulation*", by Bahar et al., 2023 [10].

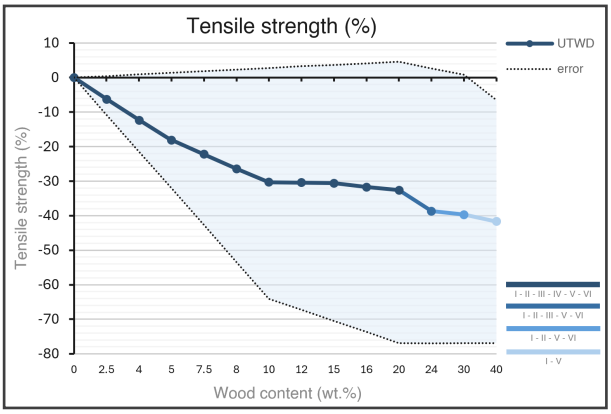


Figure A.131. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1]; II. Jasinski & Szymanowski, 2023 [51]; III. Csizmadia et al., 2013 [29]; IV. Narlioglu et al., 2021 [79]; V. Mohammadsalih et al., 2023 [77]; VI. Kelleci et al., 2022 [55].

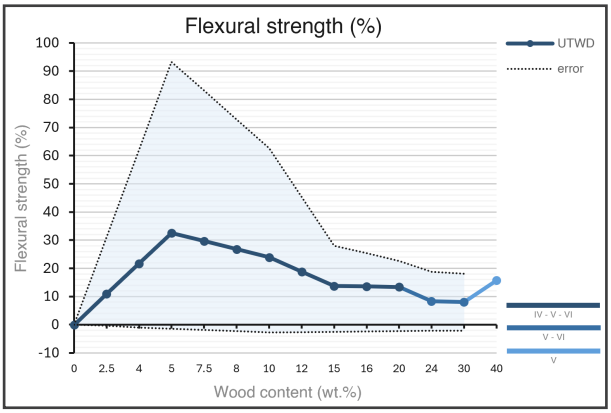


Figure A.132. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the flexural strength (relative % compared to neat PLA). Adapted from IV. Narlioglu et al., 2021 [79]; V. Mohammadsalih et al., 2023 [77]; VI. Kelleci et al., 2022 [55].

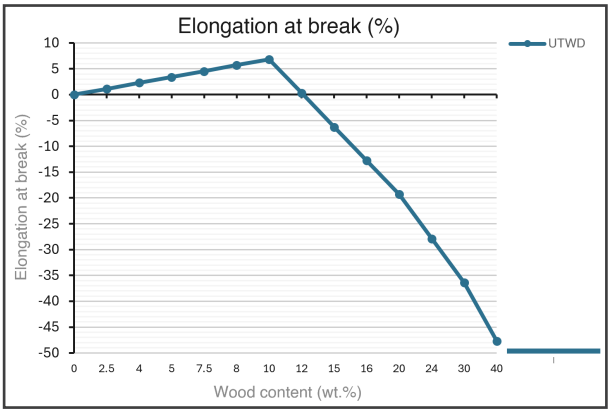


Figure A.133. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

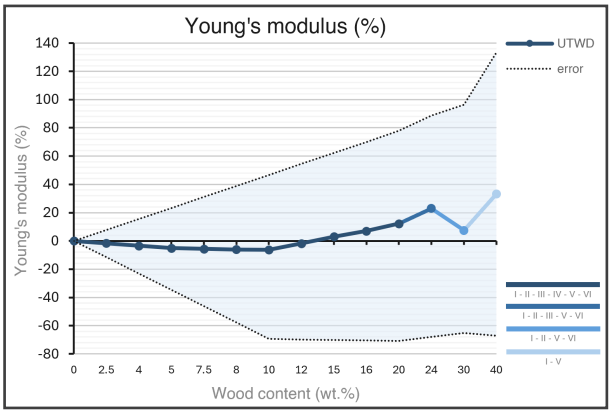


Figure A.134. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1]; II. Jasinski & Szymanowski, 2023 [51]; III. Csizmadia et al., 2013 [29]; IV. Narlioglu et al., 2021 [79]; V. Mohammadsalih et al., 2023 [77]; VI. Kelleci et al., 2022 [55].

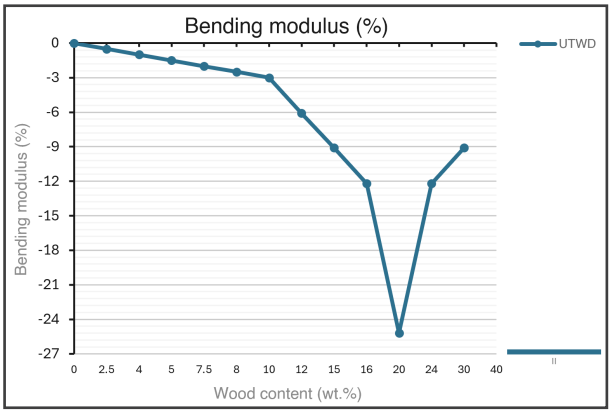


Figure A.135. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the bending modulus (relative % compared to neat PLA). Adapted from II. Jasinski & Szymanowski, 2023 [51].

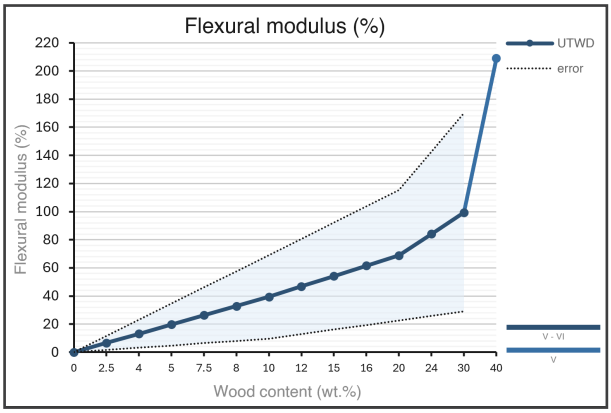


Figure A.136. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the flexural modulus (relative % compared to neat PLA). Adapted from V. Mohammadsalih et al., 2023 [77]; VI. Kelleci et al., 2022 [55].

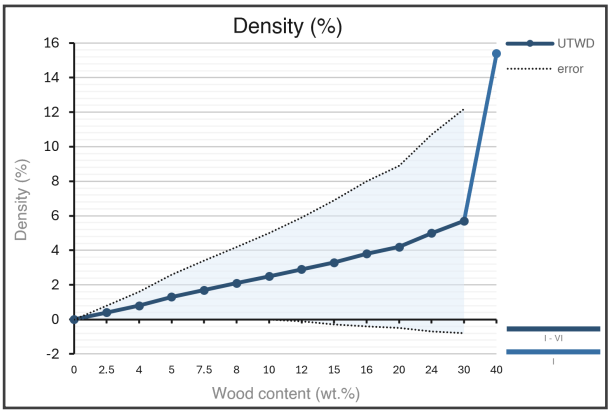


Figure A.137. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1]; VI. Kelleci et al., 2022 [55].

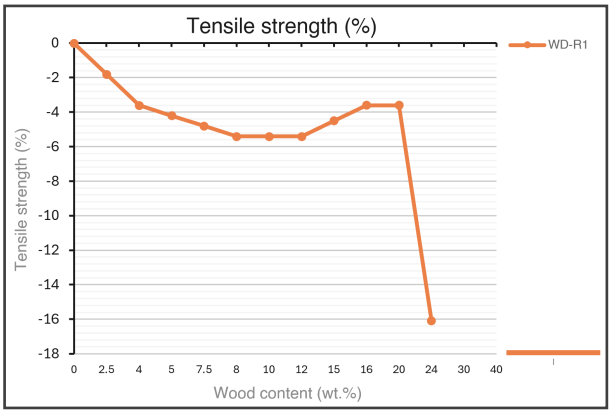


Figure A.140. Combined effect of 1wt.% phenolic resin-treated wood dust (WD-R1) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Csizmadia et al., 2013 [29].

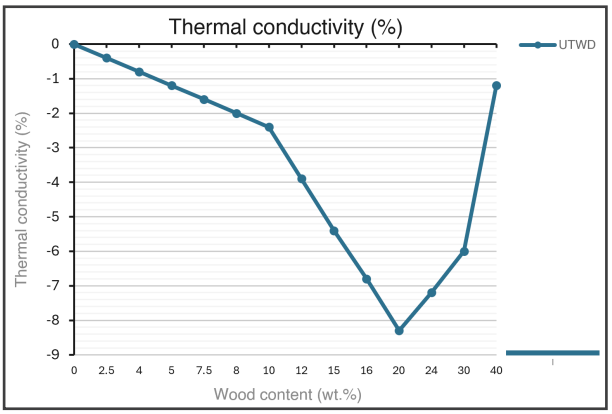


Figure A.138. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the thermal conductivity (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

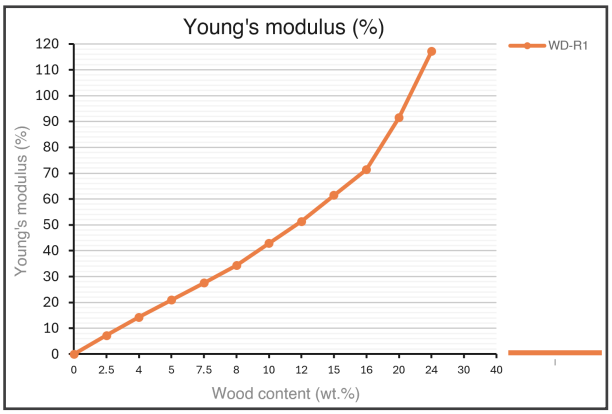


Figure A.141. Combined effect of 1wt.% phenolic resin-treated wood dust (WD-R1) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Csizmadia et al., 2013 [29].

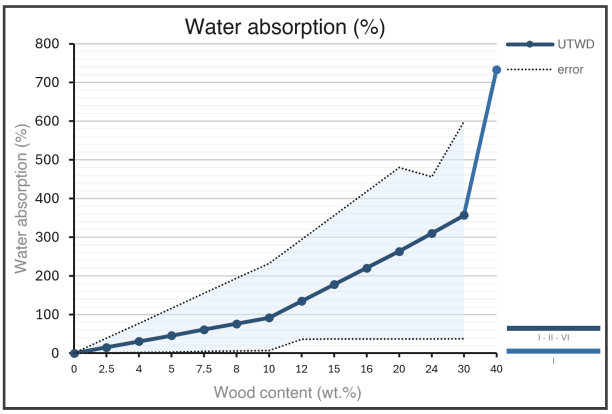


Figure A.139. Combined effect of untreated wood dust (UTWD) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1]; II. Jasinski & Szymanowski, 2023 [51]; VI. Kelleci et al., 2022 [55].

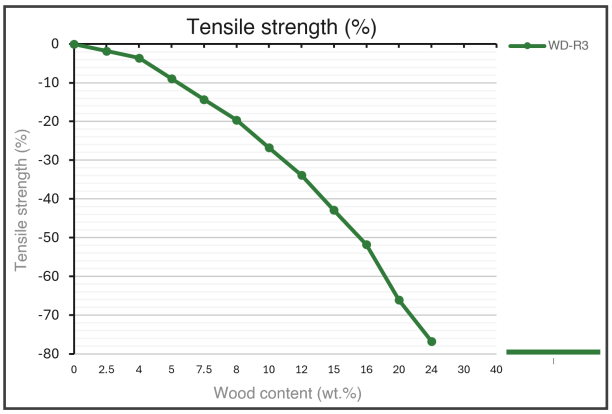


Figure A.142. Combined effect of 3wt.% phenolic resin-treated wood dust (WD-R3) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Csizmadia et al., 2013 [29].

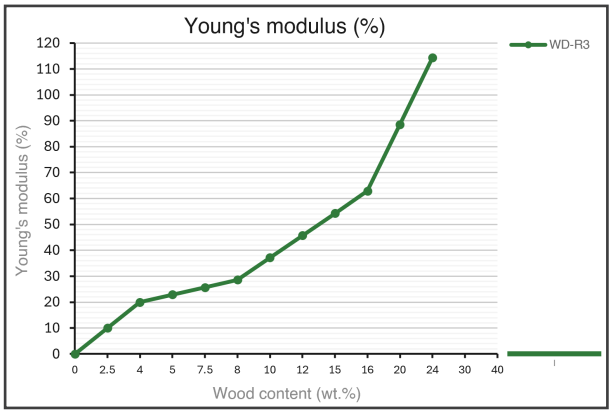


Figure A.143. Combined effect of 3wt.% phenolic resin-treated wood dust (WD-R3) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Csizmadia et al., 2013 [29].

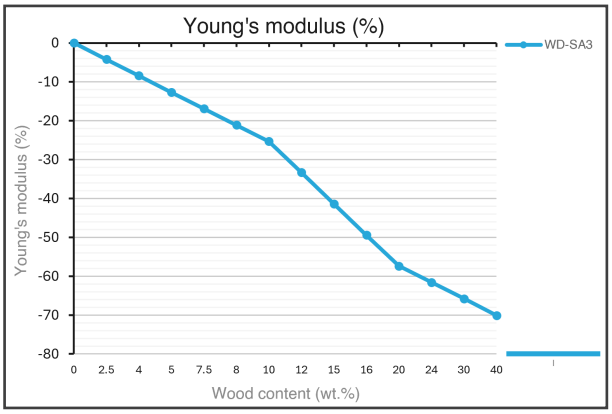


Figure A.146. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

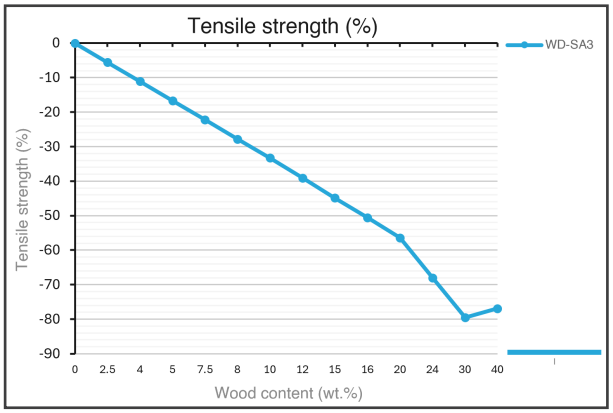


Figure A.144. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

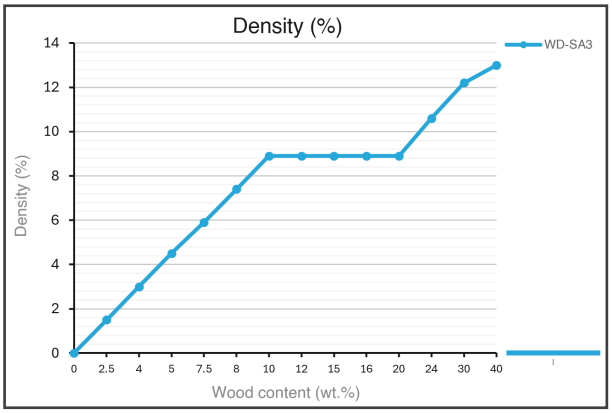


Figure A.147. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

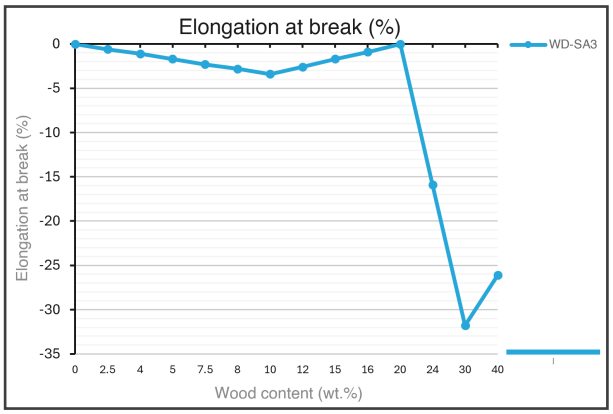


Figure A.145. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

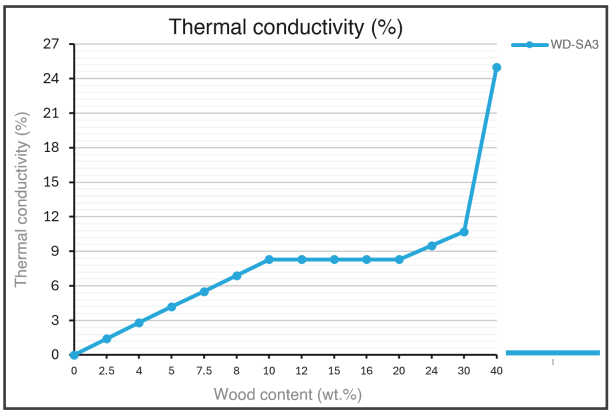


Figure A.148. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the thermal conductivity (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

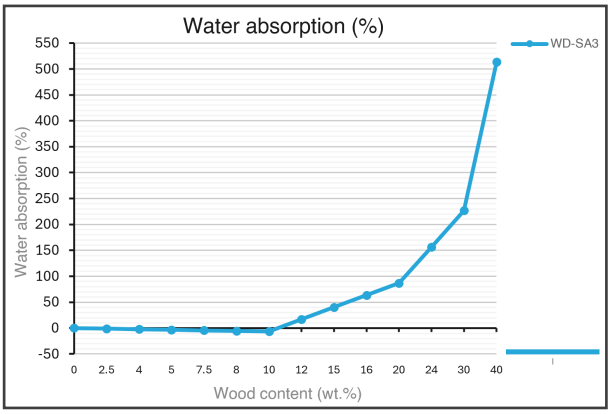


Figure A.149. Combined effect of 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio (WD-SA3) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

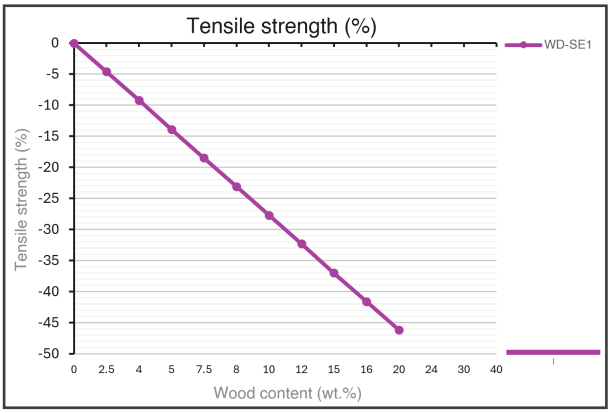


Figure A.150. Combined effect of 1wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE1) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

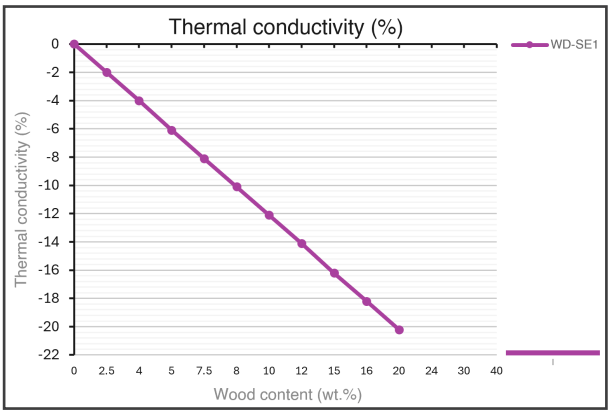


Figure A.151. Combined effect of 1wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE1) content in PLA matrix on the thermal conductivity (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

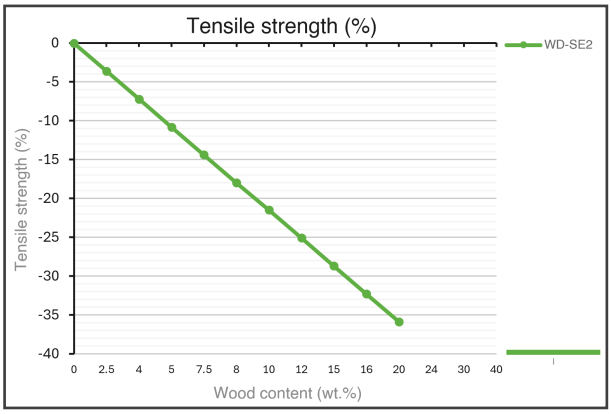


Figure A.152. Combined effect of 2wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE2) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

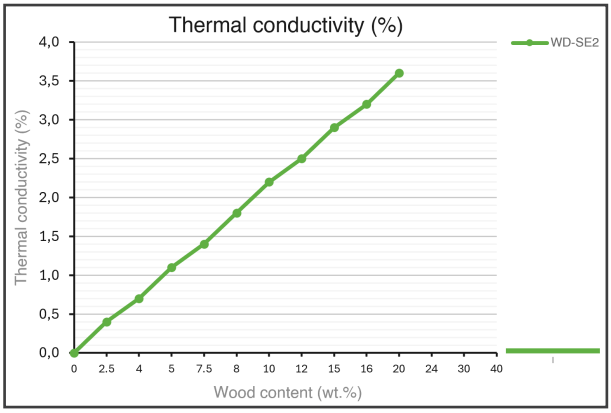


Figure A.153. Combined effect of 2wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE2) content in PLA matrix on the thermal conductivity (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

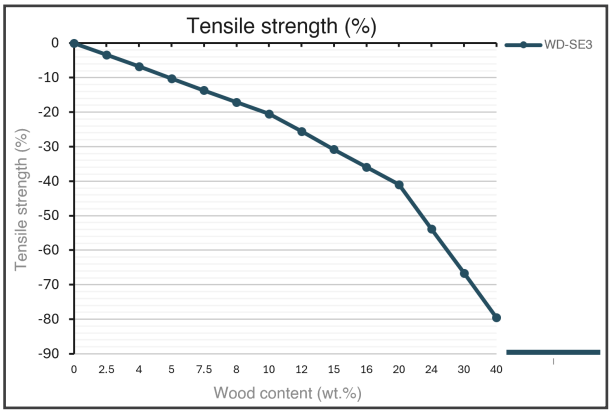


Figure A.154. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

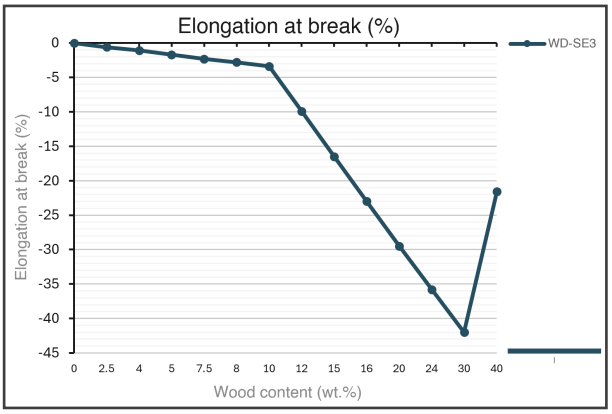


Figure A.155. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

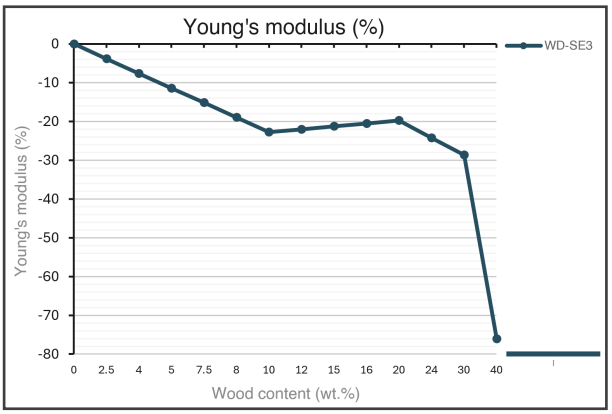


Figure A.156. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

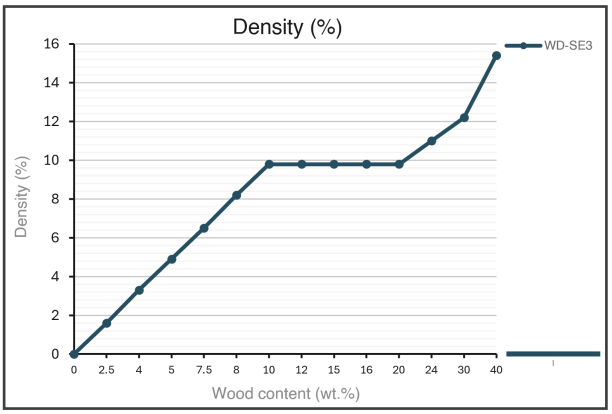


Figure A.157. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

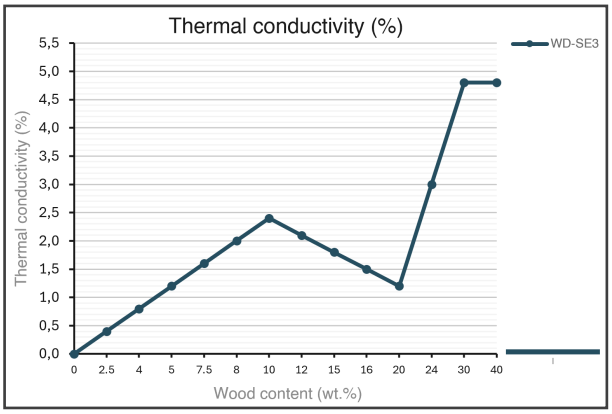


Figure A.158. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the thermal conductivity (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

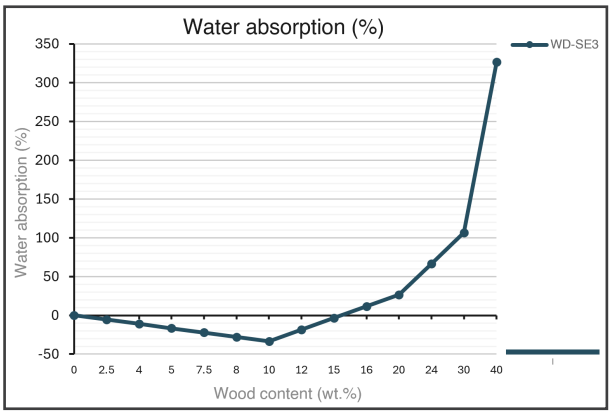


Figure A.159. Combined effect of 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio (WD-SE3) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Abdallah et al., 2022 [1].

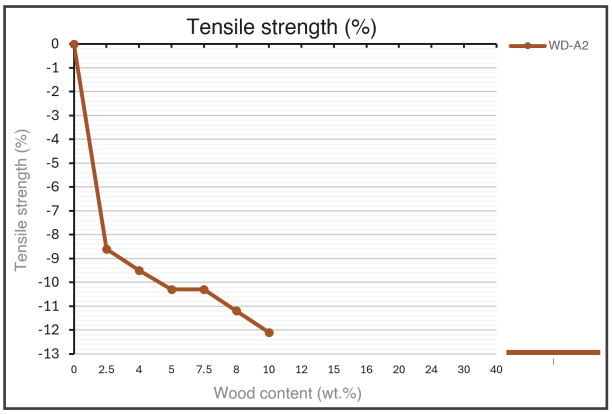


Figure A.160. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

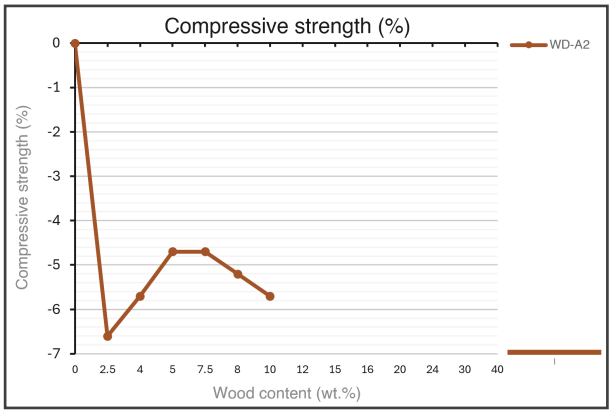


Figure A.161. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the compressive strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

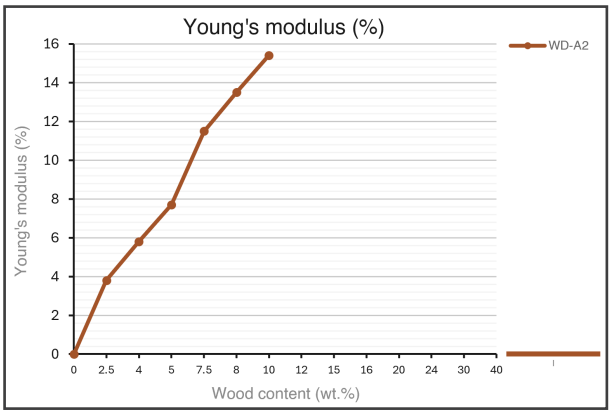


Figure A.164. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

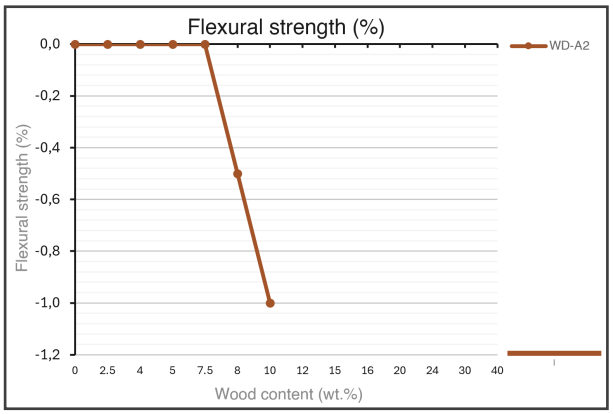


Figure A.162. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the flexural strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

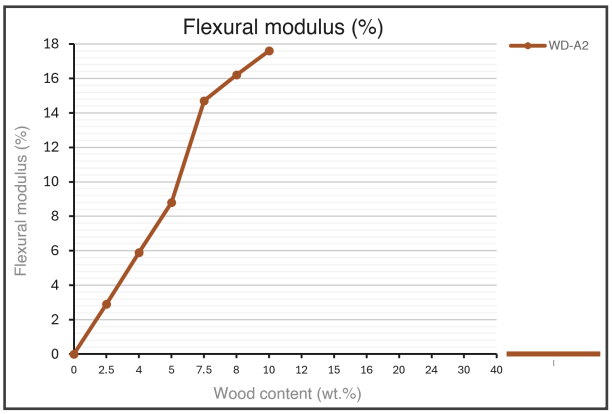


Figure A.165. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the flexural modulus (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

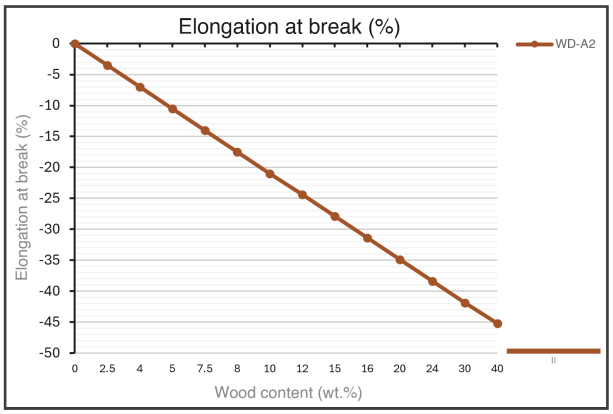


Figure A.163. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from II. Altun et al., 2012 [87].

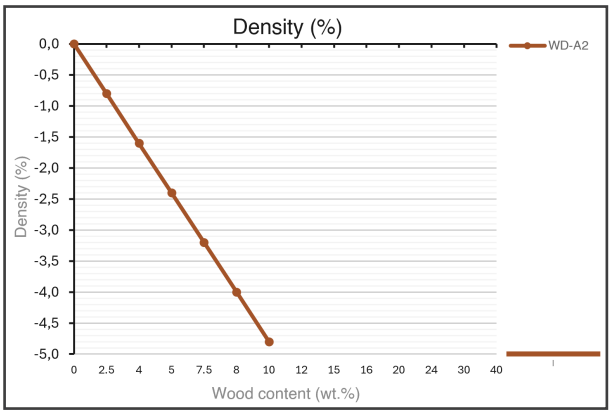


Figure A.166. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

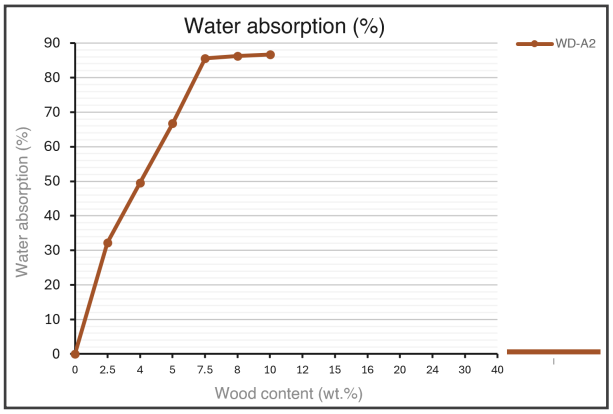


Figure A.167. Combined effect of 2wt.% Alkaline-treated wood dust (WD-A2) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

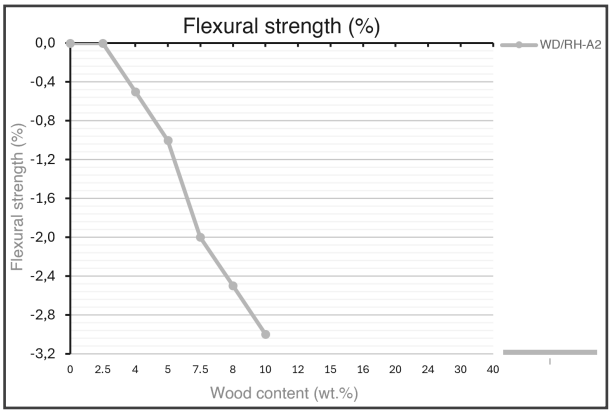


Figure A.170. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the flexural strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

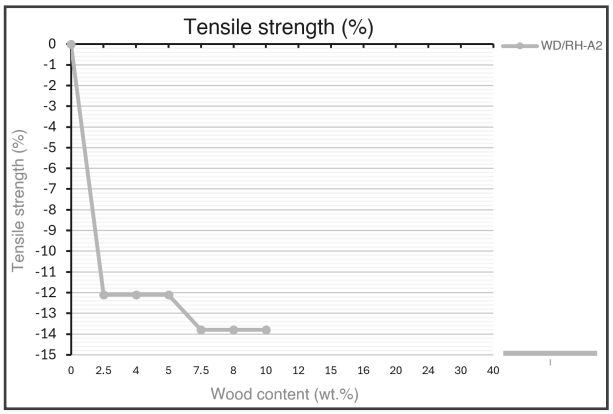


Figure A.168. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

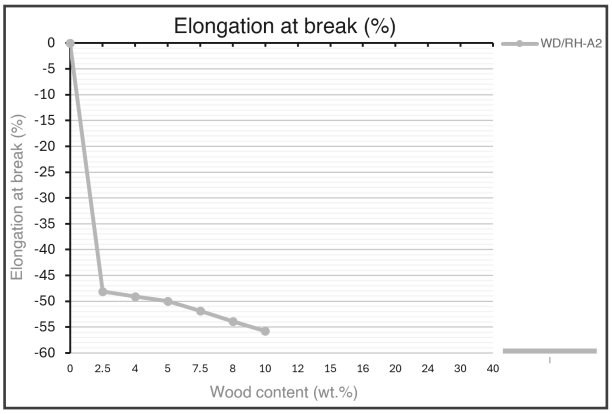


Figure A.171. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

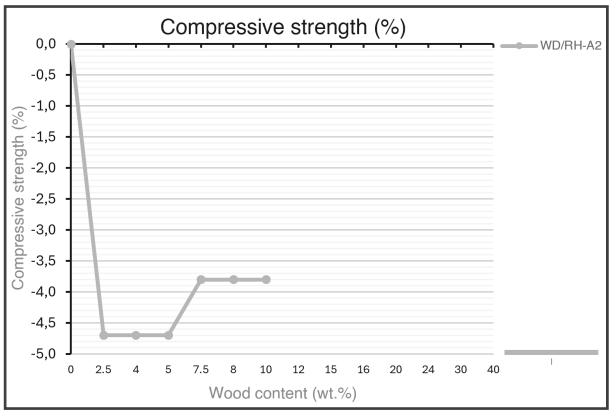


Figure A.169. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the compressive strength (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

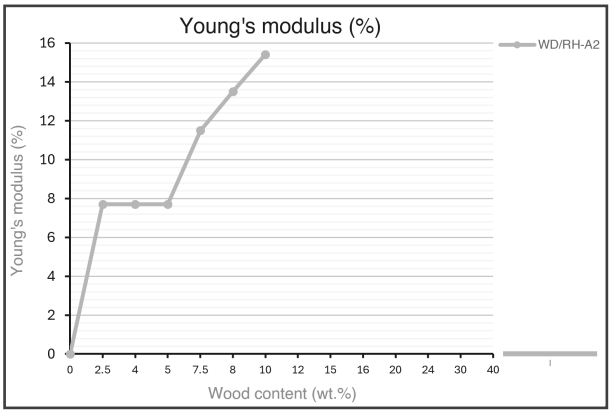


Figure A.172. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

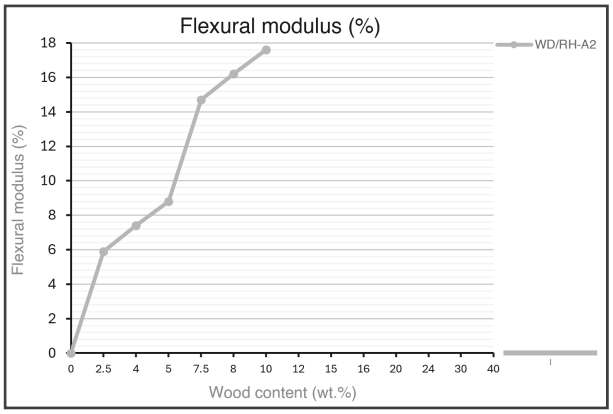


Figure A.173. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the flexural modulus (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

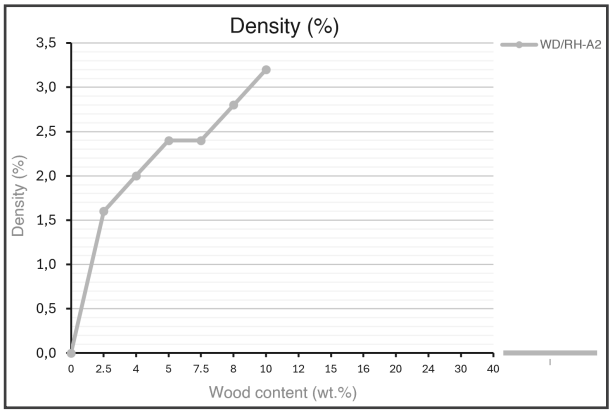


Figure A.174. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

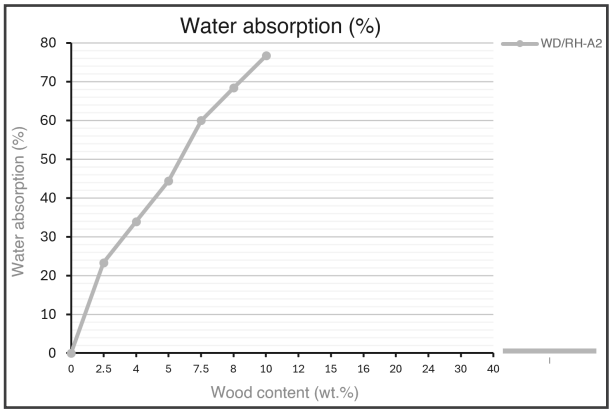


Figure A.175. Combined effect of 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio (WD/RH-A2) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Singh et al., 2022 [87].

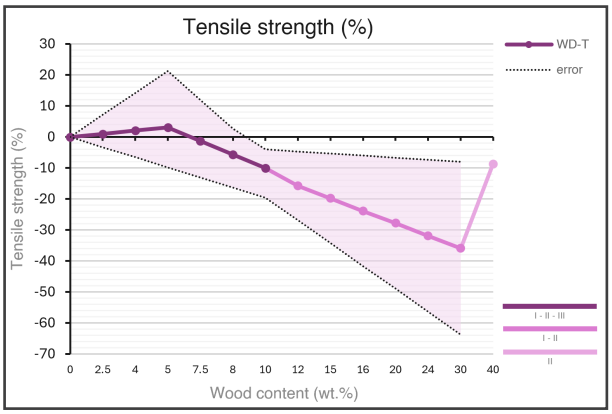


Figure A.176. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55]; II. Gregorova et al., 2011 [46]; III. Ayrilmis et al., 2021 [8].

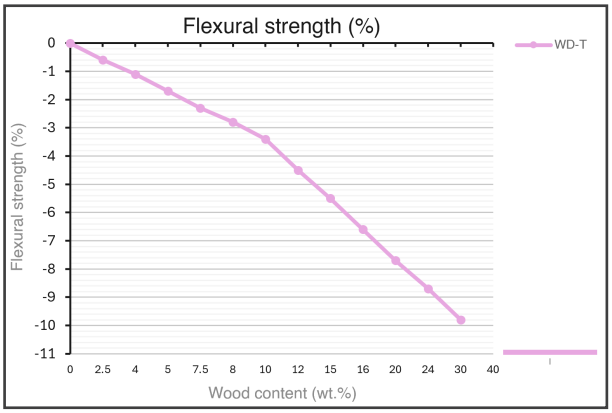


Figure A.177. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the flexural strength (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55].

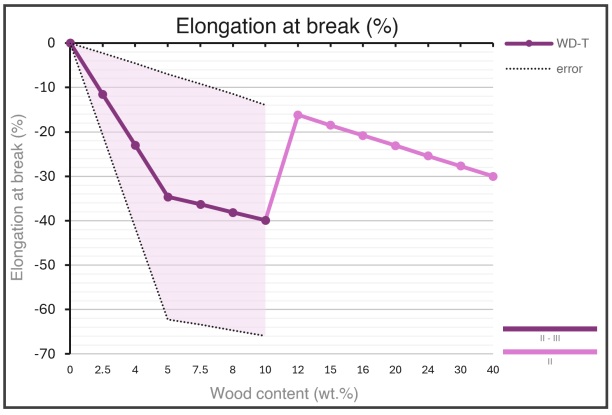


Figure A.178. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from II. Gregorova et al., 2011 [46]; III. Ayrilmis et al., 2021 [8].

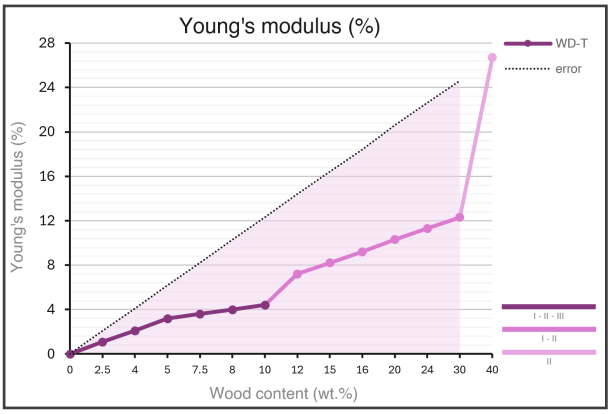


Figure A.179. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55]; II. Gregorova et al., 2011 [46]; III. Ayrilmis et al., 2021 [8].

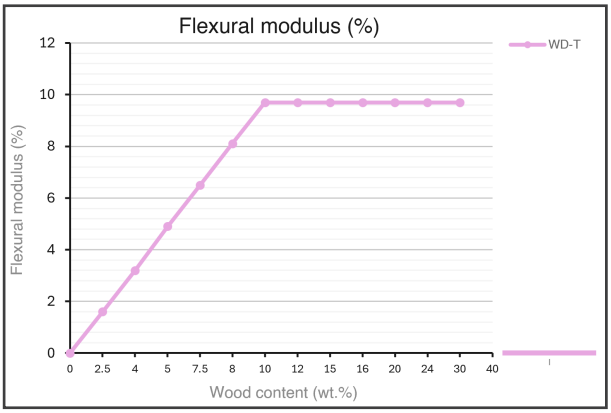


Figure A.180. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the flexural modulus (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55].

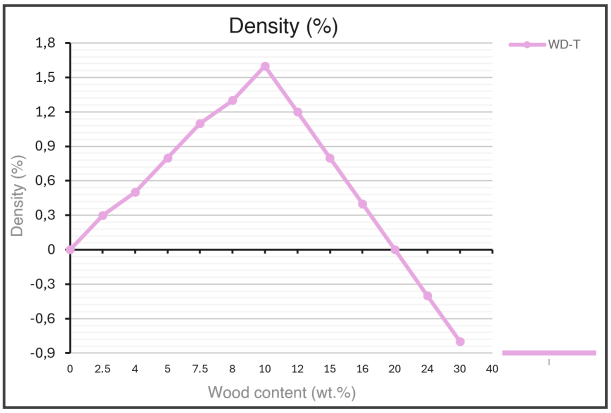


Figure A.181. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the density (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55].

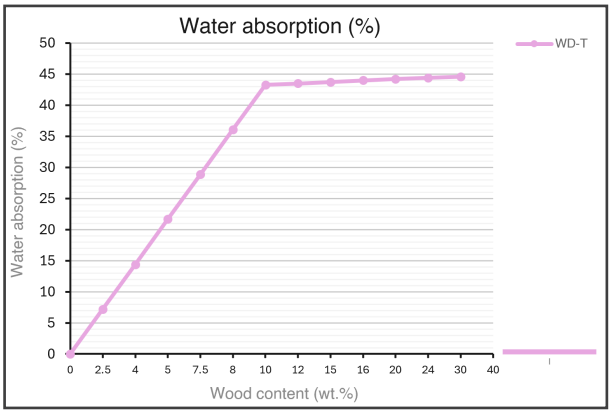


Figure A.182. Combined effect of thermally-treated wood dust (WD-T) content in PLA matrix on the water absorption (relative % compared to neat PLA). Adapted from I. Kelleci et al., 2022 [55].

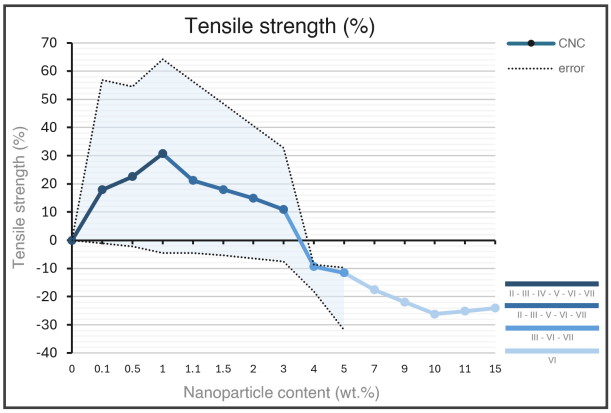


Figure A.183. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from II. Fortunati et al., 2015 [39]; III. Wei et al., 2016 [100]; IV. Agbakoba et al., 2023 [3]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88].

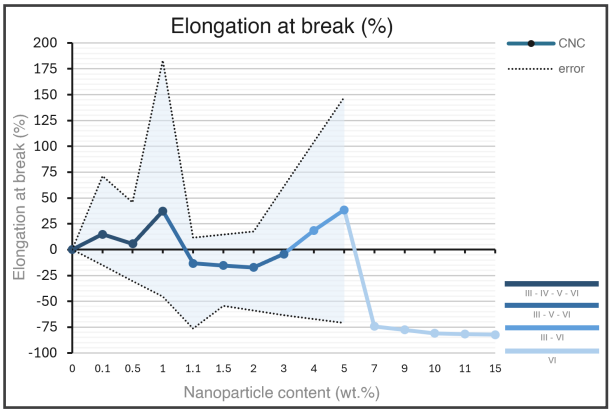


Figure A.184. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from III. Wei et al., 2016 [100]; IV. Agbakoba et al., 2023 [3]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25].

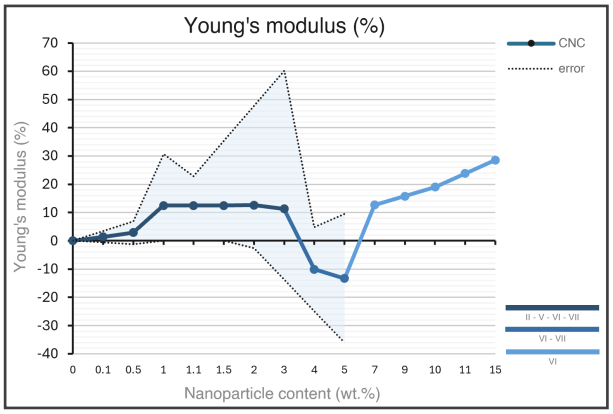


Figure A.185. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from II. Fortunati et al., 2015 [39]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88].

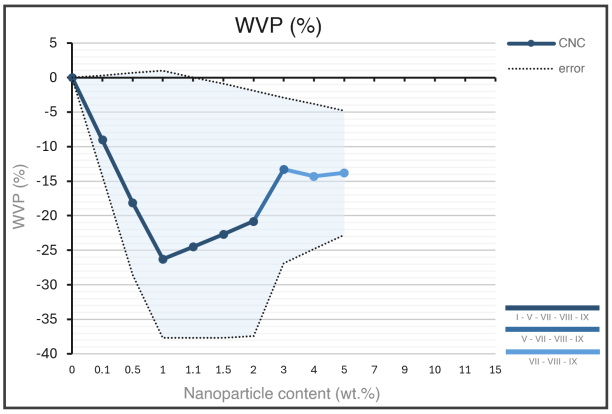


Figure A.188. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from I. Karkhanis et al., 2018 [54]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88]; VIII. Fortunati et al., 2012 [41]; IX. Fortunati et al., 2013 [40].

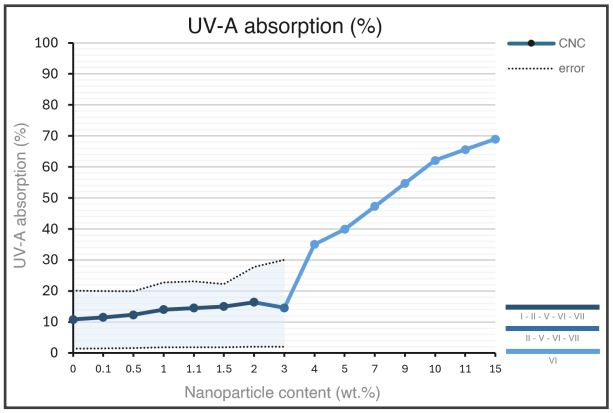


Figure A.186. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from I. Karkhanis et al., 2018 [54]; II. Fortunati et al., 2015 [39]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88].

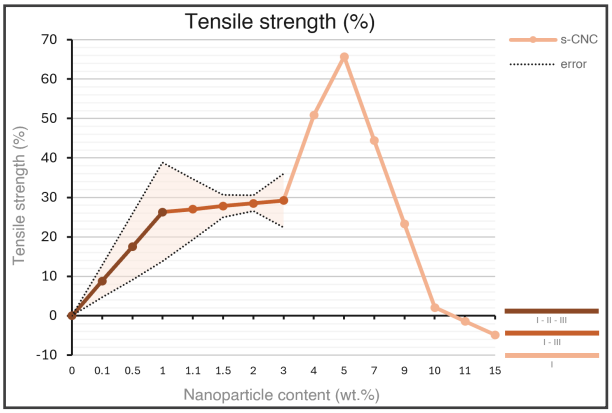


Figure A.189. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Chi & Catchmark, 2017 [25]; II. Luzi et al., 2016 [70]; III. Fortunati et al., 2015 [39].

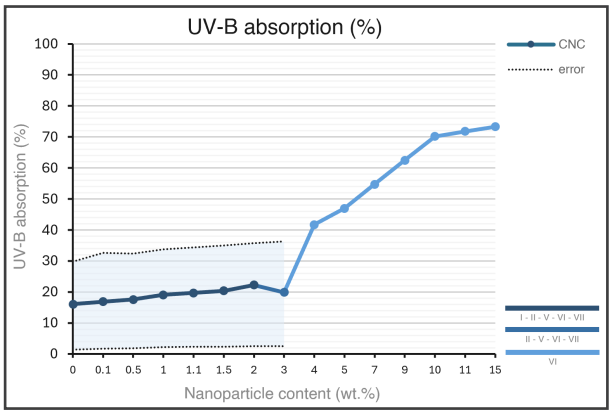


Figure A.187. Combined effect of cellulose nanocrystals (CNC) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from I. Karkhanis et al., 2018 [54]; II. Fortunati et al., 2015 [39]; V. Marmol et al., 2020 [72]; VI. Chi & Catchmark, 2017 [25]; VII. Sringam et al., 2023 [88].

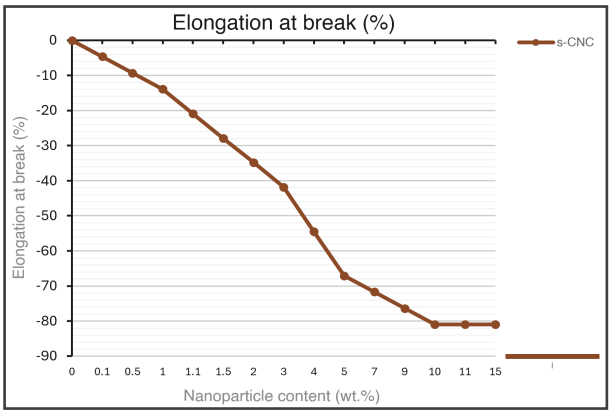


Figure A.190. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Chi & Catchmark, 2017 [25].

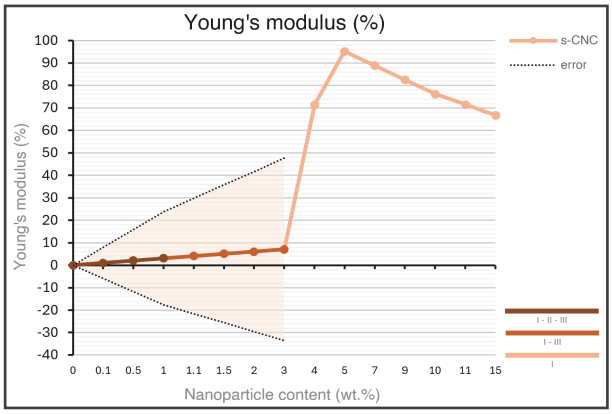


Figure A.191. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Chi & Catchmark, 2017 [25]; II. Luzi et al., 2016 [70]; III. Fortunati et al., 2015 [39].

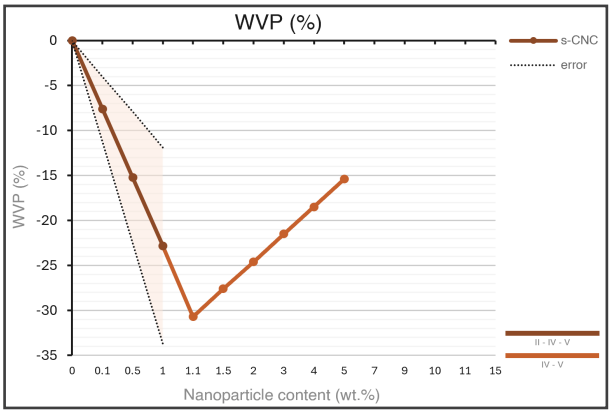


Figure A.194. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from II. Luzi et al., 2016 [70]; IV. Fortunati et al., 2012 [41]; V. Fortunati et al., 2013 [40].

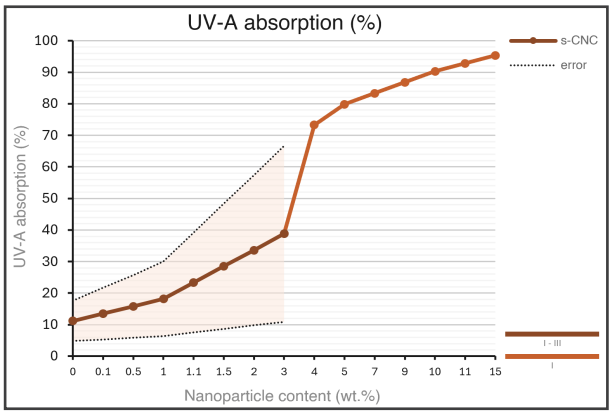


Figure A.192. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from I. Chi & Catchmark, 2017 [25]; III. Fortunati et al., 2015 [39].

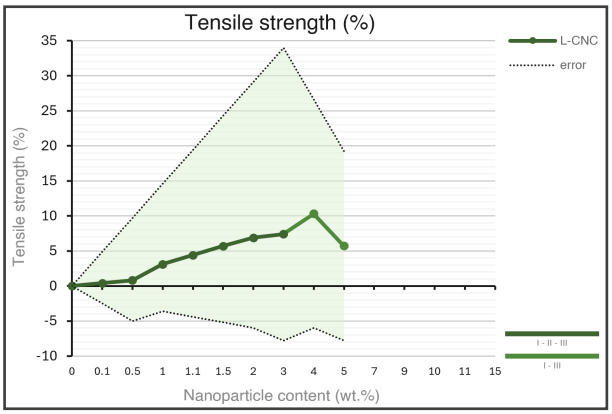


Figure A.195. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Wei et al., 2017 [99]; II. Boruvka & Prusek, 2016 [20]; III. Shojaeiarani et al., 2022 [85].

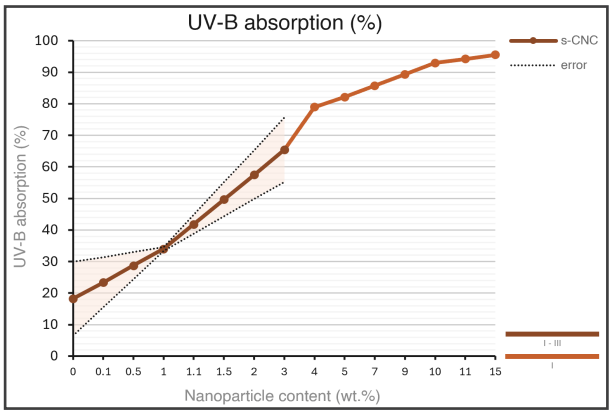


Figure A.193. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from I. Chi & Catchmark, 2017 [25]; III. Fortunati et al., 2015 [39].

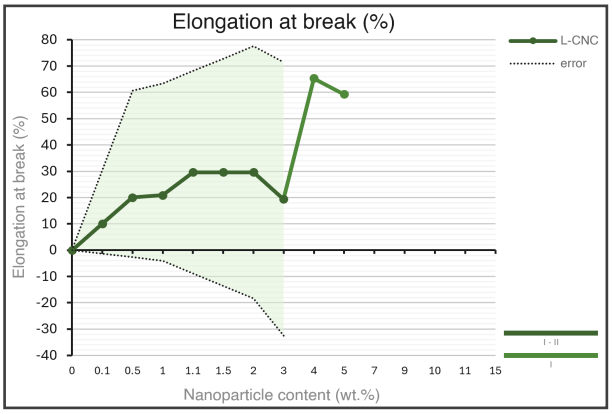


Figure A.196. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Wei et al., 2017 [99]; II. Boruvka & Prusek, 2016 [20].

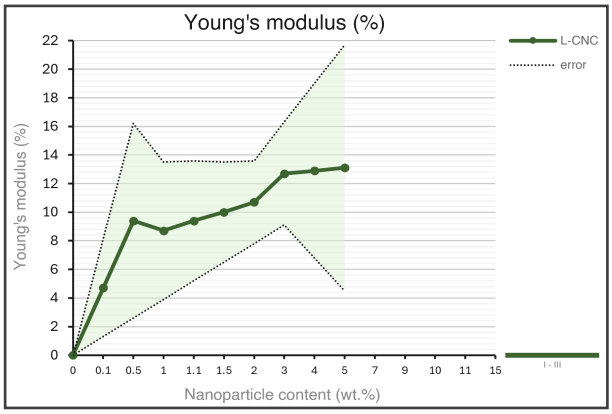


Figure A.197. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Wei et al., 2017 [99]; III. Shojaeiarani et al., 2022 [85].

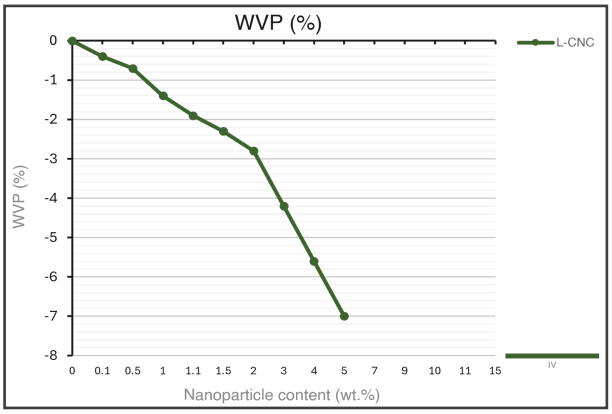


Figure A.200. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from IV. Sun et al., 2023 [89].

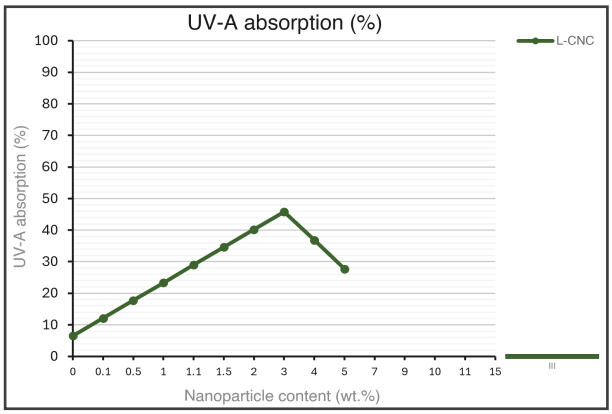


Figure A.198. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from III. Shojaeiarani et al., 2022 [85].

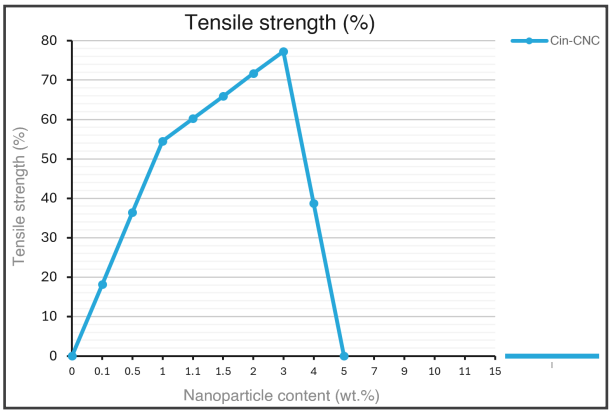


Figure A.201. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

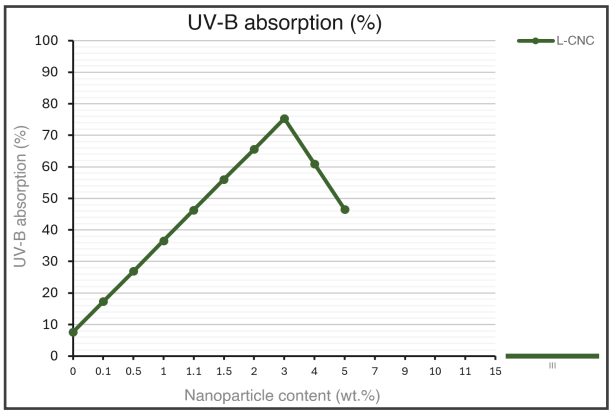


Figure A.199. Combined effect of lignin-coated cellulose nanocrystals (L-CNC) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from III. Shojaeiarani et al., 2022 [85].

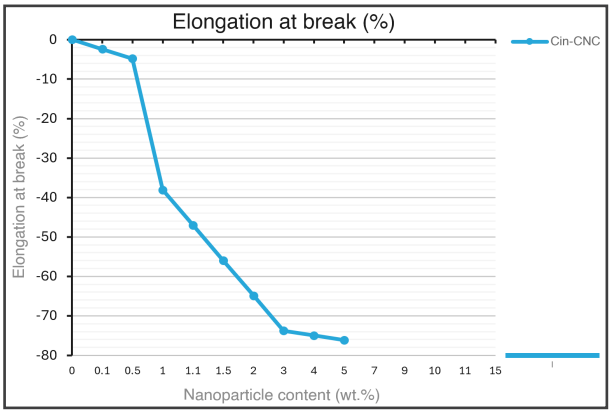


Figure A.202. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

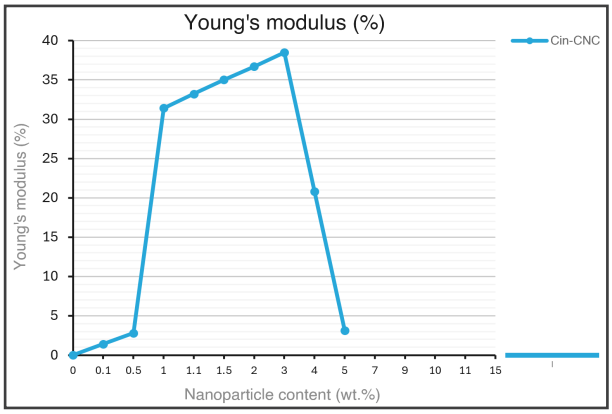


Figure A.203. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

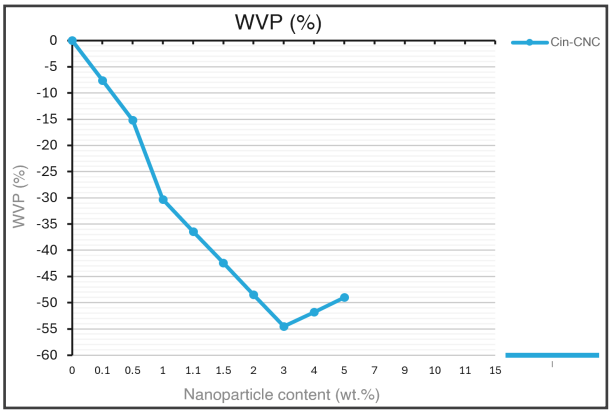


Figure A.206. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

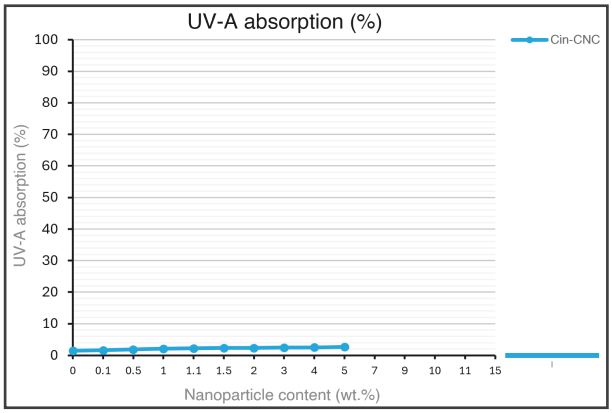


Figure A.204. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

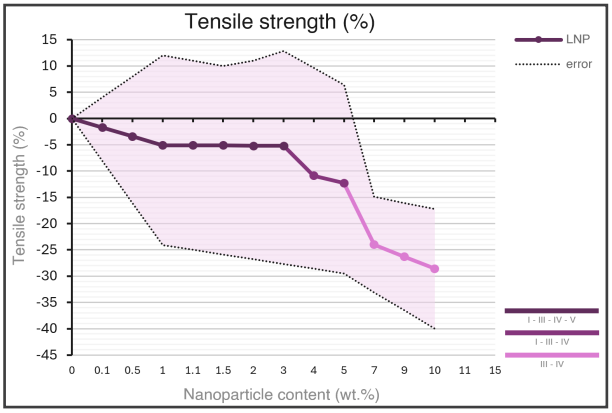


Figure A.207. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Shojaeiarani et al., 2022 [85]; III. Boarino et al., 2022 [19]; IV. Daassi et al., 2023 [30]; V. Cavallo et al., 2021 [24].

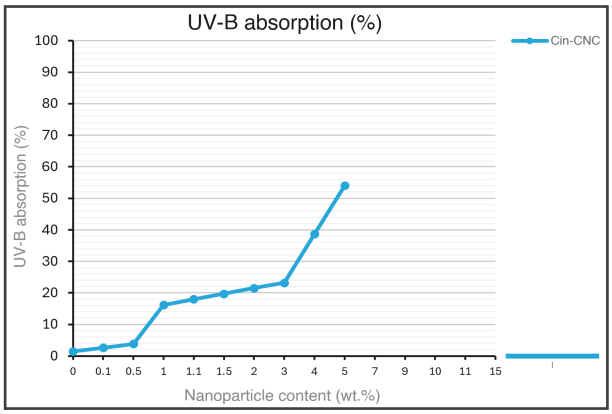


Figure A.205. Combined effect of cinnamate-grafted cellulose nanocrystals (Cin-CNC) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from I. Sringam et al., 2023 [88].

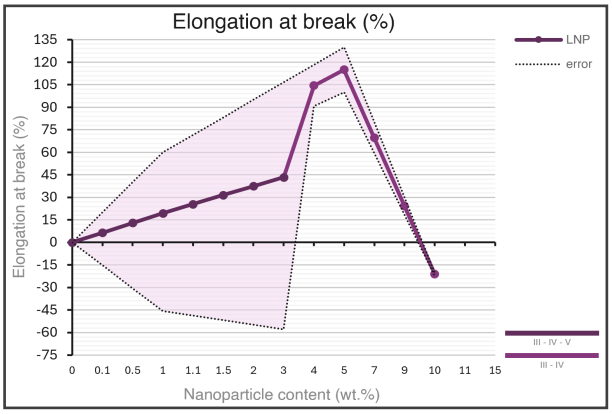


Figure A.208. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from III. Boarino et al., 2022 [19]; IV. Daassi et al., 2023 [30]; V. Cavallo et al., 2021 [24].

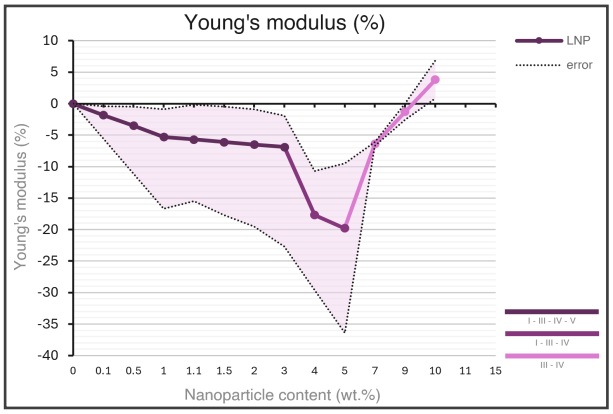


Figure A.209. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Shojaeiarani et al., 2022 [85]; III. Boarino et al., 2022 [19]; IV. Daassi et al., 2023 [30]; V. Cavallo et al., 2021 [24].

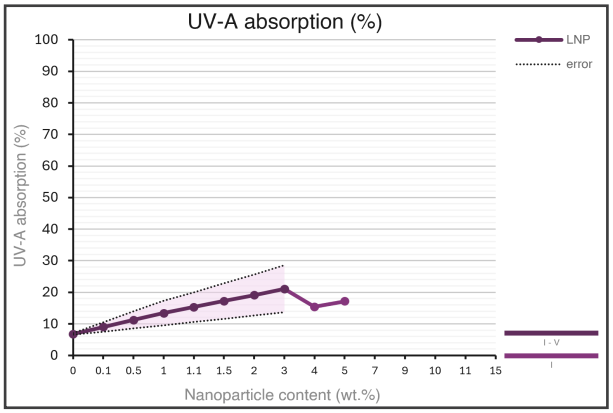


Figure A.210. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from I. Shojaeiarani et al., 2022 [85]; V. Cavallo et al., 2021 [24].

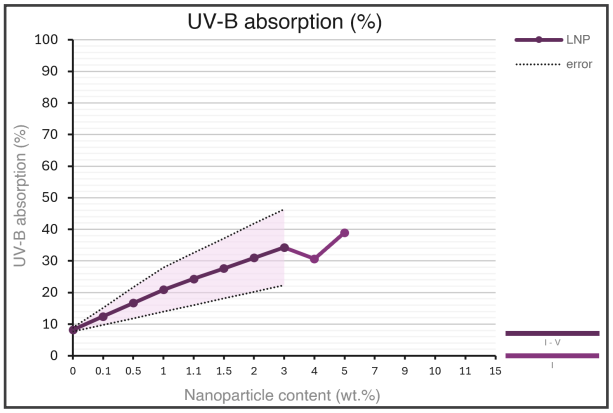


Figure A.211. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from I. Shojaeiarani et al., 2022 [85]; V. Cavallo et al., 2021 [24].

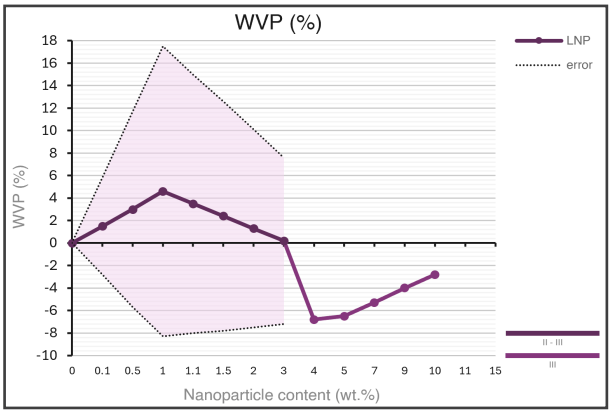


Figure A.212. Combined effect of lignin nanoparticles (LNP) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from II. Yang et al., 2015 [103]; III. Boarino et al., 2022 [19].

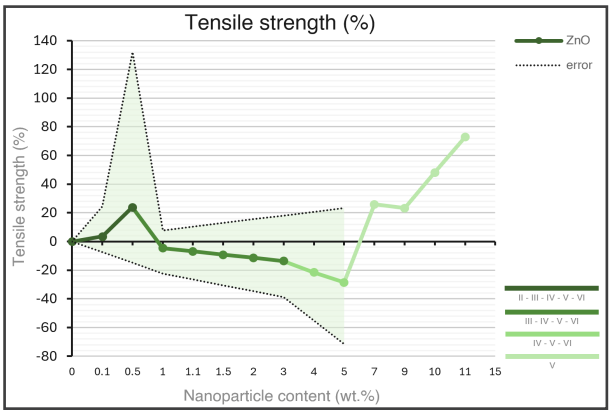


Figure A.213. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from II. Luzi et al., 2016 [70]; III. Yang et al., 2023 [104]; IV. Chong et al., 2023 [26]; V. Tan et al., 2023 [90]; VI. Jamnongkan et al., 2022 [50].

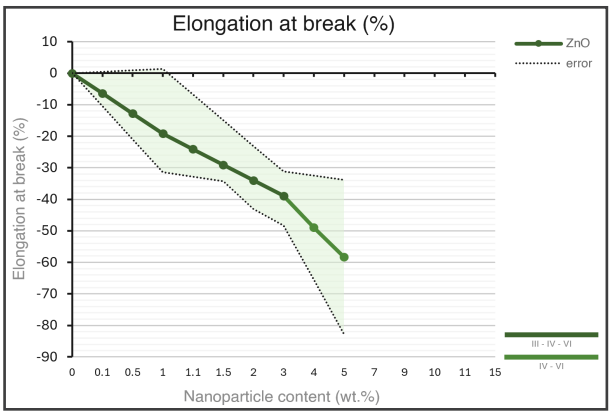


Figure A.214. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from III. Yang et al., 2023 [104]; IV. Chong et al., 2023 [26]; VI. Jamnongkan et al., 2022 [50].

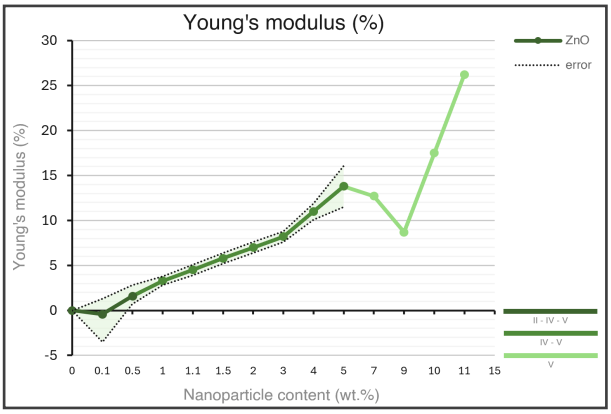


Figure A.215. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from II. Luzi et al., 2016 [70]; IV. Chong et al., 2023 [26]; V. Tan et al., 2023 [90].

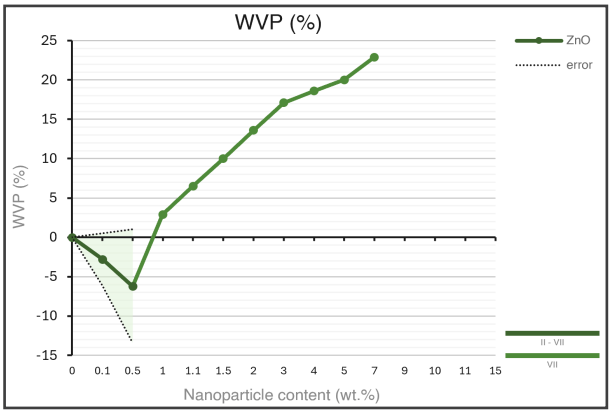


Figure A.218. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from II. Luzi et al., 2016 [70]; VII. Kim et al., 2019 [57].

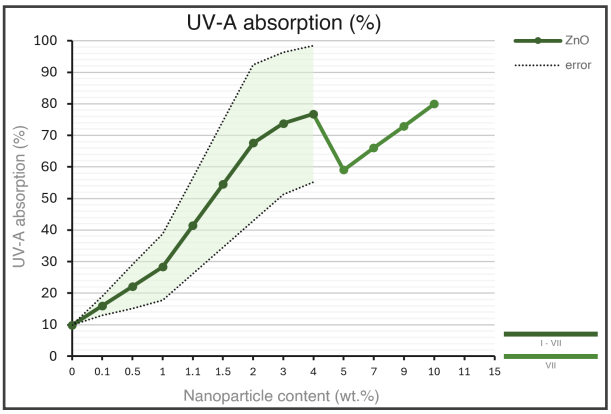


Figure A.216. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from I. Li et al., 2023 [63]; VII. Kim et al., 2019 [57].

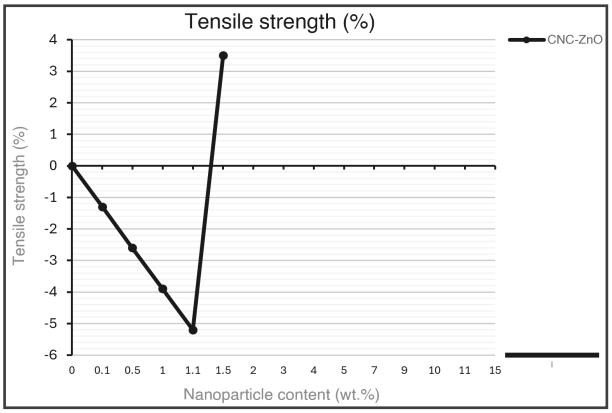


Figure A.219. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the tensile strength (relative % compared to neat PLA). Adapted from I. Luzi et al., 2016 [70].

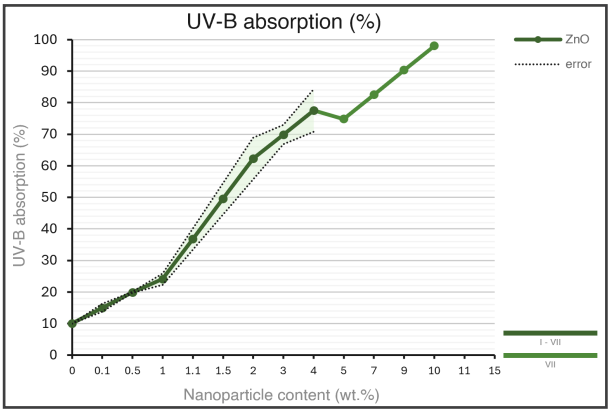


Figure A.217. Combined effect of Zinc oxide (ZnO) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from I. Li et al., 2023 [63]; VII. Kim et al., 2019 [57].

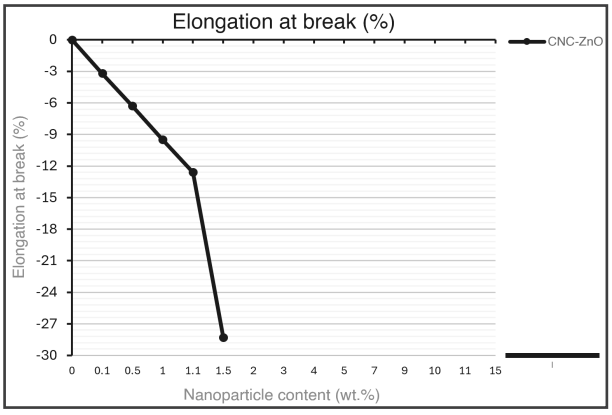


Figure A.220. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the elongation at break (relative % compared to neat PLA). Adapted from I. Luzi et al., 2016 [70].

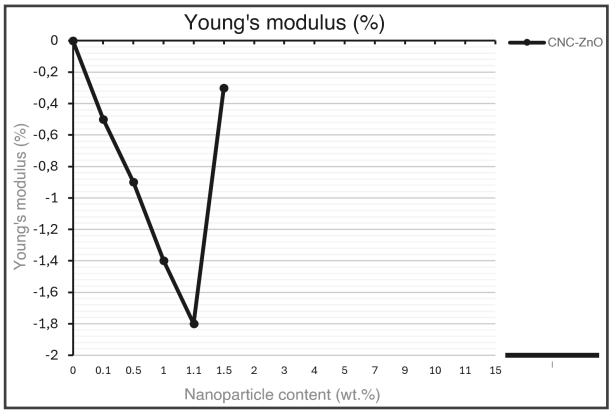


Figure A.221. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the Young's modulus (relative % compared to neat PLA). Adapted from I. Luzi et al., 2016 [70].

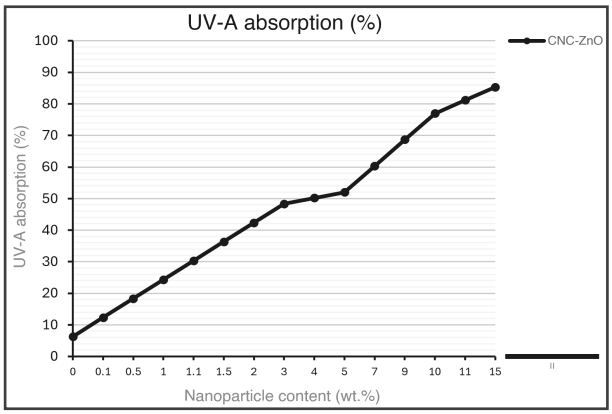


Figure A.222. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the UV-A absorption (relative % compared to neat PLA). Adapted from II. Wang et al., 2019 [97].

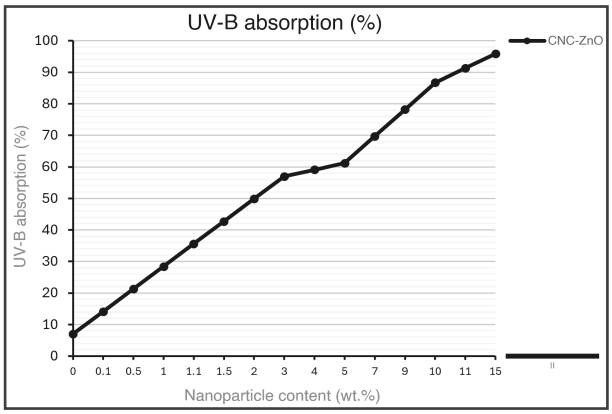


Figure A.223. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the UV-B absorption (relative % compared to neat PLA). Adapted from II. Wang et al., 2019 [97].

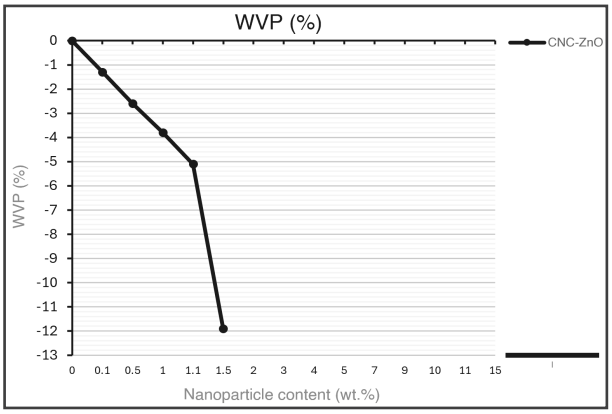


Figure A.224. Combined effect of cellulose nanocrystals-Zinc oxide (CNC-ZnO) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from I. Luzi et al., 2016 [70].

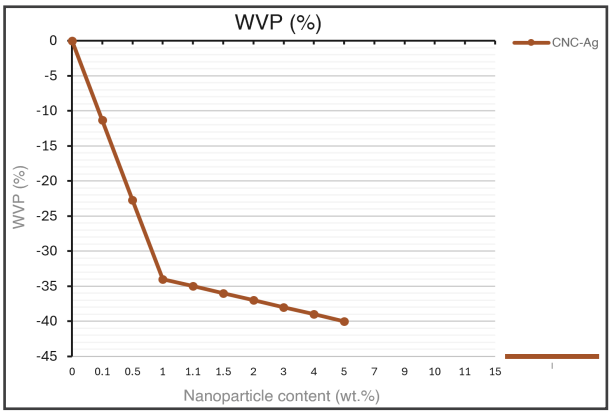


Figure A.225. Combined effect of cellulose nanocrystals + 1wt.% silver nanopowder (CNC-Ag) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from I. Fortunati et al., 2012 [41].

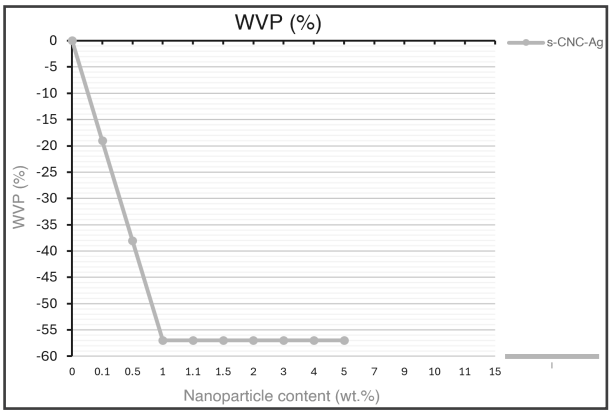


Figure A.226. Combined effect of surfactant-modified cellulose nanocrystals (CNC modified with acid phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio) + 1wt.% silver nanopowder (s-CNC-Ag) content in PLA matrix on the water vapour permeability (relative % compared to neat PLA). Adapted from I. Fortunati et al., 2013 [40].

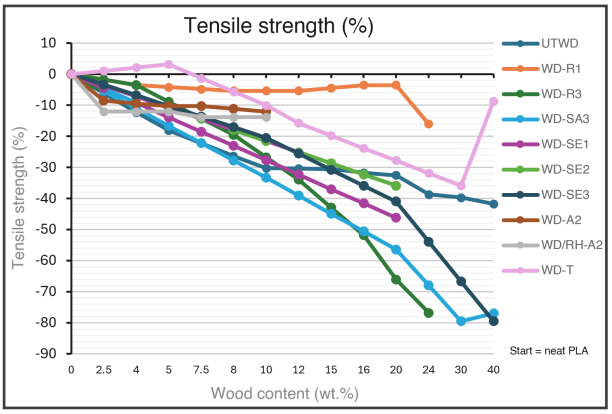


Figure A.227. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]; [51]; [29]; [79]; [77]; and [55]}, WD-R1/WD-R3: 1/3wt.% phenolic resin-treated wood dust {[29]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, WD-SE1/WD-SE2/WD-SE3: 1/2/3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]; [46]; and [8]}) on the tensile strength (relative % compared to neat PLA).

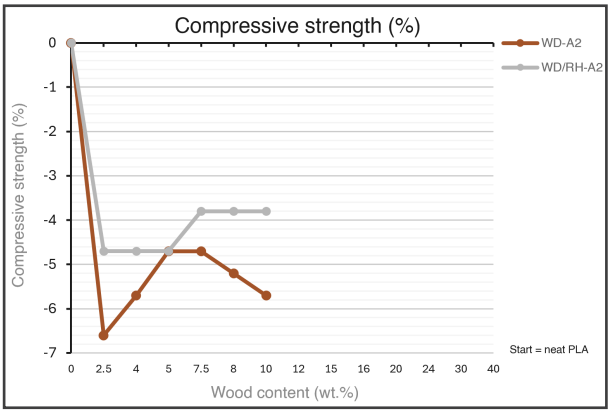


Figure A.228. Combined effect of various PLA fillers (WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, and WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}) on the compressive strength (relative % compared to neat PLA).

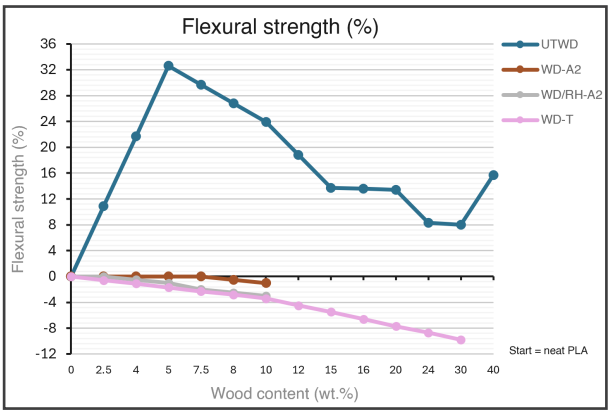


Figure A.229. Combined effect of various PLA fillers (UTWD: untreated wood dust {[79]; [77]; and [55]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]}) on the flexural strength (relative % compared to neat PLA).

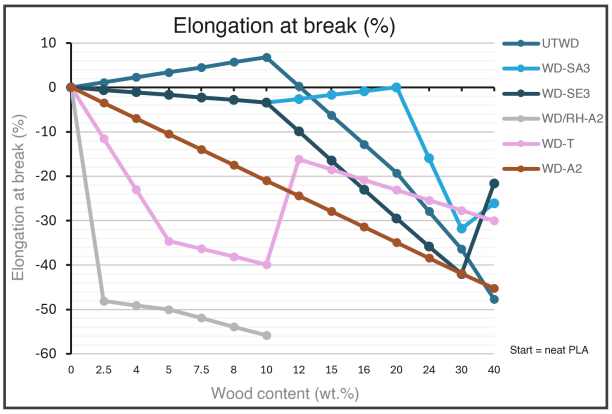


Figure A.230. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, WD-T: thermally-treated wood dust {[55]; [46]; and [8]}, and WD-A2: 2wt.% Alkaline-treated wood dust {[6]}) on the elongation at break (relative % compared to neat PLA).

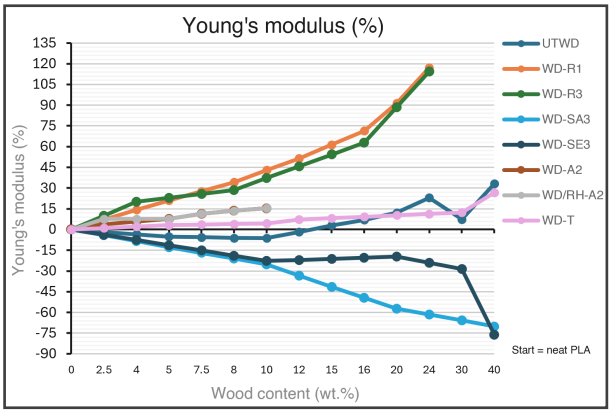


Figure A.231. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]; [51]; [29]; [79]; [77]; and [55]}, WD-R1/WD-R3: 1/3wt.% phenolic resin-treated wood dust {[29]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, WD-SE1/WD-SE2/WD-SE3: 1/2/3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]; [46]; and [8]}) on the Young's modulus (relative % compared to neat PLA).

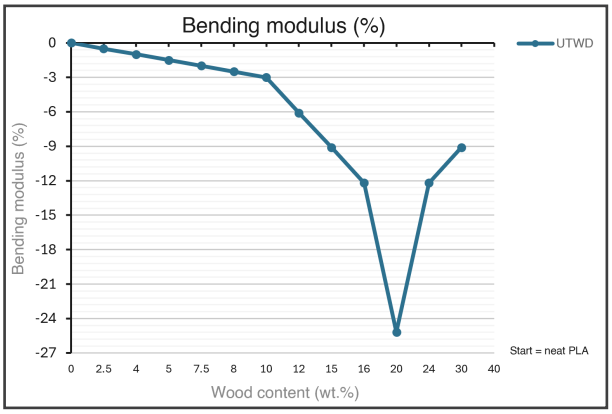


Figure A.232. Combined effect of PLA filler (UTWD: untreated wood dust {[51]}) on the bending modulus (relative % compared to neat PLA).

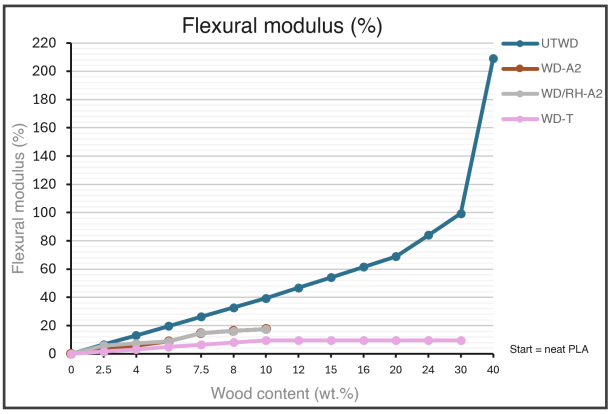


Figure A.233. Combined effect of various PLA fillers (UTWD: untreated wood dust {[77]; and [55]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]}) on the flexural modulus (relative % compared to neat PLA).

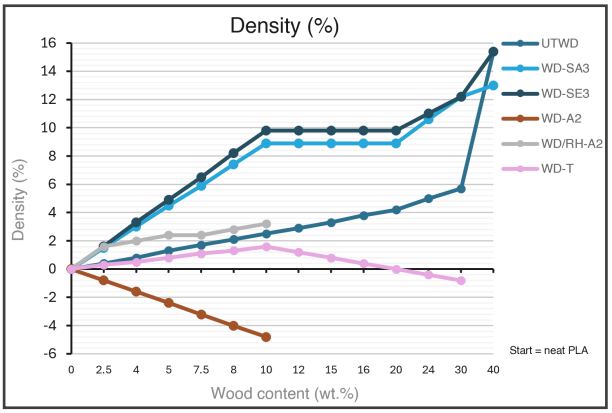


Figure A.234. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]}) on the density (relative % compared to neat PLA).

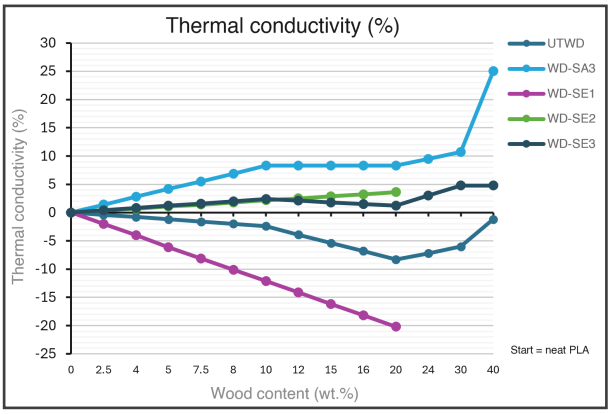


Figure A.235. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, and WD-SE1/WD-SE2/WD-SE3: 1/2/3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}) on the thermal conductivity (relative % compared to neat PLA).

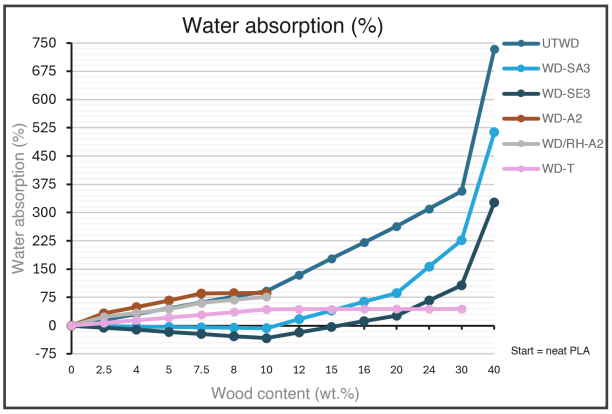


Figure A.236. Combined effect of various PLA fillers (UTWD: untreated wood dust {[1]; [51]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {[1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {[1]}, WD-A2: 2wt.% Alkaline-treated wood dust {[87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {[87]}, and WD-T: thermally-treated wood dust {[55]}) on the water absorption (relative % compared to neat PLA).

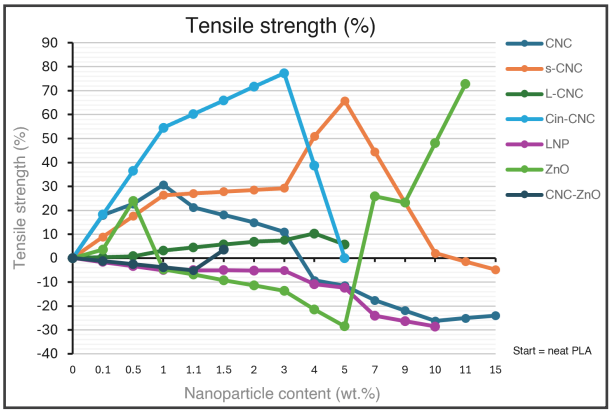


Figure A.237. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[39]; [100]; [3]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {[25]; [70]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {[99]; [20]; and [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[85]; [19]; [30]; and [24]}, ZnO: Zinc oxide {[70]; [104]; [26]; [90]; and [50]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {[70]}) on the tensile strength (relative % compared to neat PLA).

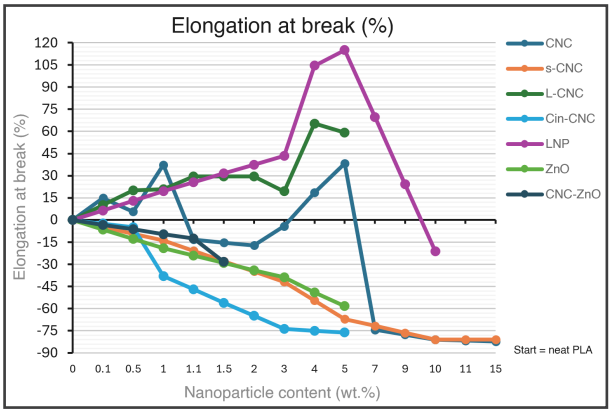


Figure A.238. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[100]; [3]; [72]; and [25]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {[25]}, L-CNC: lignin-coated cellulose nanocrystals {[99]; and [20]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[19]; [30]; and [24]}, ZnO: Zinc oxide {[104]; [26]; and [50]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {[70]}) on the elongation at break (relative % compared to neat PLA).

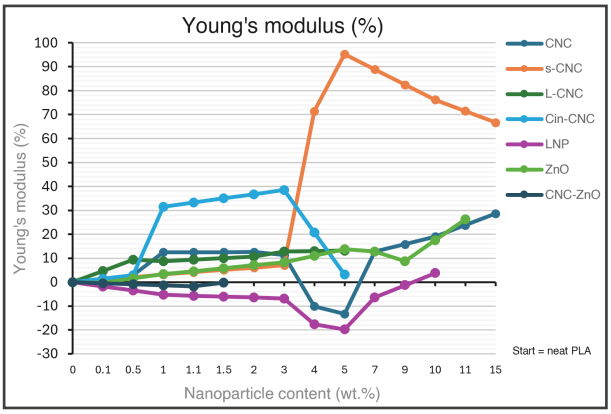


Figure A.239. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {[25]; [70]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {[99]; and [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[85]; [19]; [30]; and [24]}, ZnO: Zinc oxide {[70]; [26]; and [90]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {[70]}) on the Young's modulus (relative % compared to neat PLA).

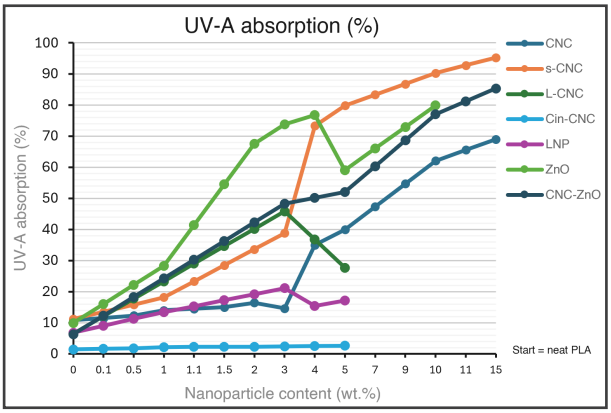


Figure A.240. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[54]; [39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {[25]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {[85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[85]; and [24]}, ZnO: Zinc oxide {[63]; and [57]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {[97]}) on the UV-A absorption (relative % compared to neat PLA).

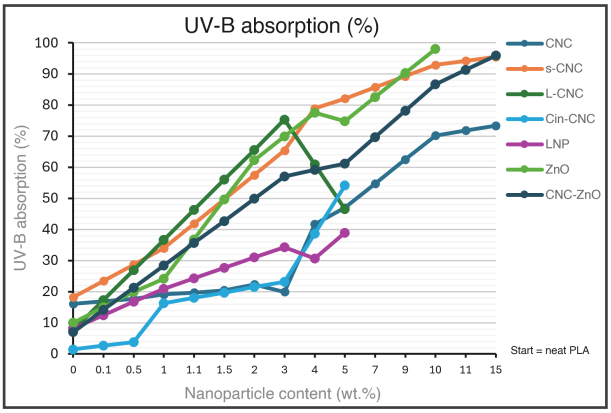


Figure A.241. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[54]; [39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {[25]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {[85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[85]; and [24]}, ZnO: Zinc oxide {[63]; and [57]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {[97]}) on the UV-B absorption (relative % compared to neat PLA).

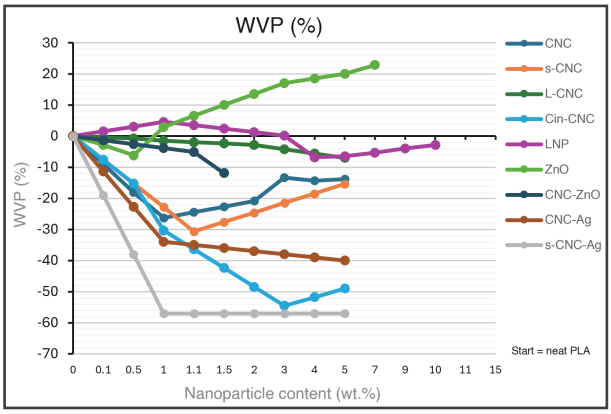


Figure A.242. Combined effect of various PLA nanoparticles (CNC: cellulose nanocrystals {[54]; [72]; [25]; [88]; [41]; and [40]}, s-CNC: surfactant-modified cellulose nanocrystals {[70]; [41]; and [40]}, L-CNC: lignin-coated cellulose nanocrystals {[99]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {[88]}, LNP: lignin nanoparticles {[103]; and [19]}, ZnO: Zinc oxide {[70]; and [57]}, CNC-ZnO: cellulose nanocrystals-Zinc oxide {[70]}, CNC-Ag: cellulose nanocrystals + 1wt.% silver nanopowder {[41]}, and s-CNC-Ag: surfactant-modified cellulose nanocrystals + 1wt.% silver nanopowder {[40]}) on the water vapour permeability (relative % compared to neat PLA).

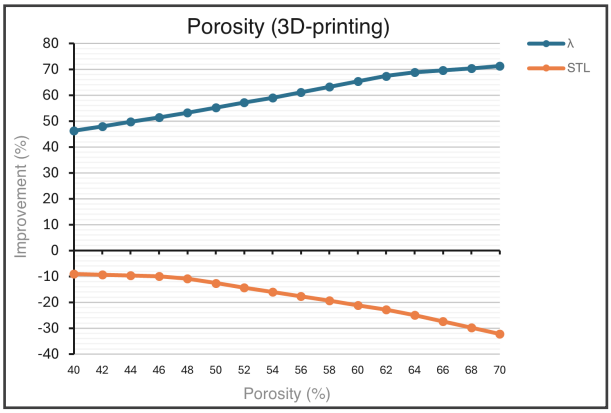


Figure A.243. Effect of PLA porosity on the thermal conductivity, and sound transmission loss (relative % compared to neat PLA). Adapted from "*Thermal and acoustic performance evalation of 3D-Printable PLA materials*", by Islam et al., 2023 [49].

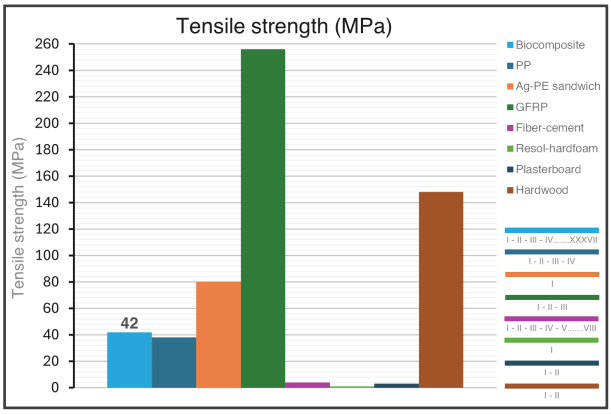


Figure A.244. Tensile strength overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, resol-hardfoam {Adapted from Edupack, 2023}, plasterboard {Adapted from [49]; and Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

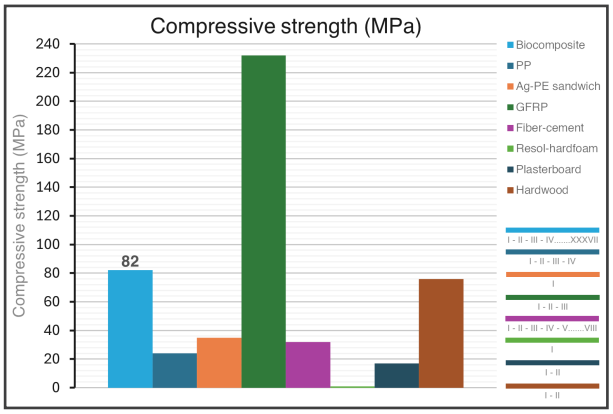


Figure A.245. Compressive strength overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, resol-hardfoam {Adapted from Edupack, 2023}, plasterboard {Adapted from [49]; and Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

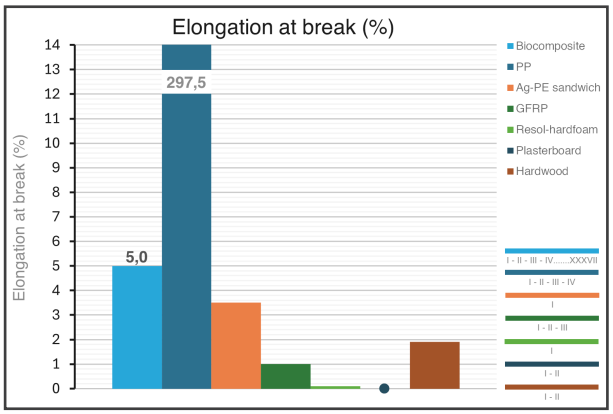


Figure A.246. Elongation at break overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, resol-hardfoam {Adapted from Edupack, 2023}, plasterboard {Adapted from [49]; and Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

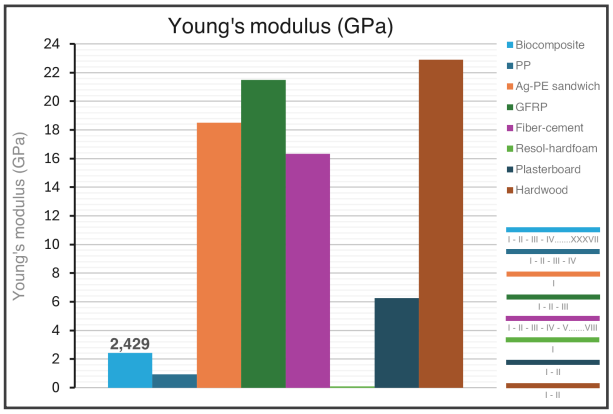


Figure A.247. Young's modulus overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, resol-hardfoam {Adapted from Edupack, 2023}, plasterboard {Adapted from [49]; and Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

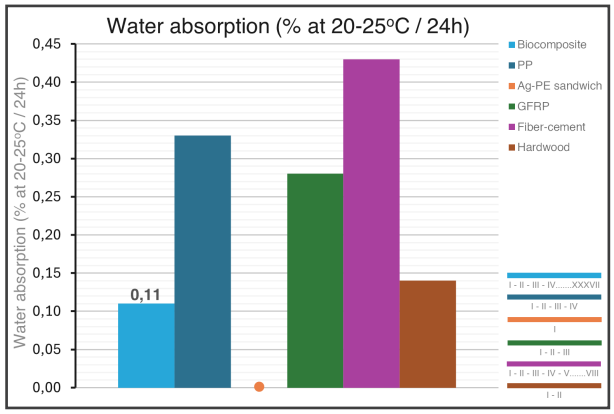


Figure A.248. Water absorption overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, and hardwood {Adapted from [48]; and Edupack, 2023}).

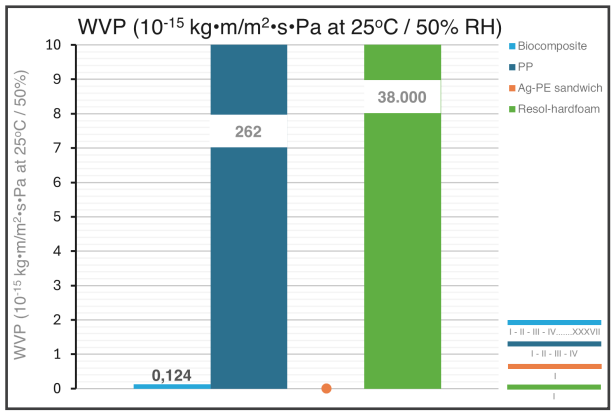


Figure A.249. Water vapour permeability overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, and resol-hardfoam {Adapted from Edupack, 2023}).

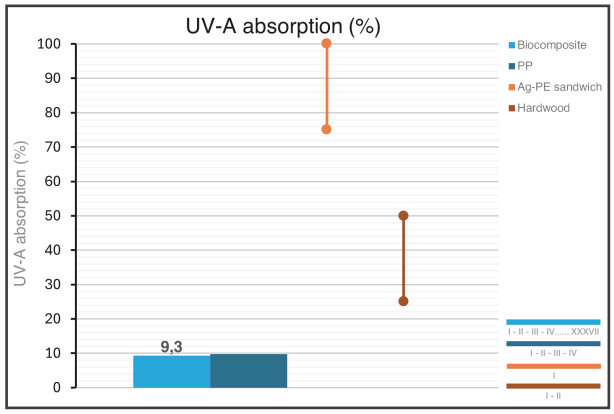


Figure A.250. UV-A absorption overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

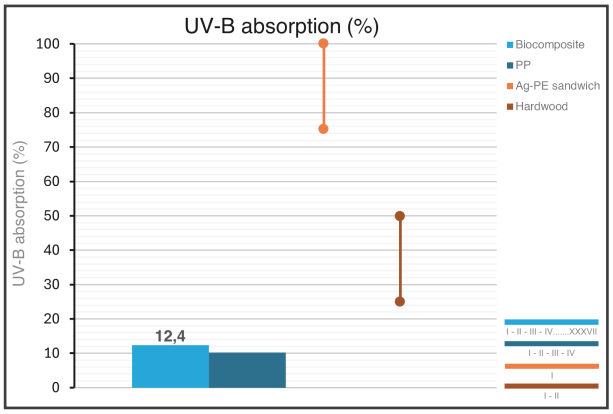


Figure A.251. UV-B absorption overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

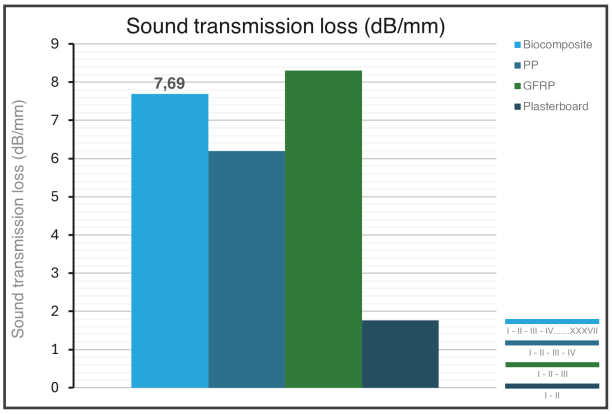


Figure A.254. Sound transmission loss overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, and plasterboard {Adapted from [49]; and Edupack, 2023}).

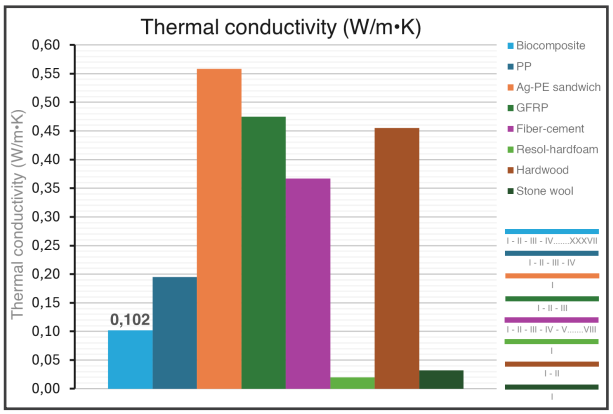


Figure A.252. Thermal conductivity overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, resol-hardfoam {Adapted from Edupack, 2023}, hardwood {Adapted from [48]; and Edupack, 2023}; and stone wool {Adapted from Edupack, 2023}).

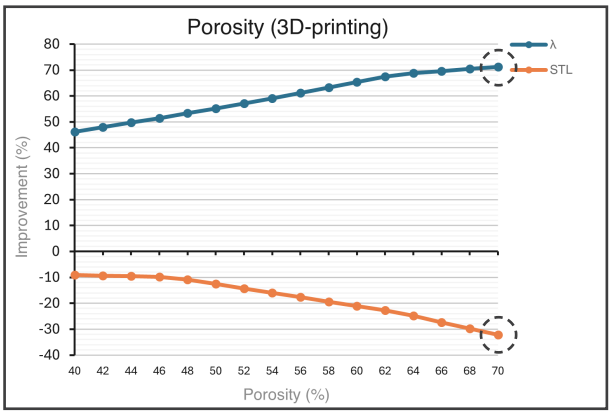


Figure A.255. Effect of PLA porosity on the thermal conductivity, and sound transmission loss (Adapted from Islam et al., 2023 [49]).

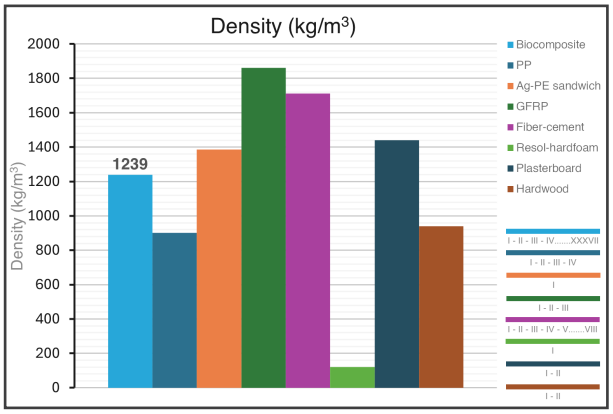


Figure A.253. Density overview biocomposite vs. building materials (biocomposite {Adapted from [1]; [3]; [6]; [8]; [14]; [19]; [20]; [24]; [25]; [26]; [29]; [30]; [35]; [39]; [40]; [41]; [46]; [50]; [51]; [54]; [55]; [57]; [61]; [63]; [70]; [72]; [77]; [79]; [85]; [87]; [88]; [90]; [97]; [99]; [100]; [103]; and [104]}, PP {Adapted from [2]; [65]; [68]; and Edupack, 2023}, Ag-PE sandwich {Adapted from Edupack, 2023}, GFRP {Adapted from [12]; [86]; and Edupack, 2023}, fiber-cement {Adapted from [47]; [59]; [62]; [64]; [84]; [92]; [98]; and [105]}, resol-hardfoam {Adapted from Edupack, 2023}, plasterboard {Adapted from [49]; and Edupack, 2023}, and hardwood {Adapted from [48]; and Edupack, 2023}).

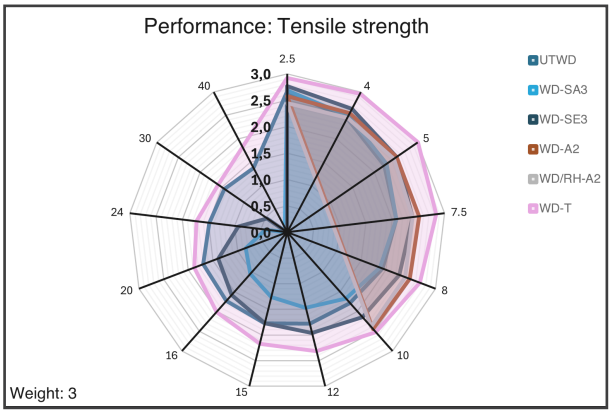


Figure A.256. Performance score of various PLA fillers (UTWD: untreated wood dust {Adapted from [1]; [51]; [29]; [79]; [77]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from [1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from [87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from [87]}, and WD-T: thermally-treated wood dust {Adapted from [8]; [46]; and [55]}) on the tensile strength.

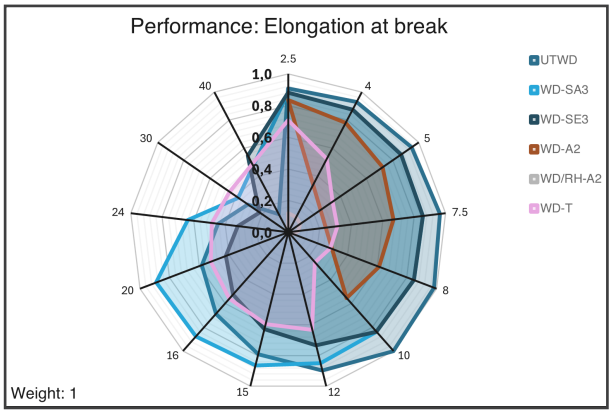


Figure A.257. Performance score of various PLA fillers (UTWD: untreated wood dust {Adapted from [1]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from [1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from [6]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from [87]}, and WD-T: thermally-treated wood dust {Adapted from [8]; [46]; and [55]}) on the elongation at break.

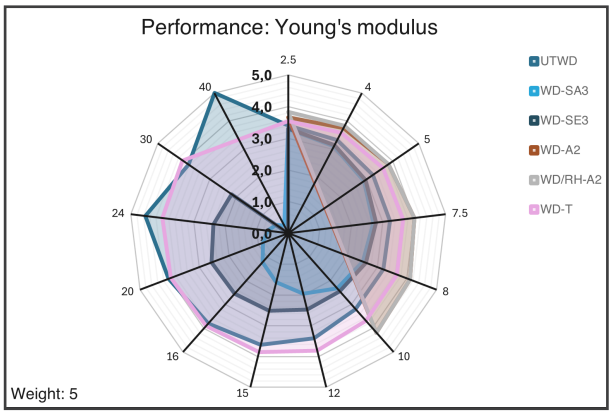


Figure A.258. Performance score of various PLA fillers (UTWD: untreated wood dust {Adapted from [1]; [51]; [29]; [79]; [77]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from [1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from [87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from [87]}, and WD-T: thermally-treated wood dust {Adapted from [8]; [46]; and [55]}) on the Young's modulus.

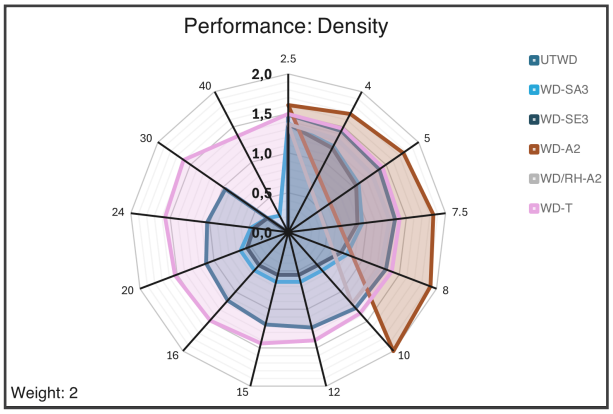


Figure A.259. Performance score of various PLA fillers (UTWD: untreated wood dust {Adapted from [1]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from [1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from [87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from [87]}, and WD-T: thermally-treated wood dust {Adapted from [55]}) on the density.

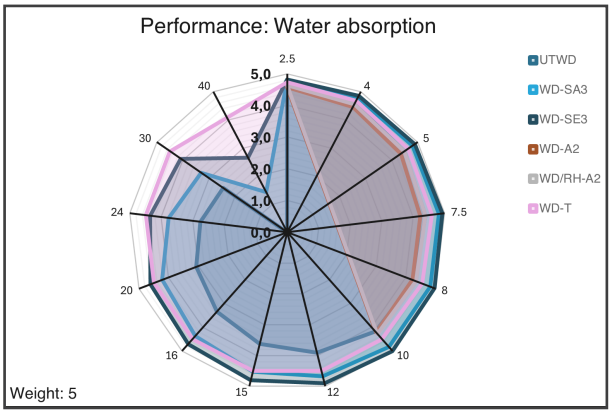


Figure A.260. Performance score of various PLA fillers (UTWD: untreated wood dust {Adapted from [1]; [51]; and [55]}, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio {Adapted from [1]}, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio {Adapted from [1]}, WD-A2: 2wt.% Alkaline-treated wood dust {Adapted from [87]}, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio {Adapted from [87]}, and WD-T: thermally-treated wood dust {Adapted from [55]}) on the water absorption.

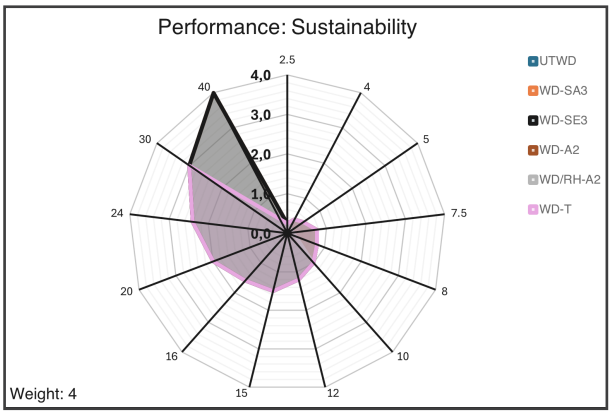


Figure A.261. Performance score of various PLA fillers (UTWD: untreated wood dust, WD-SA3: 3wt.% SA-treated date palm wood fibers in 50:50vo.% acetone:water solvent ratio, WD-SE3: 3wt.% SE-treated date palm wood fibers in 90:10vo.% ethanol:water solvent ratio, WD-A2: 2wt.% Alkaline-treated wood dust, WD/RH-A2: 2wt.% Alkaline-treated wood dust/rice husk in 50:50wt.% ratio, and WD-T: thermally-treated wood dust) on the sustainability (% fiber content).

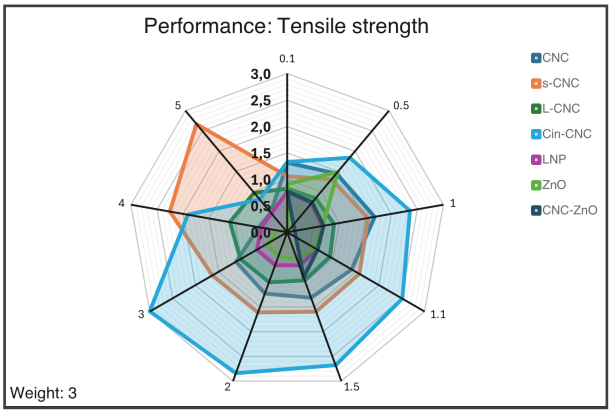


Figure A.262. Step 20: Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [39]; [100]; [3]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [25]; [70]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [99]; [20]; and [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [85]; [19]; [30]; and [24]}, ZnO: Zinc oxide {Adapted from [70]; [104]; [26]; [90]; and [50]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [70]}) on the tensile strength.

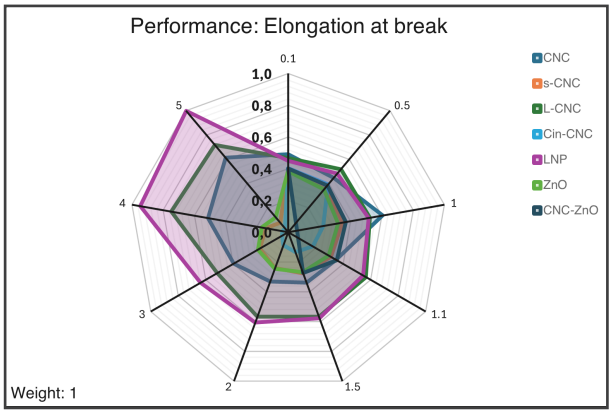


Figure A.263. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [100]; [3]; [72]; and [25]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [25]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [99]; and [20]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [19]; [30]; and [24]}, ZnO: Zinc oxide {Adapted from [104]; [26]; and [50]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [70]}) on the elongation at break.

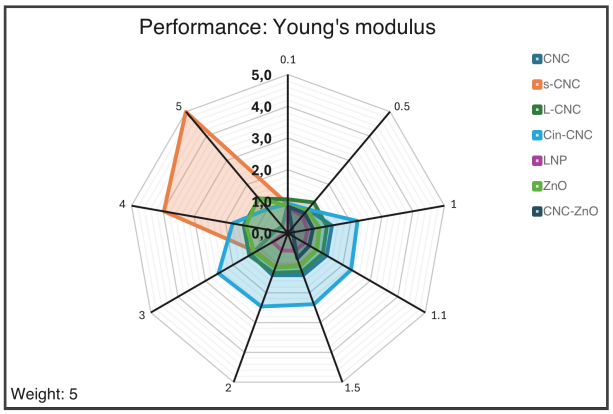


Figure A.264. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [25]; [70]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [99]; and [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [85]; [19]; [30]; and [24]}, ZnO: Zinc oxide {Adapted from [70]; [26]; and [90]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [70]}) on the Young's modulus.

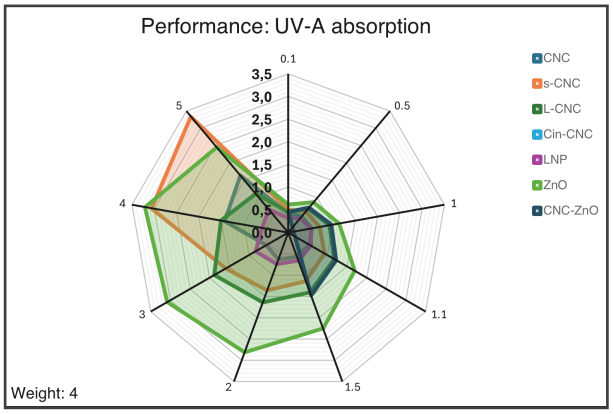


Figure A.265. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [54]; [39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [25]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [85]; and [24]}, ZnO: Zinc oxide {Adapted from [63]; and [57]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [97]}) on the UV-A absorption.

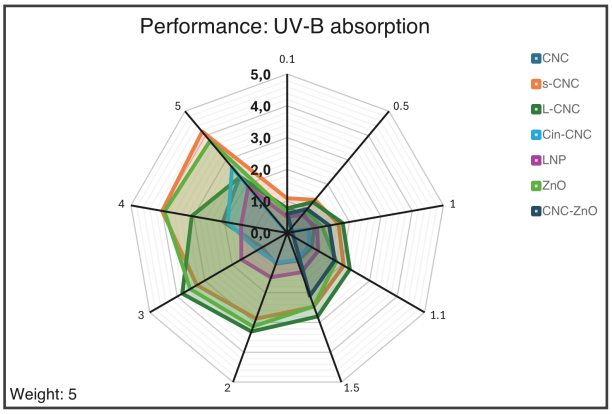


Figure A.266. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [54]; [39]; [72]; [25]; and [88]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [25]; and [39]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [85]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [85]; and [24]}, ZnO: Zinc oxide {Adapted from [63]; and [57]}, and CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [97]}) on the UV-B absorption.

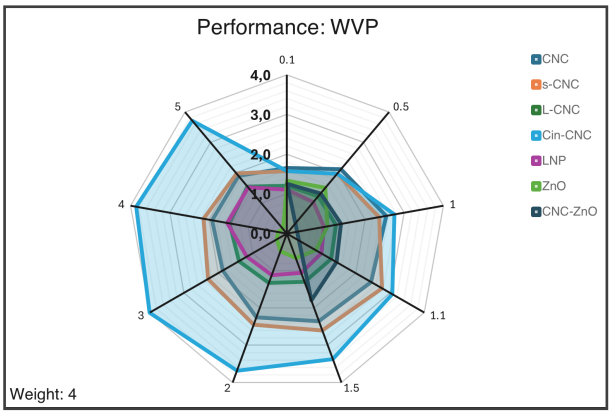


Figure A.267. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals {Adapted from [54]; [72]; [25]; [88]; [41]; and [40]}, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio {Adapted from [70]; [41]; and [40]}, L-CNC: lignin-coated cellulose nanocrystals {Adapted from [99]}, Cin-CNC: cinnamate-grafted cellulose nanocrystals {Adapted from [88]}, LNP: lignin nanoparticles {Adapted from [103]; and [19]}, ZnO: Zinc oxide {Adapted from [70]; and [57]}, CNC-ZnO: cellulose nanocrystals-Zinc oxide {Adapted from [70]} on the water vapour permeability.

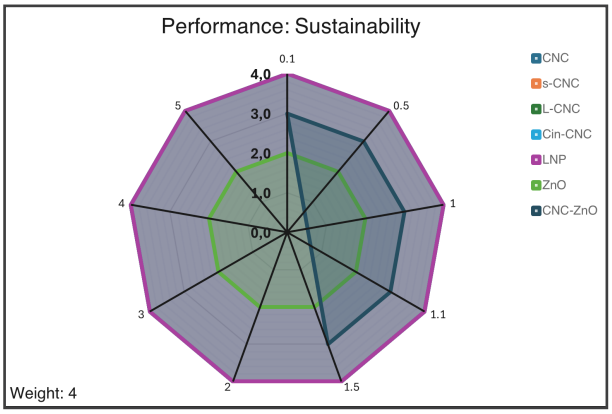


Figure A.268. Performance score of various PLA nanoparticles (CNC: cellulose nanocrystals, s-CNC: surfactant-modified cellulose nanocrystals with phosphate ester of ethoxylated nonylphenol in 1:4vo.% ratio, L-CNC: lignin-coated cellulose nanocrystals, Cin-CNC: cinnamate-grafted cellulose nanocrystals, LNP: lignin nanoparticles, ZnO: Zinc oxide, CNC-ZnO: cellulose nanocrystals-Zinc oxide) on the sustainability (organic/inorganic).

B

Facade design

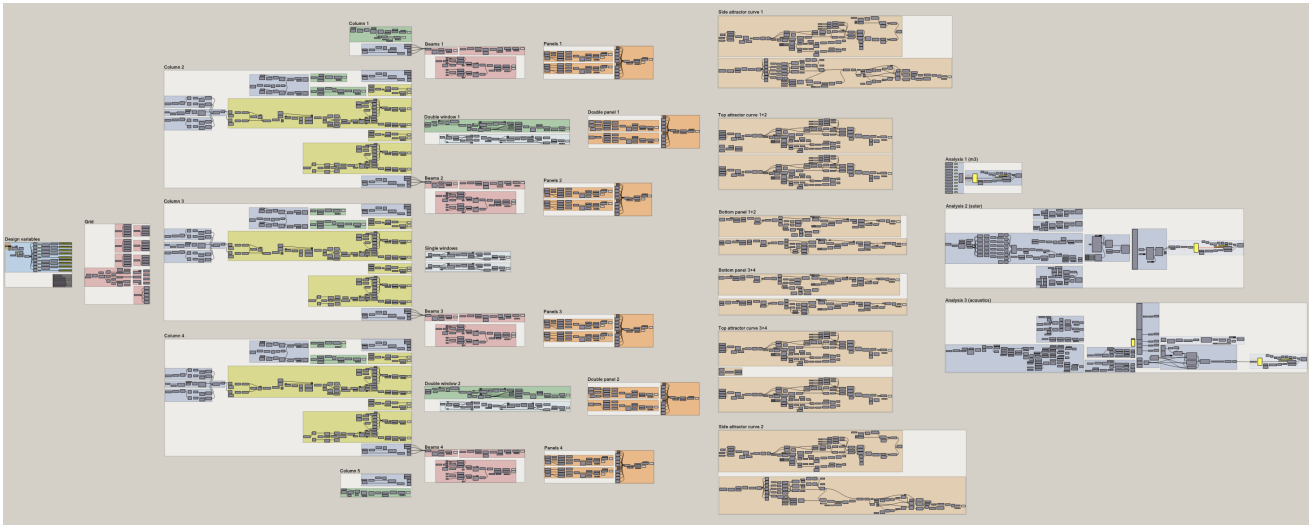


Figure B.1. Overview of the entire parametric facade design script with the creation of design variables, grid, columns, beams, windows, facade panels, and simulations.

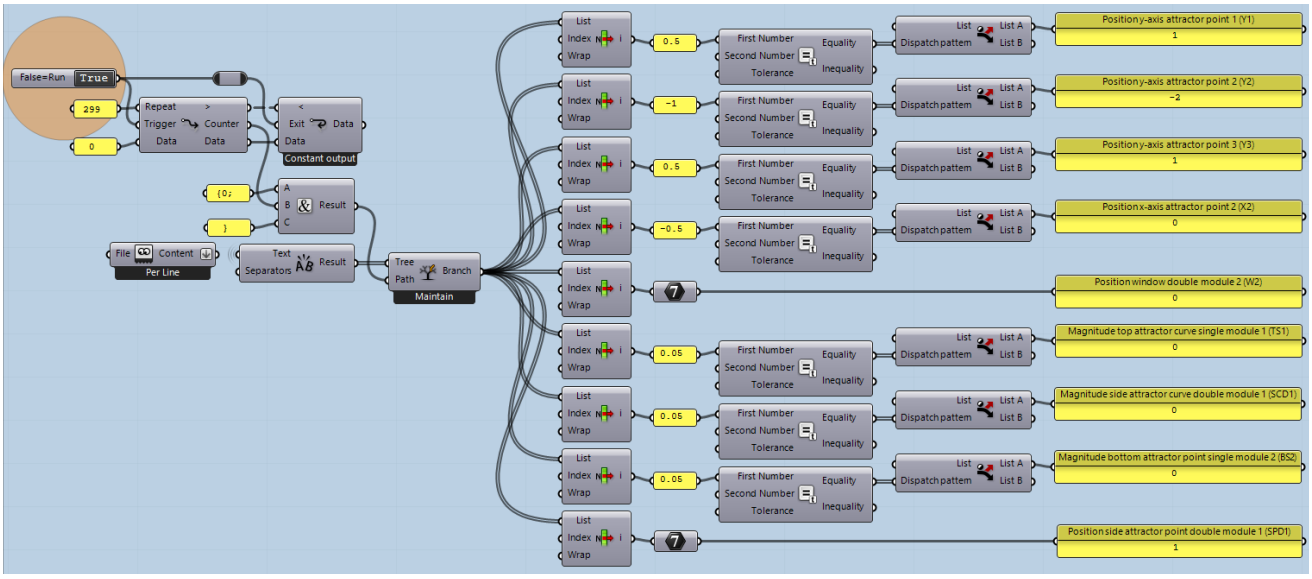
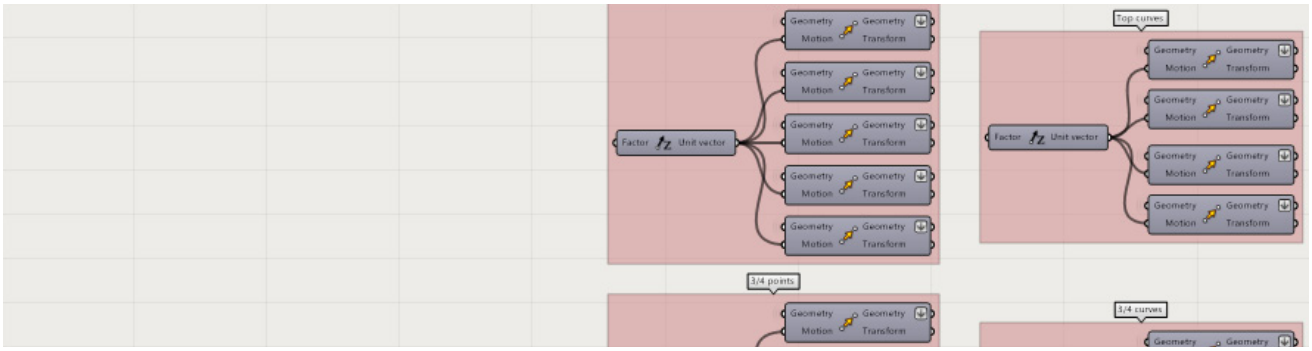


Figure B.2. Design variables script as input for the parametric model with automatic loops (using Ademone) through design vectors extracted during stratified sampling.



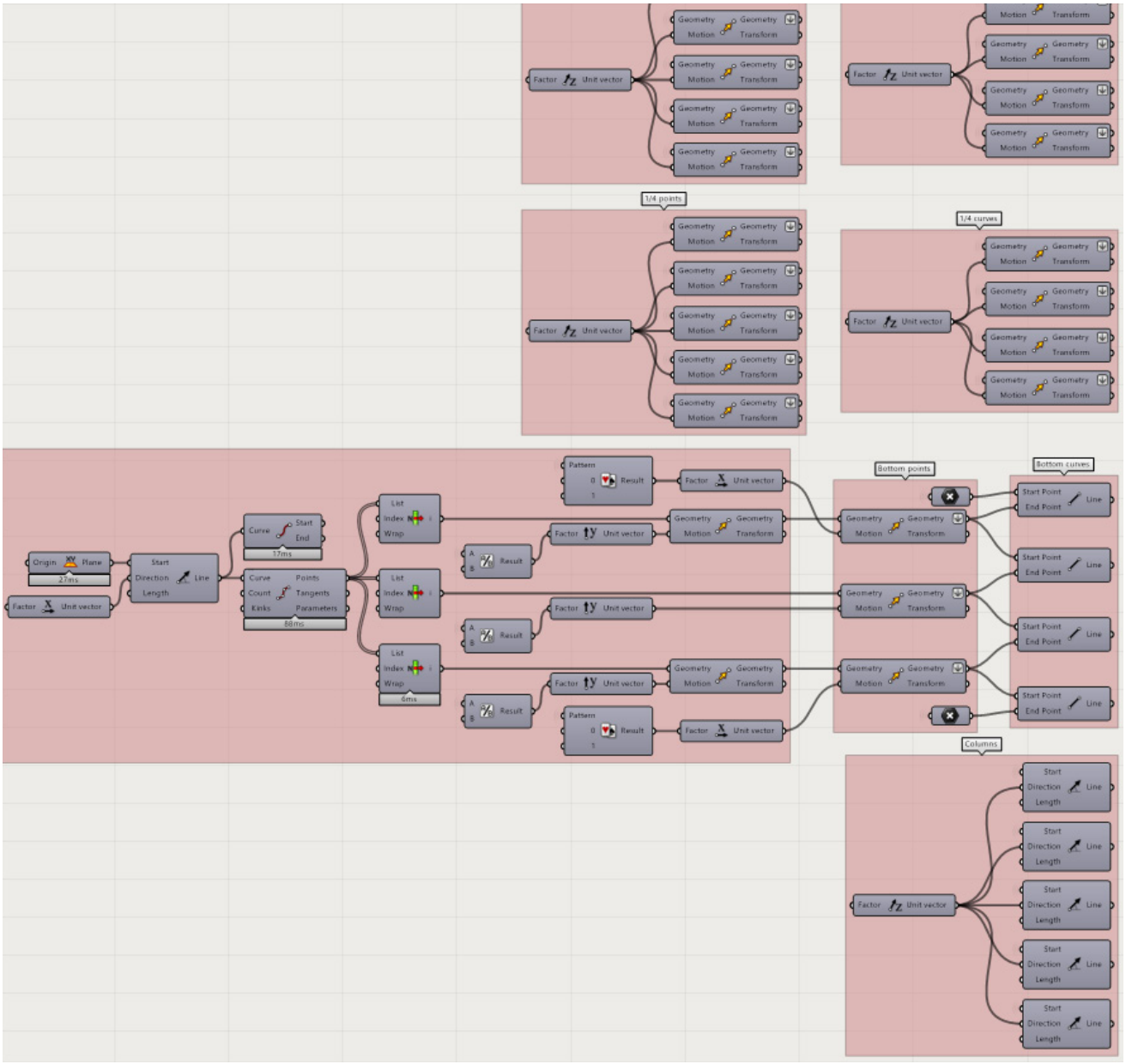


Figure B.3. The foundation of the parametric facade design: the creation of a vertical and horizontal grid of points, which is then used create the columns and beams.

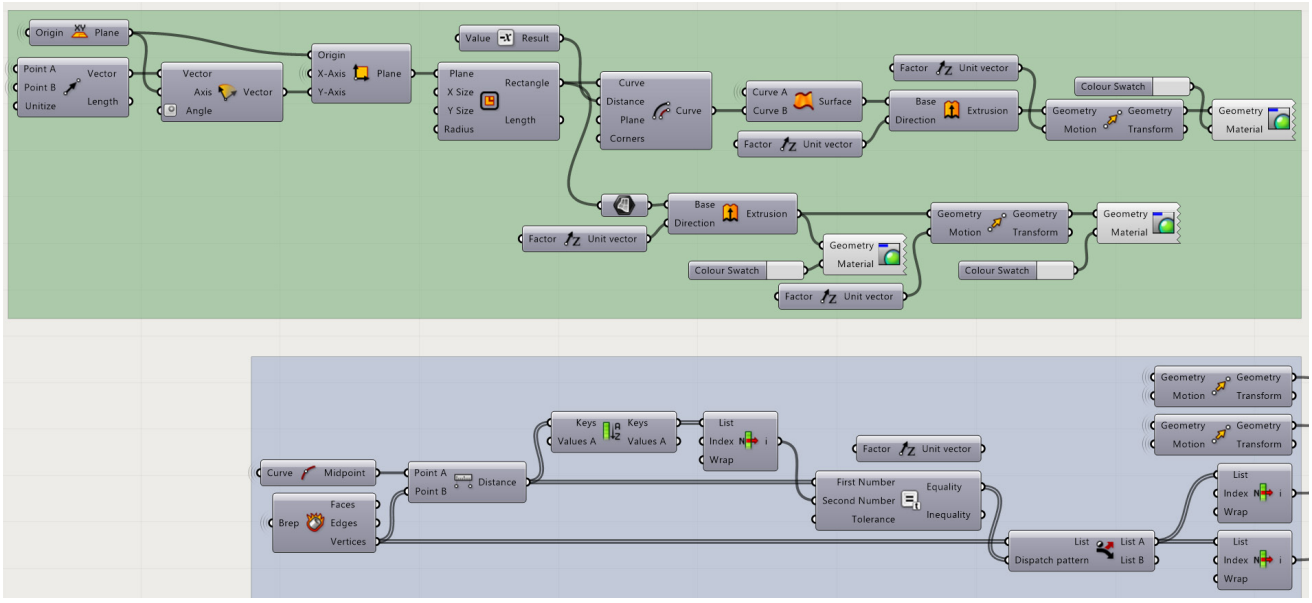


Figure B.4. The creation of the first column (from left to right) using the rectangle, offset curve, ruled surface, and extrude component to create a rectangular hollow section.

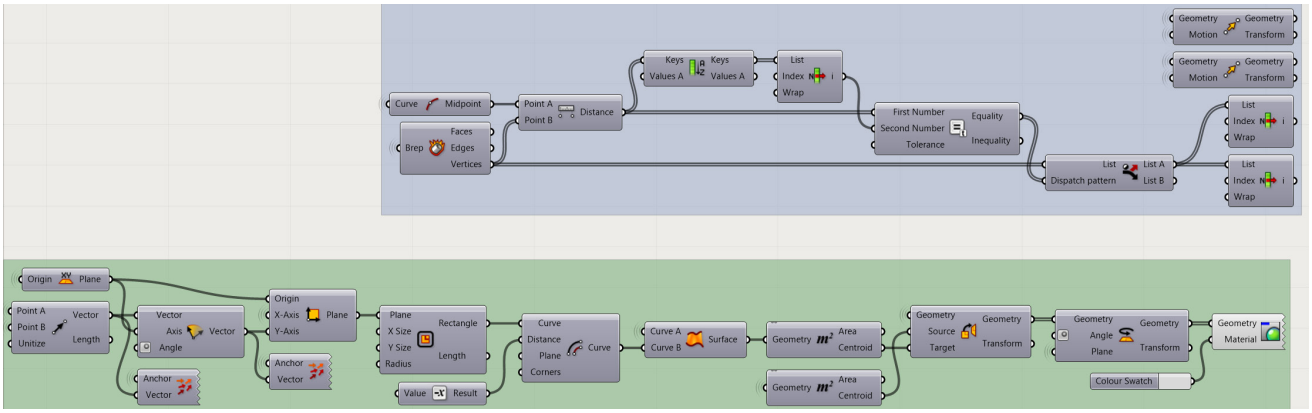


Figure B.5. The creation of the fifth column using the same method as that of the first column, though with additional rotations to ensure proper alignment with the facade.

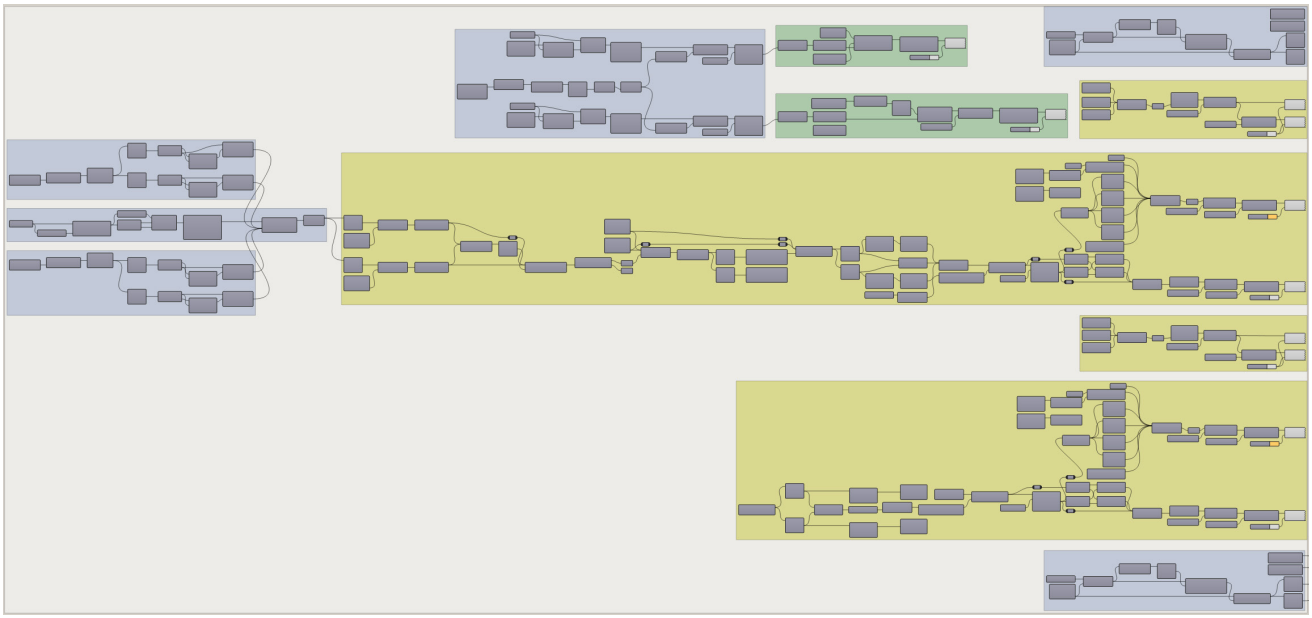


Figure B.6. Overview of the script with the creation of two of the middle columns, and the creation of insulated aluminium profiles that connect the vertical facade sections.

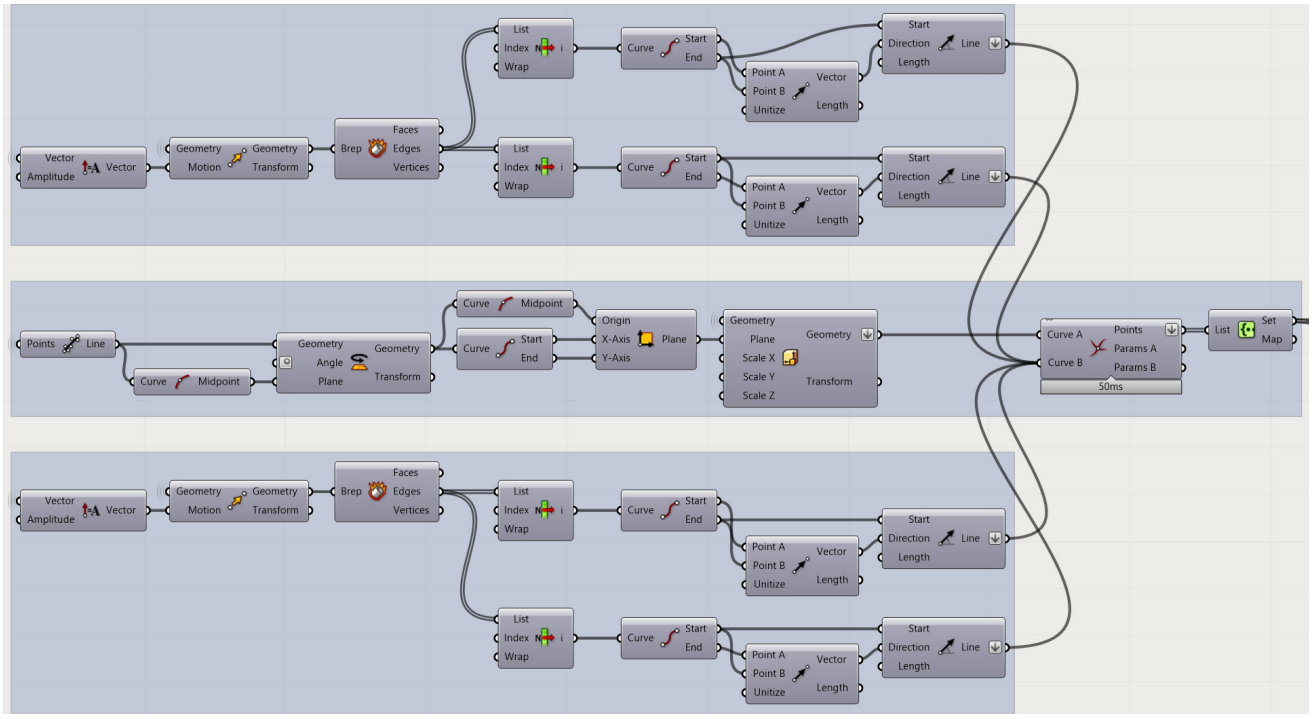


Figure B.7. Setting up two points at each intersection of two facade sections to ensure that the RHS-columns don't overlap each other when the facade changes shape.

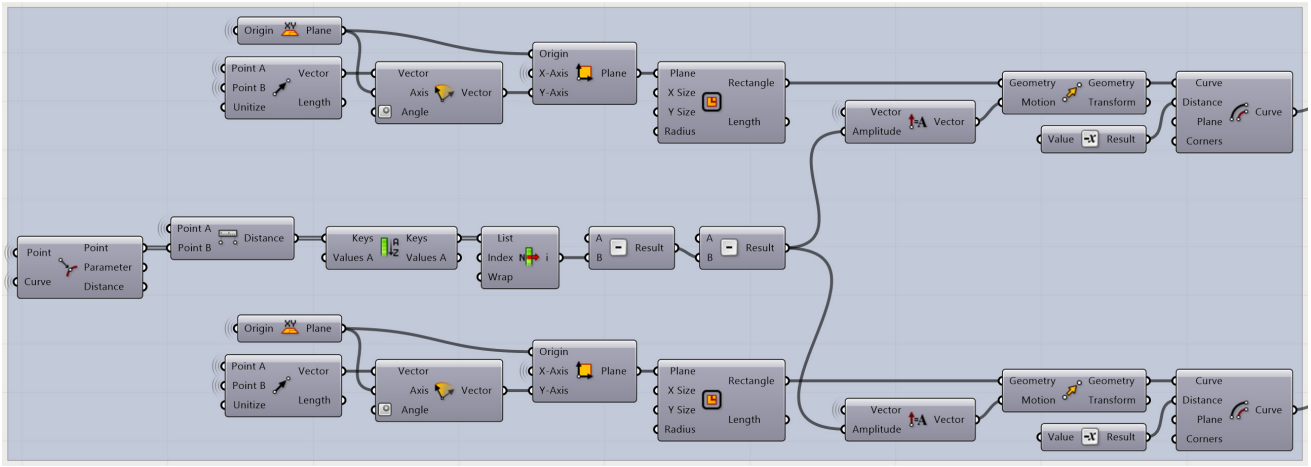


Figure B.8. The creation of two of the middle columns using same method as that of the first and fifth column, using rectangle, offset curve, ruled surface, and extrude.

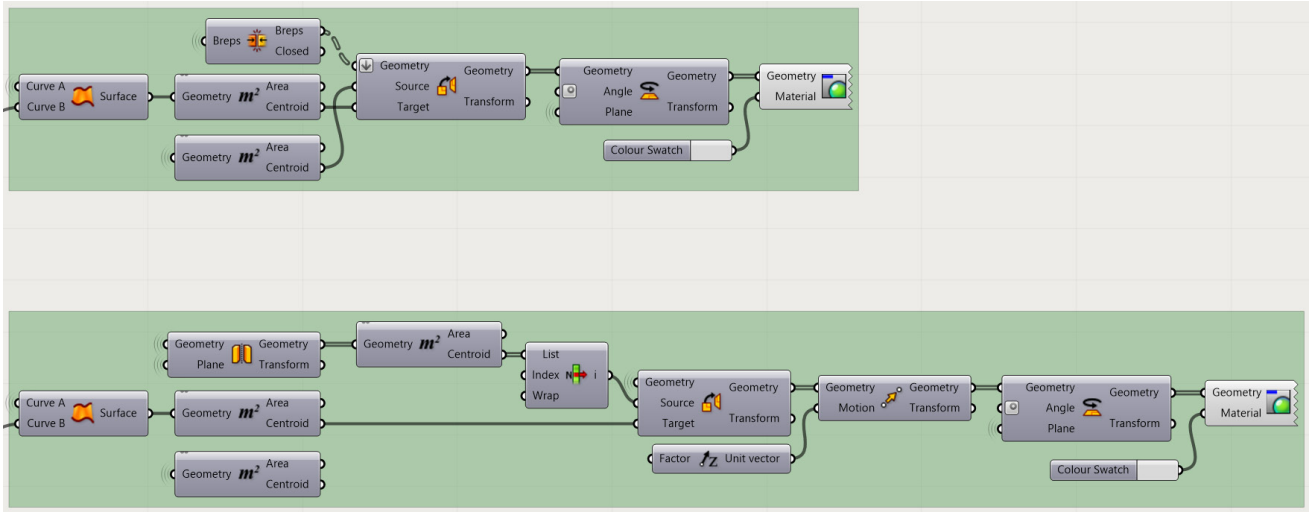


Figure B.9. Extruding one of the middle columns using ruled surface and extrude, and then using orient and rotate to move it to the coordinate for the adjacent column.

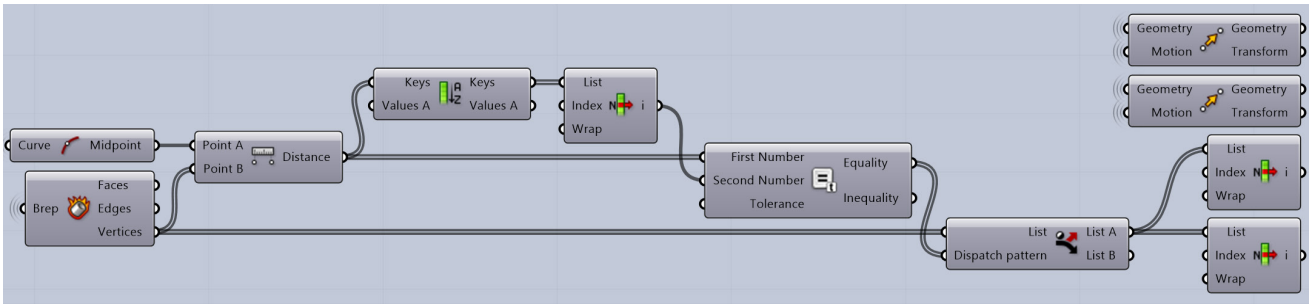


Figure B.10. Setting up points for beams using deconstruct brep on column rectangle, using dispatch pattern based on distance to ensure proper selection and move unit-z.

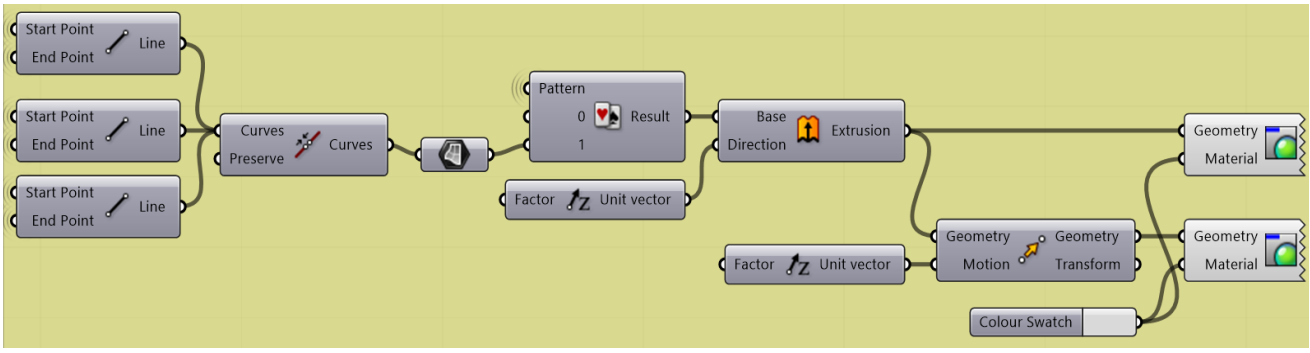


Figure B.11. The creation of the bottom and top aluminium caps for closing off the rectangular hollow columns using curves, surface, extrusion, and move with unit-z.

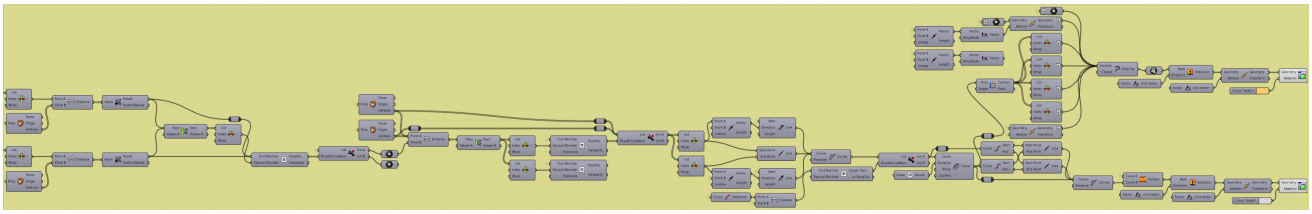


Figure B.12. The creation of the u-shaped aluminum profiles that connect the vertical facade sections - based on an angle threshold - and the placement of insulation in it.

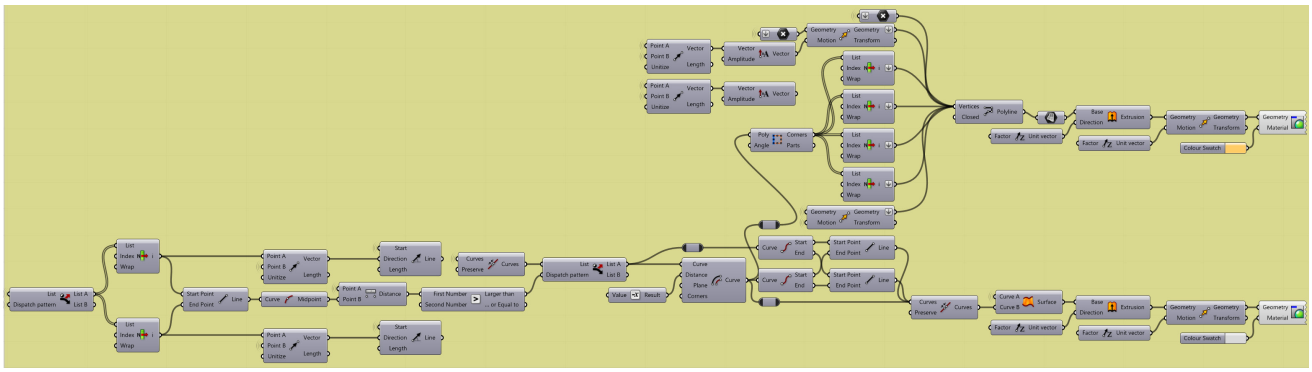


Figure B.13. The creation of the same u-shaped aluminium profiles that connect the vertical facade sections and the insulation in it, but on the side with the smallest angle.

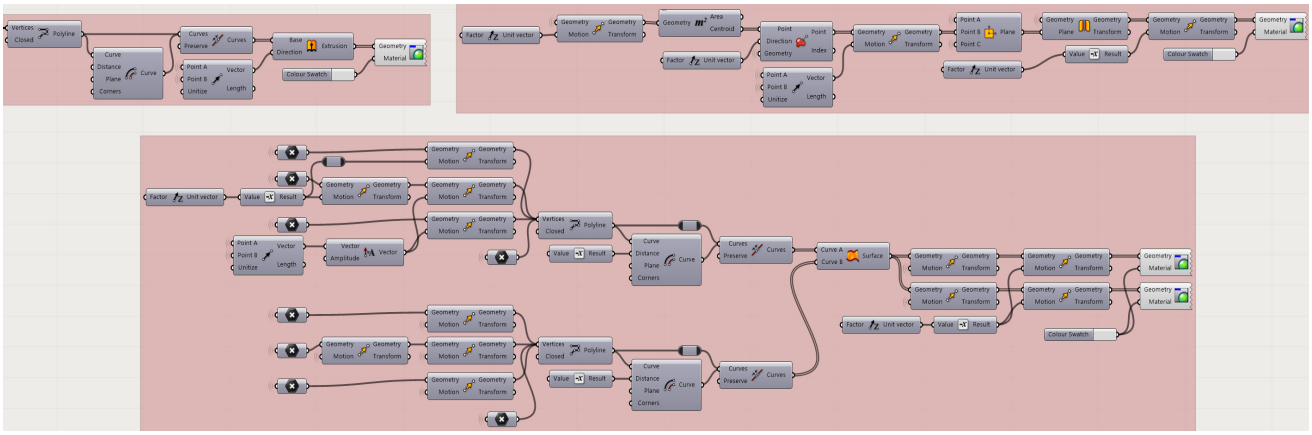


Figure B.14. The creation of the RHS-beams using the same method as employed for the first and fifth column, with copying (move) one of them to different heights.

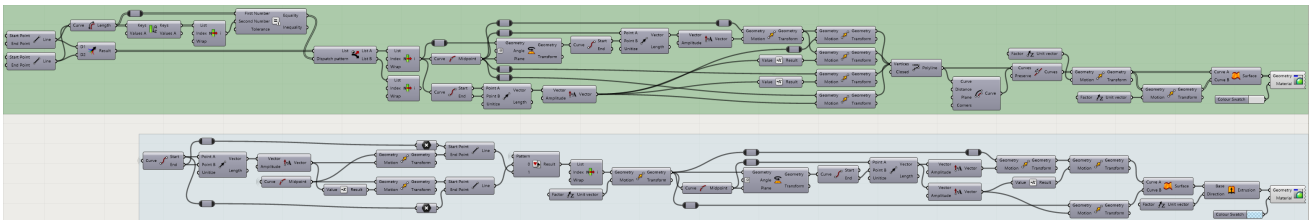


Figure B.15. The creation of the windows of the double modules, by creating a column as a vertical divider and creating the window on the right side as the initial position.

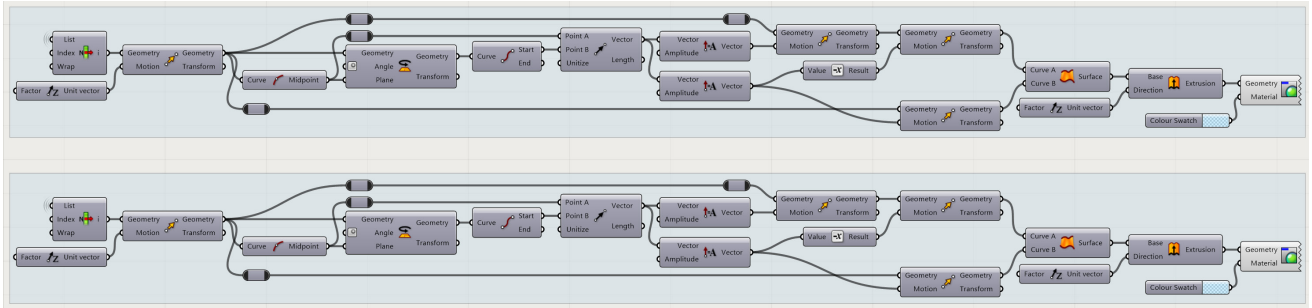


Figure B.16. The creation of the windows of the single modules, using the midpoints of the horizontal beams at 0.75 meter height, followed by extruding the window.

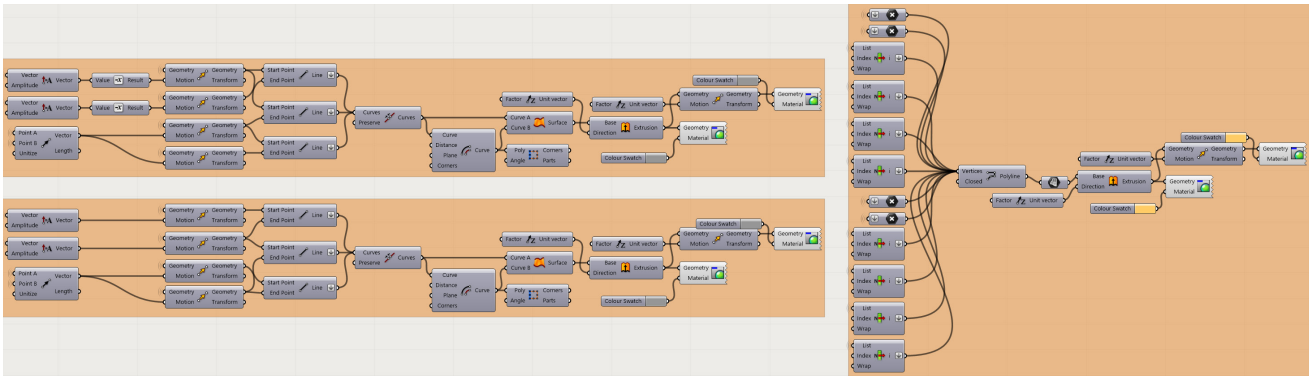


Figure B.17. The creation of insulating aluminium sandwich panels for the single modules, using the same method as the connections between vertical facade sections.

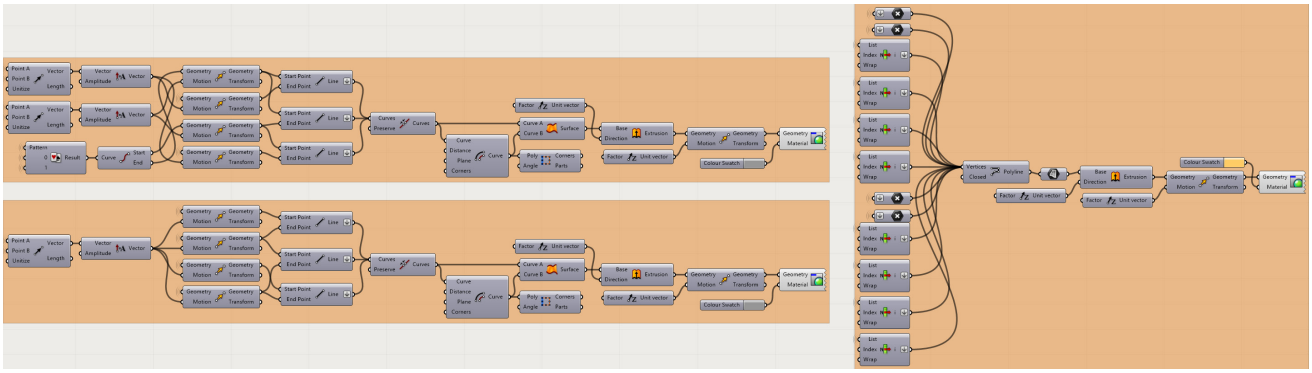


Figure B.18. The creation of insulating aluminium sandwich panels for the double modules, using the same method as the creation of the panels for the single modules.

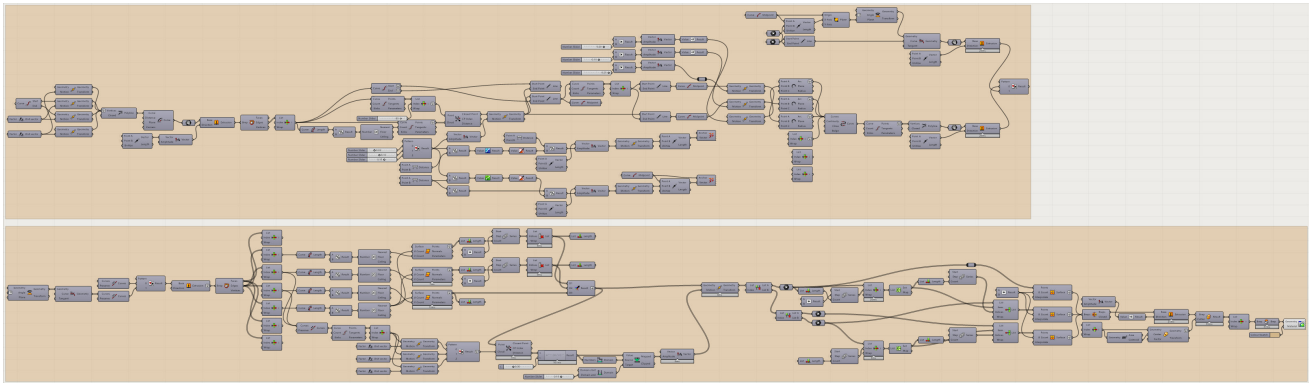


Figure B.19. The creation of the vertical biocomposite facade panels, and the creation of attraction points that can change their shape by moving material out vertically.

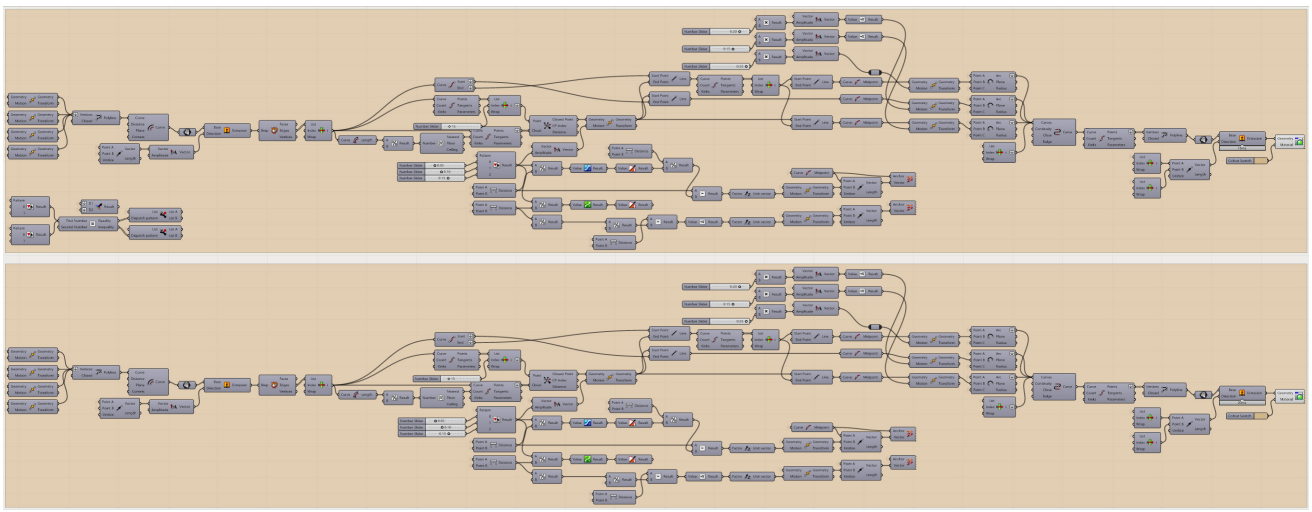


Figure B.20. The creation of the top biocomposite facade panels, and the creation of attraction points that can change their shape by moving material out horizontally.

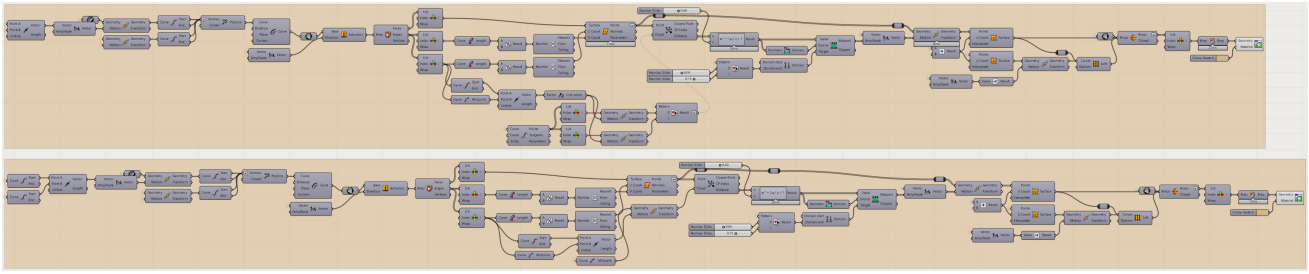


Figure B.21. The creation of the bottom biocomposite facade panels, and the creation of an attractor point at its midpoint that moves material out in a bell-shaped way.

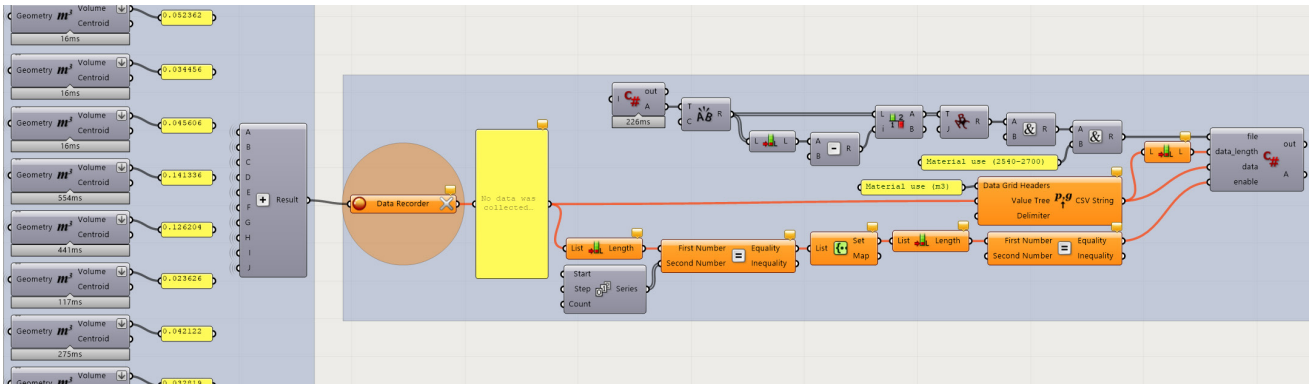


Figure B.22. The total material use performance calculation using the volume and addition component, a data recorder, and an automatic csv file saver using a C# script.

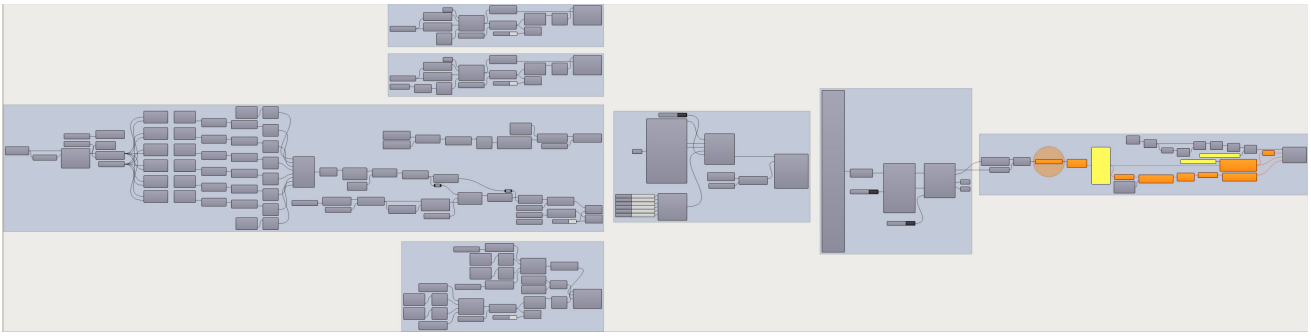


Figure B.23. Overview of the solar heat gains performance simulation, with experimental setup, solar settings, incident radiation analysis, data recorder, and csv file saver.

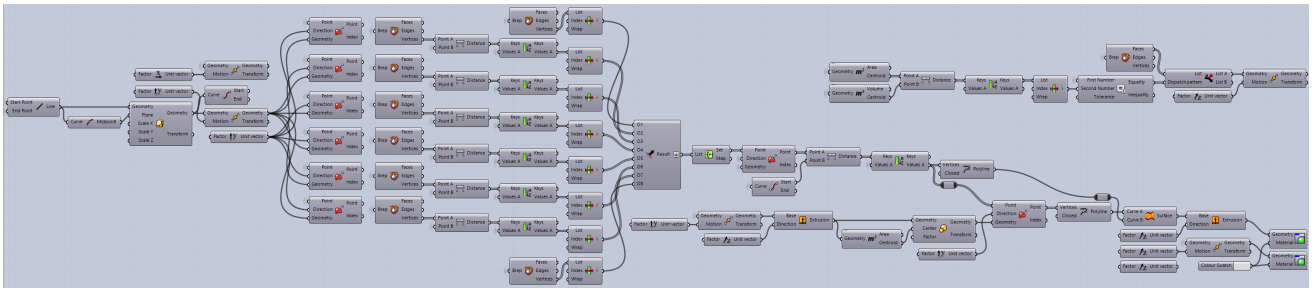


Figure B.24. The creation of the floor and roof, as part of the experimental studio setup, using the endpoints of the facade columns to create an outline which is then extruded.

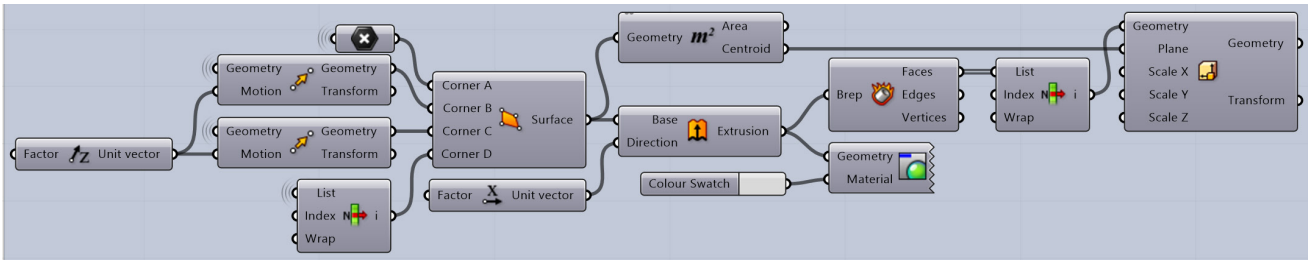


Figure B.25. The creation of the two side walls, as part of the experimental studio setup, using the top and bottom endpoints of the end columns which are then extruded.

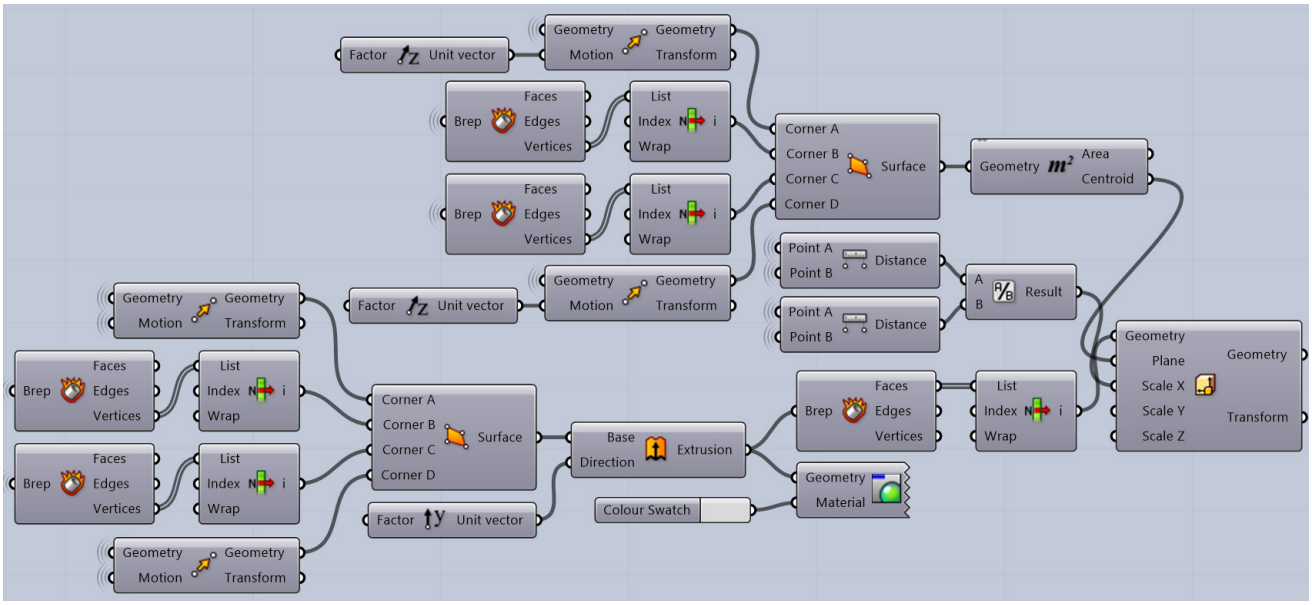


Figure B.26. The creation of the back wall, as part of the experimental studio setup, extracting the end points of both side walls, followed by using 4pt surface and extrude.

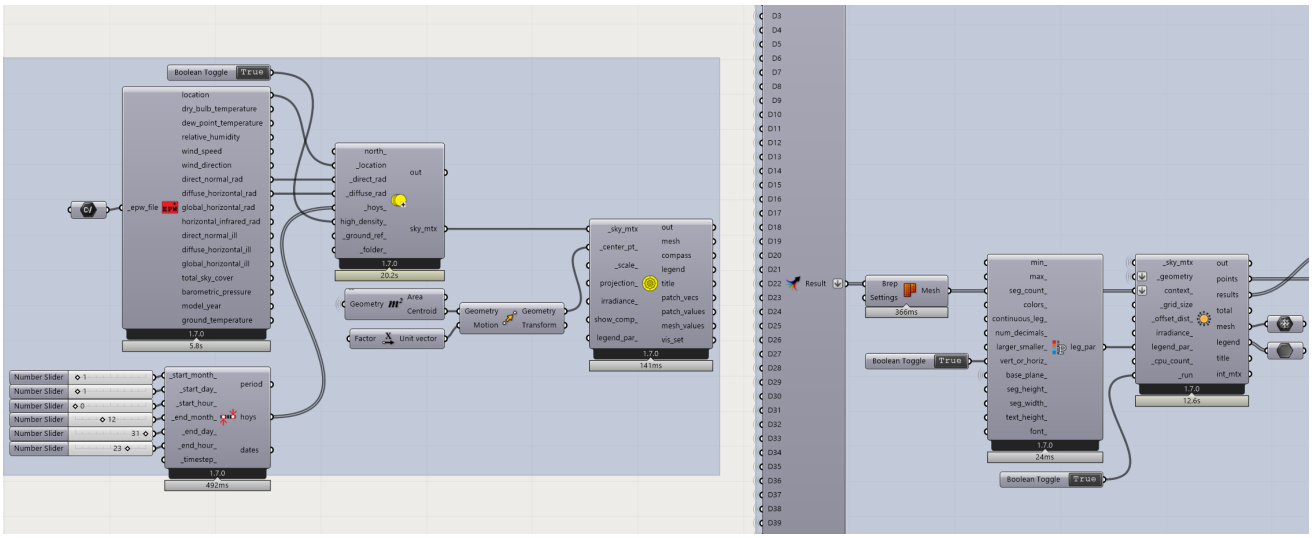


Figure B.27. Setting up the incident radiation analysis (analysing both direct and diffuse radiation) using Ladybug, based on Dutch weather conditions over a full year.

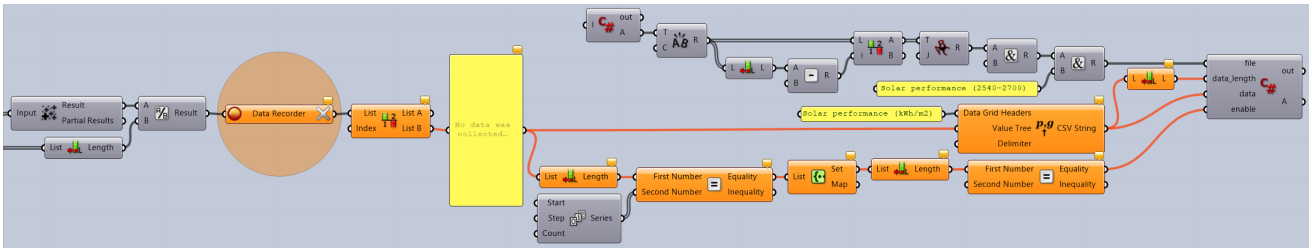


Figure B.28. Calculating the solar radiation for each of the five surfaces inside the studio, followed by dividing the total radiation by the total surface area, and csv file saver.

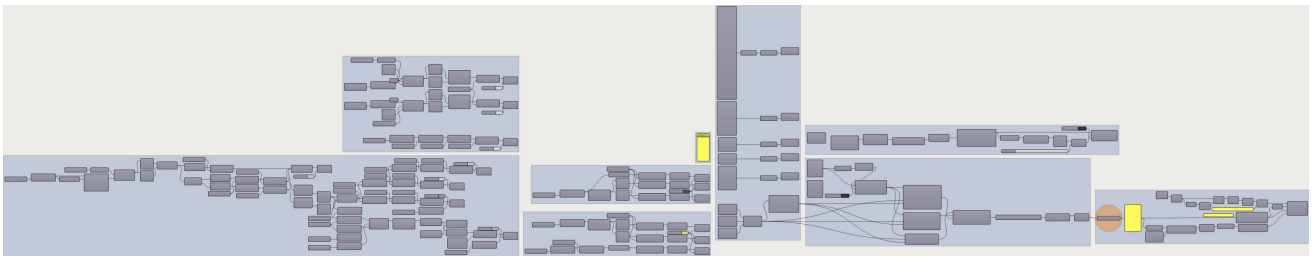


Figure B.29. Overview of the sound pressure level performance simulation, with experimental setup, material properties, acoustic simulation, data recorder, and csv saver.

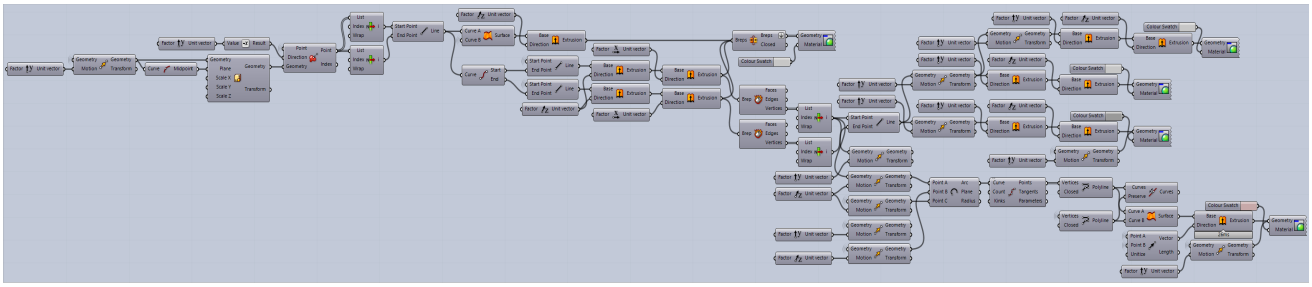


Figure B.30. The creation of a wide street, as part of the experimental setup, using the endpoints of the facade columns to create an outline which is then extruded.

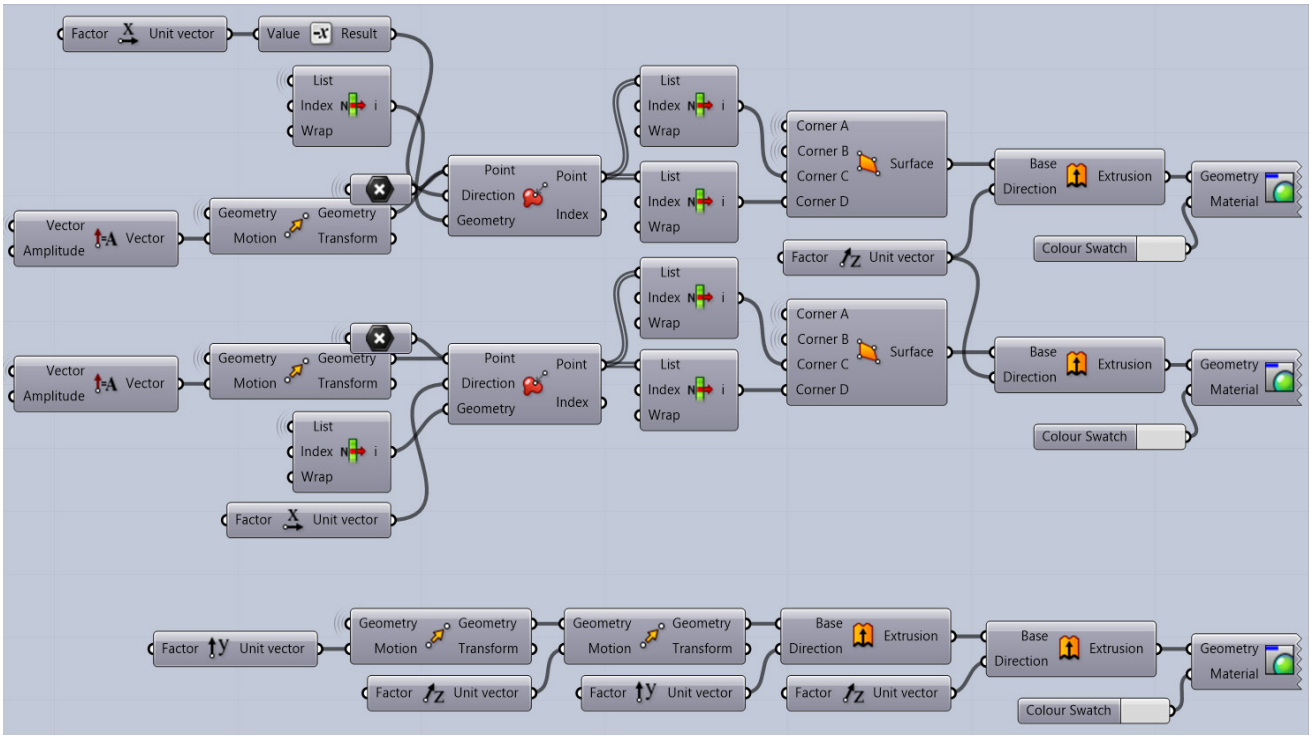


Figure B.31. The creation of two straight facades next to the biocomposite facade, using its endpoints, 4pt surface, and extrude, and the creation of the facade across the street.

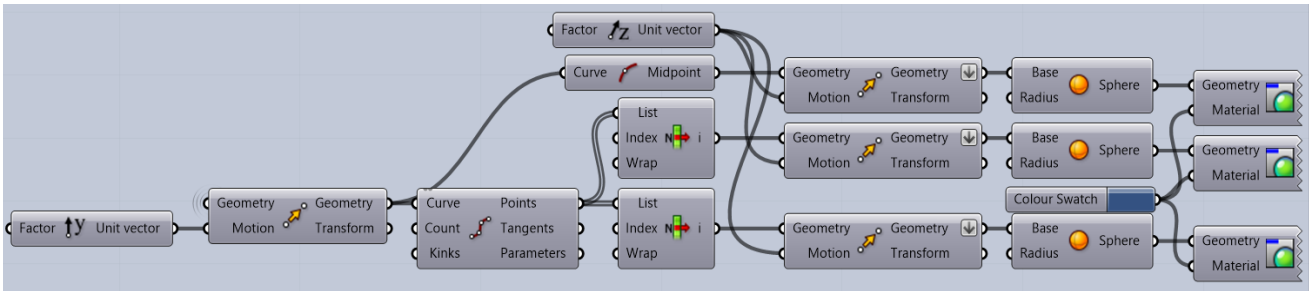


Figure B.32. Setting up the points of the sound receivers in front of the facade at 1.6 meters height, using divide curve based on the endpoints of the facade columns.

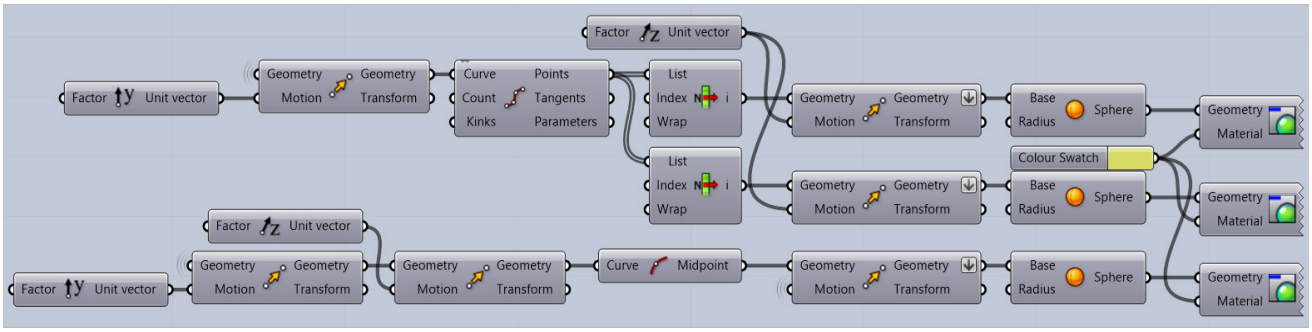


Figure B.33. Setting up the points for the sound sources in the car lanes at 0.5 meters height, and creating a panel with the ranging sound levels across octave bands.

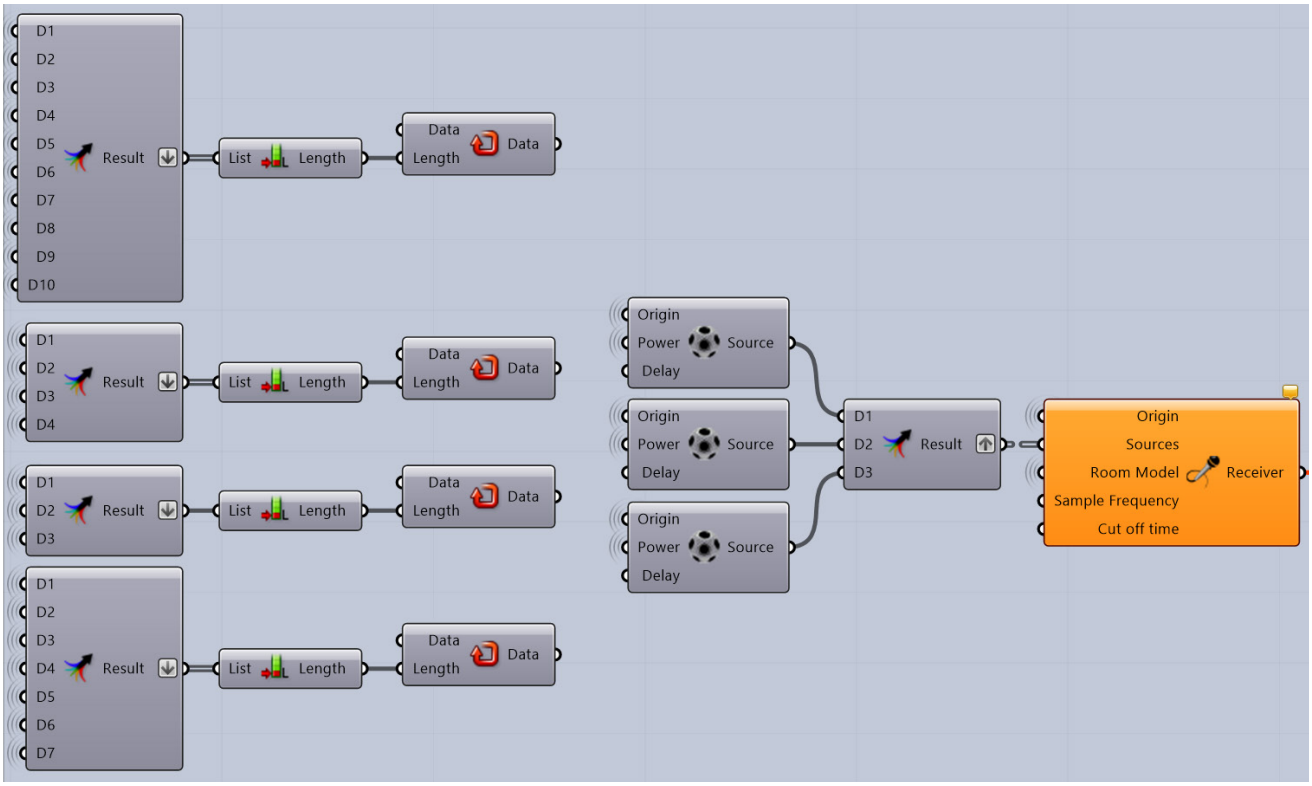


Figure B.34. Allocating materials inside Rhino3D to the facade surfaces - using merge breps and repeating material layer indices - and creating a stationary receiver model.

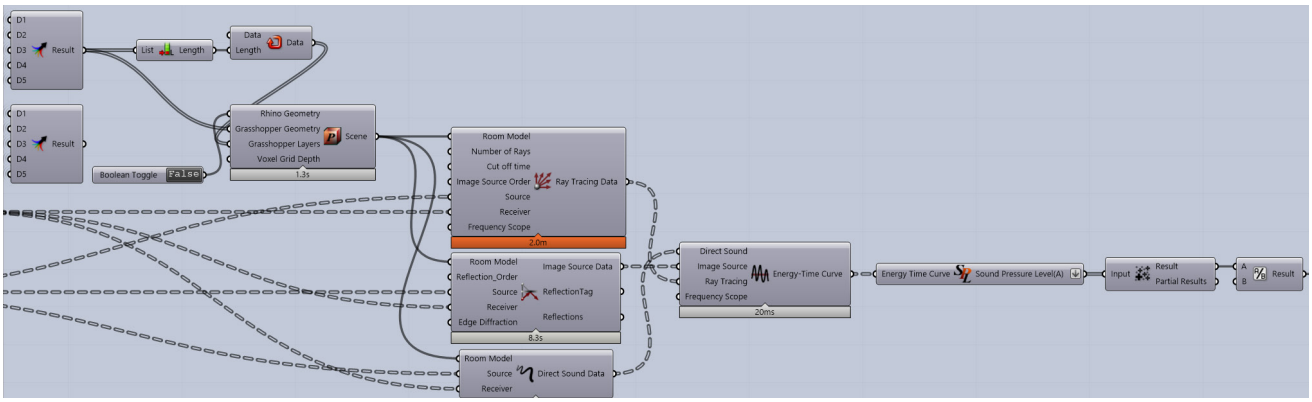


Figure B.35. Setting up a polygon scene based on geometries, a raytracing and image source component based on sound sources and receivers, and computing the SPL.

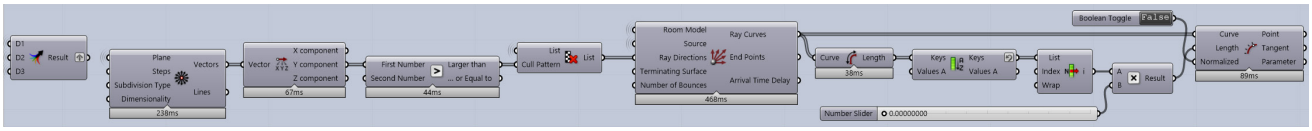


Figure B.36. Setting up a visual raytracing simulation to inspect the ray reflections on the curved surfaces of the facade, using the visualise PachyDerm rays component.

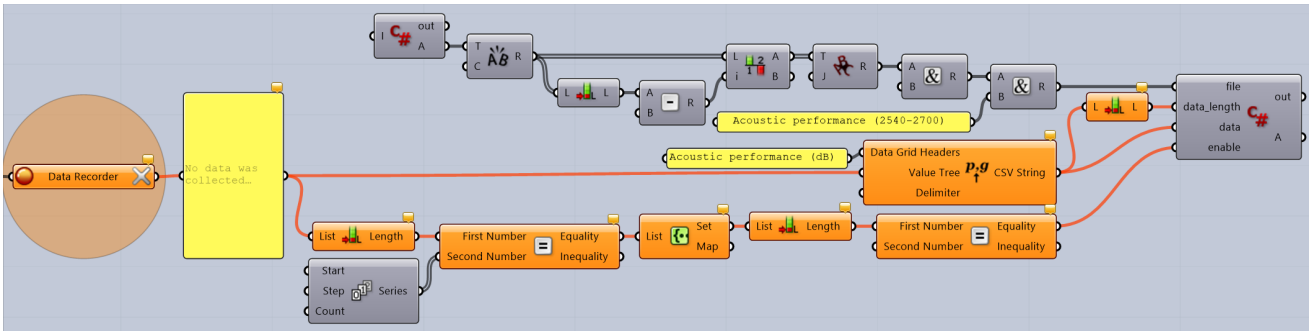


Figure B.37. Calculating the average SPL at each absorber and then calculating the average SPL of the facade, and a data recorder paired with an automatic csv saver.

Inspired by:
- freeCodeCamp (2022, June 15). *Machine learning for everybody – Full course* [Video]. YouTube. https://www.youtube.com/watch?v=i_LwzRVP7bg
- JustGlowing (n.d.). *MiniSOM*. GitHub. https://github.com/JustGlowing/minisom/blob/master/minisom.py
- Topil, L. (2024, March 26). *Understanding Linear Regression: Building and Evaluating Simple and Multiple Linear Regression Models with Python*. Medium. https://medium.com/@lekhatopil/understanding-linear-regression-building-and-evaluating-simple-and-multiple-linear-regression-7cc07068e6d0

C

Self-Organising Map

```python
# ------------------------------------------------------------------------------------------------------------------#
# Importing libraries
# ------------------------------------------------------------------------------------------------------------------#
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from minisom import MiniSom
import pickle
from scipy.interpolate import griddata
import os
import seaborn as sns
from sklearn.metrics.pairwise import euclidean_distances
from itertools import combinations
from scipy.spatial.distance import pdist, squareform
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Preparing training vectors
# ------------------------------------------------------------------------------------------------------------------#
np.random.seed(42)
Y1 = Y2 = Y3 = np.array([-2, -1, 0, 1, 2])
def prepare_input_vectors():
    np_input_vectors = np.array(np.meshgrid(Y1, Y2, Y3)).T.reshape(-1, 3)
    df_input_vectors = pd.DataFrame(np_input_vectors, columns=['Y1', 'Y2', 'Y3'])
    scaler = MinMaxScaler()
    normalized_vectors = scaler.fit_transform(df_input_vectors)
    df_normalized_vectors = pd.DataFrame(normalized_vectors, columns=['Y1', 'Y2', 'Y3'])
    print("\n1. Original input vectors:\n", df_input_vectors.describe())
    print("\n2. Normalized input vectors:\n", df_normalized_vectors.describe())
    return df_normalized_vectors
input_vectors = prepare_input_vectors()
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Testing data preparation
# ------------------------------------------------------------------------------------------------------------------#
def variance_contribution(input_vectors):
    df = pd.DataFrame(input_vectors, columns=['Y1', 'Y2', 'Y3'])
    variance = df.var()
    print("\nVariance contribution plot:\n", variance)
variance_contribution(input_vectors)

def plot_pairwise_distances(input_vectors):
    distances = pdist(input_vectors, metric='euclidean')
    plt.hist(distances, bins=50, alpha=0.7)
    plt.xlabel('Distances')
    plt.ylabel('Number of pairs')
    plt.title('Pairwise plot')
    plt.show()
plot_pairwise_distances(input_vectors)

def plot_correlation(input_vectors):
    df = pd.DataFrame(input_vectors, columns=['Y1', 'Y2', 'Y3'])
    corr = df.corr()
    sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
    plt.title('Correlation plot')
    plt.show()
plot_correlation(input_vectors)
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Training SOM
# ------------------------------------------------------------------------------------------------------------------#
def train_som(training_data, grid_size=(15, 15), ordering_epochs=900, tuning_epochs=100, initial_radius=0.945, final_radius=0.195, initial_lr=0.425, final_lr=0.02, batch_size=5, run_number=1):
    som = MiniSom(grid_size[0], grid_size[1], training_data.shape[1], sigma=initial_radius, learning_rate=initial_lr, topology='hexagonal', random_seed=42)
    som.random_weights_init(training_data)
    error_tracking = []
    total_epochs = ordering_epochs + tuning_epochs
    fig, ax = plt.subplots()
    plt.ion()
    for epoch in range(total_epochs):
        radius = initial_radius * ((final_radius / initial_radius) ** (epoch / total_epochs))
        lr = initial_lr * ((final_lr / initial_lr) ** (epoch / total_epochs))
        som.sigma = radius
        som.learning_rate = lr
        np.random.shuffle(training_data)
        for i in range(0, len(training_data), batch_size):
            batch_data = training_data[i:i + batch_size]
            som.train_batch(batch_data, num_iteration=batch_data.shape[0])
        QE = som.quantization_error(training_data)
        TE = som.topographic_error(training_data)
        combined_error = (QE + TE) / 2
        error_tracking.append((epoch + 1, QE, TE, combined_error))
        ax.clear()
        plot_training_errors_live(error_tracking, ax)
        plt.pause(0.01)
    plt.savefig(f'som_training_errors_{run_number}.png', dpi=300, bbox_inches='tight')
    plt.ioff()
    plt.close(fig)
    best_entry = min(error_tracking, key=lambda x: x[3])
    best_quantization_error = best_entry[1]
    best_topographic_error = best_entry[2]
    best_combined_error = best_entry[3]
    print(f"Best Combined Error: {best_combined_error:.4f}")
    print(f"Best Quantization Error (QE): {best_quantization_error:.4f}")
    print(f"Best Topographic Error (TE): {best_topographic_error:.4f}")
    return som, error_tracking

def plot_training_errors_live(error_tracking, ax):
    epochs = [entry[0] for entry in error_tracking]
    QE = [entry[1] for entry in error_tracking]
    TE = [entry[2] for entry in error_tracking]
    combined_errors = [entry[3] for entry in error_tracking]
    ax.plot(epochs, QE, label='Quantization Error (QE)')
    ax.plot(epochs, TE, label='Topographic Error (TE)')
    ax.plot(epochs, combined_errors, label='Combined Error', linestyle='--')
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Error')
    ax.set_ylim(0, 1)
    ax.set_title('Live SOM Training Errors Over Epochs')
    ax.legend()
    ax.grid(True, linestyle='--', linewidth=0.5, color='grey', which='both')

def hyperparameter_optimization(training_data, hyperparams_list):
    best_error = float('inf')
    best_hyperparams = None
    best_error_tracking = None
    best_som = None
    for idx, params in enumerate(hyperparams_list):
        print(f"Running optimization for hyperparameter set {idx + 1}/{len(hyperparams_list)}: {params}")
```

```
        som, error_tracking = train_som(
            training_data=training_data,
            grid_size=(15, 15),
            ordering_epochs=900,
            tuning_epochs=100,
            initial_radius=params['initial_radius'],
            final_radius=params['final_radius'],
            initial_lr=params['initial_lr'],
            final_lr=params['final_lr'],
            run_number=idx + 1)
        combined_errors = [entry[3] for entry in error_tracking]
        min_combined_error = min(combined_errors)
        if min_combined_error < best_error:
            best_error = min_combined_error
            best_hyperparams = params
            best_error_tracking = error_tracking
            best_som = som
            with open('best_overall_som_model.p', 'wb') as outfile:
                pickle.dump(som, outfile)
    print(f"\nBest hyperparameters found: {best_hyperparams}")
    print(f"Best combined error: {best_error:.4f}")
    return best_hyperparams, best_error, best_error_tracking, best_som
hyperparameter_settings = [{'initial_radius': 0.945, 'final_radius': 0.195, 'initial_lr': 0.425, 'final_lr': 0.02}]
input_vectors = input_vectors.to_numpy() if isinstance(input_vectors, pd.DataFrame) else input_vectors
best_params, best_error, error_tracking, best_som = hyperparameter_optimization(input_vectors, hyperparameter_settings)
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Training visualization SOM
# ------------------------------------------------------------------------------------------------------------------#
def load_best_som(filename='best_overall_som_model.p'):
    with open(filename, 'rb') as infile:
        som = pickle.load(infile)
    return som
best_som = load_best_som()

def count_activated_nodes_training(som, data):
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    activation_map = np.zeros((grid_rows, grid_cols))
    for vector in data.values:
        bmu = som.winner(vector)
        activation_map[bmu] += 1
    num_activated_nodes = np.sum(activation_map > 0)
    total_nodes = grid_rows * grid_cols
    activation_percentage = (num_activated_nodes / total_nodes) * 100
    print(f'Activated Nodes: {num_activated_nodes}/{total_nodes} ({activation_percentage:.2f}%)')
    return activation_map > 0
activated_nodes_mask = count_activated_nodes_training(best_som, input_vectors)

def plot_umatrix(som):
    weights = som.get_weights()
    grid_rows, grid_cols = weights.shape[0], weights.shape[1]
    umatrix = np.zeros((grid_rows, grid_cols))
    for i in range(grid_rows):
        for j in range(grid_cols):
            neighbors = [(i + dr, j + dc) for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]
                         if 0 <= i + dr < grid_rows and 0 <= j + dc < grid_cols]
            dist_sum = sum(np.linalg.norm(weights[i, j] - weights[n[0], n[1]]) for n in neighbors)
            umatrix[i, j] = dist_sum / len(neighbors)
    plt.imshow(umatrix, cmap='coolwarm', interpolation='nearest', vmin=0)
    plt.colorbar()
    plt.xticks(np.arange(0, grid_cols, 1))
    plt.yticks(np.arange(0, grid_rows, 1))
    plt.title('U-matrix plot')
    plt.show()
plot_umatrix(best_som)
```

```
def plot_QE_errors(som, data):
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    errors = np.zeros((grid_rows, grid_cols))
    bmus = [som.winner(d) for d in data.values]
    for i in range(grid_rows):
        for j in range(grid_cols):
            weights = som.get_weights()[i, j]
            bmu_data_points = [data.values[i] for i, bmu in enumerate(bmus) if bmu == (i, j)]
            if bmu_data_points:
                quantization_error = np.mean([np.linalg.norm(weights - d) for d in bmu_data_points])
                errors[i, j] = quantization_error
            else:
                errors[i, j] = np.nan
    plt.imshow(errors, cmap='coolwarm', interpolation='nearest', vmin=0, vmax=1)
    plt.colorbar()
    plt.xticks(np.arange(0, grid_cols, 1))
    plt.yticks(np.arange(0, grid_rows, 1))
    plt.title('Quantization error plot', fontsize=14)
    plt.show()
plot_QE_errors(best_som, input_vectors)

def plot_variables(som, num_features, feature_names):
    weights = som.get_weights()
    fig, axs = plt.subplots(1, num_features, figsize=(4 * num_features, 4))
    for i in range(num_features):
        ax = axs[i]
        weight_plane = weights[:, :, i]
        im = ax.imshow(weight_plane, cmap='coolwarm', aspect='equal', vmin=0, vmax=1)
        ax.set_title(feature_names[i])
        fig.colorbar(im, ax=ax, fraction=0.05, pad=0.05)
        ax.set_xticks(np.arange(0, weights.shape[1], 1))
        ax.set_yticks(np.arange(0, weights.shape[0], 1))
    plt.tight_layout()
    plt.show()
feature_names = ['Y1', 'Y2', 'Y3']
plot_variables(best_som, len(feature_names), feature_names)
# ------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------#
# Clustering visualization SOM
# ------------------------------------------------------------------------------------------------------------#
def count_activated_nodes_clustering(som, data):
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    activation_map = np.zeros((grid_rows, grid_cols))
    for x in data.values:
        bmu = som.winner(x)
        activation_map[bmu] += 1
    num_activated_nodes = np.sum(activation_map > 0)
    total_nodes = grid_rows * grid_cols
    activation_percentage = (num_activated_nodes / total_nodes) * 100
    print(f'Activated Nodes: {num_activated_nodes}/{total_nodes} ({activation_percentage:.2f}%)')
count_activated_nodes_clustering(best_som, input_vectors)

def plot_2D_activation_surface(som, data):
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    activation_map = np.zeros((grid_rows, grid_cols))
    for x in data.values:
        bmu = som.winner(x)
        activation_map[bmu] += 1
    activation_map_normalized = activation_map / np.max(activation_map)
    plt.figure(figsize=(6, 6))
    for row in range(grid_rows):
        for column in range(grid_cols):
            if activation_map[row, column] > 0:
                plt.scatter(column, row, s=activation_map_normalized[row, column] * 100, color='grey', alpha=0.6)
            else:
                plt.scatter(column, row, s=10, color='lightgrey', alpha=0.3)
    plt.xticks(np.arange(0, grid_cols, 1))
```

```
    plt.yticks(np.arange(0, grid_rows, 1))
    plt.title('2D SOM Network plot', fontsize=14)
    plt.gca().invert_yaxis()
    plt.show()
plot_2D_activation_surface(best_som, input_vectors)

def plot_3D_activation_surface(som, data):
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    activation_map = np.zeros((grid_rows, grid_cols))
    for x in data.values:
        bmu = som.winner(x)
        activation_map[bmu] += 1
    activation_map_normalized = activation_map / np.max(activation_map)
    high_res_factor = 100
    x_high_res = np.linspace(0, grid_rows - 1, grid_rows * high_res_factor)
    y_high_res = np.linspace(0, grid_cols - 1, grid_cols * high_res_factor)
    X_high_res, Y_high_res = np.meshgrid(x_high_res, y_high_res)
    X, Y = np.meshgrid(np.arange(grid_rows), np.arange(grid_cols))
    activation_map_high_res = griddata((X.flatten(), Y.flatten()), activation_map_normalized.flatten(),
(X_high_res, Y_high_res), method='cubic')
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(X_high_res, Y_high_res, activation_map_high_res, cmap='plasma', edgecolor='none',
linewidth=0, antialiased=True)
    ax.set_title('3D SOM Network')
    ax.set_zlim(0, 1)
    ax.set_xticks(np.arange(0, grid_rows, 1))
    ax.set_yticks(np.arange(0, grid_cols, 1))
    plt.show()
plot_3D_activation_surface(best_som, input_vectors)
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Extracting node design vectors and checks
# ------------------------------------------------------------------------------------------------------------------#
def load_best_som(filename='best_overall_som_model.p'):
    with open(filename, 'rb') as infile:
        som = pickle.load(infile)
    return som
best_som = load_best_som()
def get_node_assignments(som, input_data):
    node_assignments = {}
    grid_rows, grid_cols = som.get_weights().shape[0], som.get_weights().shape[1]
    for row in range(grid_rows):
        for column in range(grid_cols):
            node_assignments[(row, column)] = []
    for _, vector in input_data.iterrows():
        bmu = som.winner(vector.values)
        node_assignments[bmu].append(vector.values)
    return node_assignments
def save_node_assignments(node_assignments, filename='node_assignments.csv'):
    data_to_save = []
    for node, vectors in node_assignments.items():
        for vector in vectors:
            row = [node[0], node[1]] + vector.tolist()
            data_to_save.append(row)
    columns = ['row', 'col'] + [f'Y{i+1}' for i in range(input_vectors.shape[1])]
    df = pd.DataFrame(data_to_save, columns=columns)
    df.to_csv(filename, index=False)
node_assignments = get_node_assignments(best_som, input_vectors)
save_node_assignments(node_assignments, 'node_assignments.csv')
node_data = pd.read_csv('node_assignments.csv')
def map_y_values(y_value):
    mapping = {0: -1, 0.25: -0.5, 0.5: 0, 0.75: 0.5, 1: 1}
    return mapping[y_value]
node_data['Y1'] = node_data['Y1'].map(map_y_values)
node_data['Y2'] = node_data['Y2'].map(map_y_values)
node_data['Y3'] = node_data['Y3'].map(map_y_values)
```

```
extra_columns = pd.DataFrame(np.tile([-0.5, 0.0, 0.05, 0.05, 0.05, 1], (node_data.shape[0], 1)), columns=['X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1'])
altered_node_data = pd.concat([node_data, extra_columns], axis=1)
altered_node_data.to_csv('altered_node_assignments.csv', index=False)
node_design_vectors_additional = altered_node_data.iloc[:, 2:]
node_design_vectors_additional.to_csv('node_design_vectors.csv', index=False, header=False)

def check_duplicates(node_data):
    duplicates = node_data.duplicated(subset=['Y1', 'Y2', 'Y3'], keep=False)
    dup_count = duplicates.sum()
    print(f"Total duplicates found: {dup_count}")
    print("Duplicate vectors:", node_data[duplicates])
def check_value_representation(node_data, values=[-0.5, 0.5]):
    for value in values:
        count = (node_data[['Y1', 'Y2', 'Y3']] == value).sum().sum()
        print(f"Total occurrences of {value}: {count}")
def prepare_original_combinations():
    Y1 = Y2 = Y3 = np.array([-2, -1, 0, 1, 2])
    np_input_vectors = np.array(np.meshgrid(Y1, Y2, Y3)).T.reshape(-1, 3)
    df_input_vectors = pd.DataFrame(np_input_vectors, columns=['Y1', 'Y2', 'Y3'])
    scaler = MinMaxScaler()
    normalized_vectors = scaler.fit_transform(df_input_vectors)
    return pd.DataFrame(normalized_vectors, columns=['Y1', 'Y2', 'Y3'])
def check_unique_combinations(original_data, node_data):
    missing_combinations = original_data[~original_data.isin(node_data[['Y1', 'Y2', 'Y3']].to_dict(orient='list')).all(axis=1)]
    print(f"Missing combinations not mapped to any BMU: {len(missing_combinations)}", missing_combinations)
def check_proximity(node_data):
    grouped_nodes = node_data.groupby(['row', 'col'])
    for node, vectors in grouped_nodes:
        if len(vectors) > 1:
            vector_values = vectors[['Y1', 'Y2', 'Y3']].values
            ed_matrix = euclidean_distances(vector_values)
            print(f"Node {node} has {len(vectors)} vectors with pairwise ED:")
            print(pd.DataFrame(ed_matrix, index=vectors.index, columns=vectors.index))

check_duplicates(node_data)
check_value_representation(node_data)
original_combinations = prepare_original_combinations()
node_data = pd.read_csv('node_assignments.csv')
check_unique_combinations(original_combinations, node_data)
check_proximity(node_data)
# ------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------#
# Extracting design vectors for KAN
# ------------------------------------------------------------------------------------------------------------#
Y1 = Y2 = Y3 = np.array([-1, -0.5, 0, 0.5, 1])
X2 = np.array([-0.5, 0.5])
W2 = np.array([0.0, 1.0])
TS1 = SCD1 = np.array([0.05, 0.10, 0.15])
BS2 = np.array([0.05, 0.15])
SPD1 = np.array([0.0, 1.0, 2.0])
additional_vectors = pd.DataFrame()
needed_vectors_count = 4050
initial_vectors_df = pd.read_csv('KAN_training_vectors_1-2700.csv', header=None, decimal=',')
initial_vectors_df.columns = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
initial_vectors_df = initial_vectors_df.astype(float)
all_possible_combinations = np.array(np.meshgrid(Y1, Y2, Y3, X2, W2, TS1, SCD1, BS2, SPD1)).T.reshape(-1, 9)
all_vectors_df = pd.DataFrame(all_possible_combinations, columns=['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1'])
print(all_vectors_df)
remaining_vectors_df = pd.merge(all_vectors_df, initial_vectors_df, on=['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1'], how='outer', indicator=True)
remaining_vectors_df = remaining_vectors_df[remaining_vectors_df['_merge'] == 'left_only'].drop(columns=['_merge'])
remaining_vectors_df = remaining_vectors_df.reset_index(drop=True)
remaining_vectors_np = remaining_vectors_df.to_numpy()
```

```
print(remaining_vectors_df)
combination_variables = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'SCD1', 'BS2']
combination_mesh = np.array(np.meshgrid(Y1, Y2, Y3, X2, W2, SCD1, BS2)).T.reshape(-1, len(combination_variables))
all_combinations_df = pd.DataFrame(combination_mesh, columns=combination_variables)
all_combinations_df['TS1'] = np.random.choice(TS1, size=len(all_combinations_df))
all_combinations_df['SPD1'] = np.random.choice(SPD1, size=len(all_combinations_df))
final_combinations_df = all_combinations_df[['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']]
print(final_combinations_df)
sampled_01_df = pd.merge(final_combinations_df, initial_vectors_df, on=['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1',
'SCD1', 'BS2', 'SPD1'], how='left', indicator=True)
sampled_01_df = sampled_01_df[sampled_01_df['_merge'] == 'left_only'].drop(columns=['_merge'])
sampled_01_df = sampled_01_df.reset_index(drop=True)
print(sampled_01_df)
needed_vectors_count = 4050 - len(sampled_01_df)
print(f"{needed_vectors_count} vectors needed to reach 4050")
sampling_df = pd.merge(remaining_vectors_df, sampled_01_df, on=['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1',
'BS2', 'SPD1'], how='left', indicator=True)
sampling_df = sampling_df[sampling_df['_merge'] == 'left_only'].drop(columns=['_merge'])
sampling_df = sampling_df.reset_index(drop=True)
print(sampling_df)
sampled_02_df = sampling_df.sample(n=needed_vectors_count, random_state=42)
sampled_02_df = sampled_02_df.reset_index(drop=True)
print(sampled_02_df)
final_sampled_df = pd.concat([sampled_01_df, sampled_02_df], ignore_index=True)
print(final_sampled_df)
final_sampled_df.to_csv('KAN_training_vectors_2701-6750.csv', index=False)
num_chunks = (len(final_sampled_df) // 300) + 1
chunk_size = 300
chunks = [final_sampled_df.iloc[i * chunk_size:(i + 1) * chunk_size] for i in range(num_chunks)]
for idx, chunk in enumerate(chunks):
    start_idx = 2701 + idx * chunk_size
    end_idx = start_idx + len(chunk) - 1
    filename = f'KAN_training_vectors_{start_idx}-{end_idx}.csv'
    chunk.to_csv(filename, index=False)
final_sampled_df_updated = final_sampled_df.copy()
final_sampled_df_updated['W2'] = final_sampled_df_updated['W2'].replace({0: -0.475, 1: 0.475})
final_sampled_df_updated['SPD1'] = final_sampled_df_updated['SPD1'].replace({0.0: -0.11, 1.0: 0, 2.0: 0.11})
final_sampled_df_updated.to_csv('KAN_training_vectors_2701-6750_updated_ranges.csv', index=False)
# ------------------------------------------------------------------------------------------------------------#
```

Inspired by:
- freeCodeCamp (2022, June 15). *Machine learning for everybody – Full course* [Video]. YouTube. https://www.youtube.com/watch?v=i_LwzRVP7bg
- Xiaoming, K. (2024). *PyKAN: Tutorials*. GitHub. https://github.com/KindXiaoming/pykan/tree/master/tutorials
- Daniel (2024). *Implementation of a KAN for regression*. GitHub. https://github.com/team-daniel/KAN/blob/master/KAN_regression.ipynb
- Bethell, D. (2024, May 13). *Demystifying Kolmogorov-Arnold Networks: A Beginner-Friendly Guide with Code*. https://daniel-bethell.co.uk/posts/kan/
- DataScienceByExample (2023, June 24). *How to evaluate and visualize regression*. DataScienceByExample. https://www.datasciencebyexample.com/2023/06/24/how-to-evaluate-and-visualize-regression/

D

Kolmogorov-Arnold Network

```
# ------------------------------------------------------------------------------------------------------------------#
# Importing libraries
# ------------------------------------------------------------------------------------------------------------------#
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import os
import torch
from kan import KAN
from kan.utils import create_dataset_from_data, ex_round
import random
import shutil
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Preprocessing data
# ------------------------------------------------------------------------------------------------------------------#
random.seed(42)
device = torch.device('succeeded' if torch.cuda.is_available() else 'failed')
geometry_vectors = pd.read_csv('Dataset/KAN_training_vectors_1-6750.csv', header=None, decimal=',')
geometry_vectors.columns = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
material_use = pd.read_csv('Dataset/1A material use (1-6750).txt', header=0)
solar_performance = pd.read_csv('Dataset/1B solar performance (1-6750).txt', header=0)
acoustic_performance = pd.read_csv('Dataset/1C acoustic performance (1-6750).txt', header=0)
min_length = min(len(geometry_vectors), len(material_use), len(solar_performance), len(acoustic_performance))
geometry_vectors = geometry_vectors.iloc[:min_length].reset_index(drop=True)
material_use = material_use.iloc[:min_length].reset_index(drop=True)
solar_performance = solar_performance.iloc[:min_length].reset_index(drop=True)
acoustic_performance = acoustic_performance.iloc[:min_length].reset_index(drop=True)
material_use.columns, solar_performance.columns, acoustic_performance.columns = ['Material'], ['Solar'],
['Acoustic']
performance_metrics = pd.concat([material_use, solar_performance, acoustic_performance], axis=1)
data = pd.concat([geometry_vectors, performance_metrics], axis=1)
input_features = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
scaler = StandardScaler()
data_standardized = pd.DataFrame(scaler.fit_transform(data[input_features]), columns=input_features)
data_standardized[['Material', 'Solar', 'Acoustic']] = data[['Material', 'Solar', 'Acoustic']]
x_input = torch.tensor(data_standardized[input_features].values, dtype=torch.float32, device=device)
y_material = torch.tensor(data_standardized[['Material']].values, dtype=torch.float32, device=device)
y_solar = torch.tensor(data_standardized[['Solar']].values, dtype=torch.float32, device=device)
y_acoustic = torch.tensor(data_standardized[['Acoustic']].values, dtype=torch.float32, device=device)
dataset_material = create_dataset_from_data(x_input, y_material, train_ratio=0.8, device=device)
dataset_solar = create_dataset_from_data(x_input, y_solar, train_ratio=0.8, device=device)
dataset_acoustic = create_dataset_from_data(x_input, y_acoustic, train_ratio=0.8, device=device)

Y1 = Y2 = Y3 = np.array([-1, -0.5, 0, 0.5, 1]); X2 = np.array([-0.5, 0.5]); W2 = np.array([0, 1]); TS1 = SCD1 =
np.array([0.05, 0.10, 0.15]); BS2 = np.array([0.05, 0.15]); SPD1 = np.array([0, 1, 2])
total_input_vectors = np.array(np.meshgrid(Y1, Y2, Y3, X2, W2, TS1, SCD1, BS2, SPD1)).T.reshape(-1, 9)
total_data_standardized = pd.DataFrame(scaler.transform(total_input_vectors), columns=input_features)

def plot_live_validation_loss(val_losses):
    plt.ion()
    fig, ax = plt.subplots()
    ax.set_title('Live Validation Loss Over Steps')
    ax.set_xlabel('Steps')
```

```
    ax.set_ylabel('Validation Loss')
    ax.grid(True, linestyle='--', linewidth=0.5, color='grey', which='both')
    line, = ax.plot([], [], label='Validation Loss')
    ax.legend()
    for step in range(len(val_losses)):
        line.set_xdata(range(step + 1))
        line.set_ydata(val_losses[:step + 1])
        ax.relim()
        ax.autoscale_view()
        plt.pause(0.1)
    plt.pause(10)
    plt.close(fig)
# ----------------------------------------------------------------------------------------------------------------#

# ----------------------------------------------------------------------------------------------------------------#
# Training and evaluating KAN Models
# ----------------------------------------------------------------------------------------------------------------#
def train_kan_model(training_data, testing_data, name, hyperparameter_options, img_folder='training_imgs'):
    if os.path.exists(img_folder):
        shutil.rmtree(img_folder)
    os.makedirs(img_folder, exist_ok=True)
    best_hyperparameters = None
    best_model = None
    best_checkpoint_version = None
    r2 = None
    for hyperparameters in hyperparameter_options:
        num_nodes = hyperparameters['num_nodes']
        grid = hyperparameters['grid']
        k = hyperparameters['k']
        steps = hyperparameters['steps']
        lamb = hyperparameters['lamb']
        lamb_l1 = hyperparameters['lamb_l1']
        lamb_entropy = hyperparameters['lamb_entropy']
        lamb_coef = hyperparameters['lamb_coef']
        lamb_coefdiff = hyperparameters['lamb_coefdiff']
        beta = hyperparameters['beta']
        y_th = hyperparameters['y_th']
        lr = hyperparameters['lr']
        model = KAN(width=[9] + num_nodes + [1], grid=grid, k=k, seed=42, device=device)
        val_losses = []
        fit_results = model.fit(
            training_data,
            opt="LBFGS",
            steps=steps,
            lamb=lamb,
            lamb_l1=lamb_l1,
            lamb_entropy=lamb_entropy,
            lamb_coef=lamb_coef,
            lamb_coefdiff=lamb_coefdiff,
            lr=lr,
            save_fig=False,
            beta=beta,
            in_vars=[f'${var}$' for var in input_features],
            out_vars=[f'{name}'],
            singularity_avoiding=True,
            y_th=y_th,
            update_grid=True,
            grid_update_num=1,
            start_grid_update_step=1,
            stop_grid_update_step=steps - 1)
        if fit_results and isinstance(fit_results, dict):
            val_losses = fit_results.get('test_loss', [])
            plot_live_validation_loss(val_losses)
        feature_scores = model.feature_score.cpu().detach().numpy()
        feature_scores_percentage = 100 * feature_scores / feature_scores.sum()
        feature_scores_percentage_rounded = [f"{score:.2f}" for score in feature_scores_percentage]
        influence_data = pd.DataFrame({'Variable': input_features, 'Influence Percentage': [round(score, 2) for
score in feature_scores_percentage]})
```

```
        csv_filename = f'{name}_influence_percentages.csv'
        influence_data.to_csv(csv_filename, index=False)
        in_vars_with_percentage = [f'{var} ({score}%)' for var, score in zip(input_features,
feature_scores_percentage_rounded)]
        x_test_input = testing_data['test_input']
        y_test_ground_truth = testing_data['test_label']
        y_test_predictions = model(x_test_input).cpu().detach().numpy()
        y_test_ground_truth = y_test_ground_truth.cpu().detach().numpy()
        mse = mean_squared_error(y_test_ground_truth, y_test_predictions)
        r2 = r2_score(y_test_ground_truth, y_test_predictions)
        out_vars_with_r2 = [f'{name} (R²: {r2:.2f})']
        final_img_path = f'final_{name}_plot.jpg'
        model.plot(
            beta=beta,
            metric='forward_n',
            scale=2,
            tick=False,
            sample=False,
            in_vars=in_vars_with_percentage,
            out_vars=out_vars_with_r2,
            varscale=0.175)
        plt.savefig(final_img_path, bbox_inches='tight', dpi=300)
        plt.close()
        node_th = hyperparameters['node_th']
        edge_th = hyperparameters['edge_th']
        model = model.prune(node_th=node_th, edge_th=edge_th)
        pruned_img_path = f'pruned_{name}_plot.jpg'
        model.plot(
            beta=beta,
            metric='forward_n',
            scale=2,
            tick=False,
            sample=False,
            in_vars=in_vars_with_percentage,
            out_vars=out_vars_with_r2,
            varscale=0.175)
        plt.savefig(pruned_img_path, bbox_inches='tight', dpi=300)
        plt.close()
        if fit_results and isinstance(fit_results, dict):
            train_loss = fit_results.get('train_loss', float('inf'))[-1] ** 2
            test_loss = fit_results.get('test_loss', float('inf'))[-1] ** 2
        else:
            train_loss = float('inf')
            test_loss = float('inf')
        checkpoint_version = f'epoch_1_{name}_grid{grid}_k{k}'
        model.saveckpt(f'./model/{checkpoint_version}')
        best_model = model
        best_checkpoint_version = checkpoint_version
        best_hyperparameters = hyperparameters
    return best_checkpoint_version, best_model, r2
material_hyperparameter_options = [{'num_nodes': [4, 2], 'grid': 5, 'k': 3, 'steps': 100, 'lr': 1, 'lamb': 0,
'lamb_l1': 1, 'lamb_entropy': 2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000, 'node_th': 0.01,
'edge_th': 0.01}]
solar_hyperparameter_options = [{'num_nodes': [4, 2], 'grid': 5, 'k': 3, 'steps': 300, 'lr': 1, 'lamb': 0,
'lamb_l1': 1, 'lamb_entropy':2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000, 'node_th': 0.01,
'edge_th': 0.01}]
acoustic_hyperparameter_options = [{'num_nodes': [8, 4], 'grid': 5, 'k': 3, 'steps': 250, 'lr': 1, 'lamb': 0,
'lamb_l1': 1, 'lamb_entropy': 2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000, 'node_th': 0.01,
'edge_th': 0.1}]
best_checkpoint_material, model_material, r2_material = train_kan_model(dataset_material, dataset_material,
'Material Use', material_hyperparameter_options)
best_checkpoint_solar, model_solar, r2_solar = train_kan_model(dataset_solar, dataset_solar, 'Solar Performance',
solar_hyperparameter_options)
best_checkpoint_acoustic, model_acoustic, r2_acoustic = train_kan_model(dataset_acoustic, dataset_acoustic,
'Acoustic Performance', acoustic_hyperparameter_options)

def plot_actual_vs_predicted_with_enhancements(y_testing, y_predictions, feature, r2, num_training, num_total,
unit):
    y_testing = y_testing.flatten()
```

```
    y_predictions = y_predictions.flatten()
    a, b = np.polyfit(y_testing, y_predictions, 1)
    fit_line = a * np.array(y_testing) + b
    computed_r2 = r2_score(y_testing, y_predictions)
    fig = plt.figure(figsize=(6, 6))
    min_validation = min(min(y_testing), min(y_predictions))
    max_validation = max(max(y_testing), max(y_predictions))
    plt.xlim(min_validation, max_validation)
    plt.ylim(min_validation, max_validation)
    scatter_color = 'gray'
    plt.scatter(y_testing, y_predictions, color=scatter_color, alpha=0.65, label='Datapoints', zorder=1)
    plt.plot([min_validation, max_validation], [min_validation, max_validation], color='darkblue', linestyle='--', label='Best Fit', zorder=1)
    plt.plot(y_testing, fit_line, color='red', linestyle='-', label='Actual Fit', zorder=2)
    plt.xlabel(f'Actual {feature} ({unit})')
    plt.ylabel(f'Predicted {feature} ({unit})')
    title = f'KAN (8-4) - Accuracy Plot - {feature} (R²: {r2:.2f})'
    plt.title(title)
    percent_training = round((num_training / num_total) * 100)
    plt.text(0.97, 0.03, f'Training data: {num_training}\nDesign space: {num_total}\nTraining ratio: {percent_training:.2f}%',
             fontsize=10, bbox=dict(facecolor='white', edgecolor='lightgray', alpha=1), verticalalignment='bottom', horizontalalignment='right', transform=plt.gca().transAxes)
    plt.legend(loc='upper left', framealpha=1, edgecolor='lightgray', fancybox=False)
    plt.grid(True)
    save_filename = f'KAN_{feature.replace(" ", "_").lower()}_accuracy_plot.png'
    plt.savefig(save_filename, bbox_inches='tight', dpi=300)
    plt.show()
def get_predictions_and_plot(model, dataset, target_feature, r2_value, num_training, num_total, unit):
    x_testing_tensor = dataset['test_input']
    y_testing_tensor = dataset['test_label']
    with torch.no_grad():
        y_prediction_tensor = model(x_testing_tensor)
    y_testing = y_testing_tensor.cpu().numpy()
    y_pred = y_prediction_tensor.cpu().numpy()
    plot_actual_vs_predicted_with_enhancements(y_testing, y_pred, target_feature, r2_value, num_training, num_total, unit)
get_predictions_and_plot(model_material, dataset_material, 'Material Use', r2_material, num_training=6750, num_total=27000, unit='m³')
get_predictions_and_plot(model_solar, dataset_solar, 'Solar Performance', r2_solar, num_training=6750, num_total=27000, unit='kWh/m²')
get_predictions_and_plot(model_acoustic, dataset_acoustic, 'Acoustic Performance', r2_acoustic, num_training=6750, num_total=27000, unit='dB')

def training_process(training_data, target_name, hyperparameter_options, image='training_imgs'):
    if os.path.exists(image):
        shutil.rmtree(image)
    os.makedirs(image, exist_ok=True)
    for hyperparameters in hyperparameter_options:
        num_nodes = hyperparameters['num_nodes']
        grid = hyperparameters['grid']
        k = hyperparameters['k']
        steps = hyperparameters['steps']
        lamb = hyperparameters['lamb']
        lamb_l1 = hyperparameters['lamb_l1']
        lamb_entropy = hyperparameters['lamb_entropy']
        lamb_coef = hyperparameters['lamb_coef']
        lamb_coefdiff = hyperparameters['lamb_coefdiff']
        beta = hyperparameters['beta']
        y_th = hyperparameters['y_th']
        lr = hyperparameters['lr']
        model = KAN(width=[9] + num_nodes + [1], grid=grid, k=k, seed=42, device=device)
        for step in range(1, steps+1):
            fit_results = model.fit(
                training_data,
                opt="LBFGS",
                steps=step,
                lamb=lamb,
```

```
                lamb_l1=lamb_l1,
                lamb_entropy=lamb_entropy,
                lamb_coef=lamb_coef,
                lamb_coefdiff=lamb_coefdiff,
                lr=lr,
                save_fig=False,
                beta=beta,
                in_vars=None,
                out_vars=None,
                singularity_avoiding=True,
                y_th=y_th,
                update_grid=True,
                grid_update_num=1,
                start_grid_update_step=1,
                stop_grid_update_step=steps)
            img_path = os.path.join(image, f'{target_name}_step_{step:03d}.jpg')
            model.plot(
                beta=beta,
                metric='forward_n',
                scale=2,
                tick=False,
                sample=False,
                in_vars=None,
                out_vars=None)
            plt.savefig(img_path, bbox_inches='tight', dpi=300)
            plt.close()
material_hyperparameter_options = [{'num_nodes': [4, 2], 'grid': 5, 'k': 3, 'steps': 10, 'lr': 1, 'lamb': 0, 'lamb_l1': 1, 'lamb_entropy': 2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000}]
solar_hyperparameter_options = [{'num_nodes': [4, 2], 'grid': 5, 'k': 3, 'steps': 10, 'lr': 1, 'lamb': 0, 'lamb_l1': 1, 'lamb_entropy':2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000}]
acoustic_hyperparameter_options = [{'num_nodes': [8, 4], 'grid': 5, 'k': 3, 'steps': 10, 'lr': 1, 'lamb': 0, 'lamb_l1': 1, 'lamb_entropy': 2, 'lamb_coef': 0, 'lamb_coefdiff': 0, 'beta': 100, 'y_th': 1000}]
training_process(dataset_material, 'Material Use', material_hyperparameter_options, image='training_video_frames_Material')
training_process(dataset_solar, 'Solar Performance', solar_hyperparameter_options, image='training_video_frames_Solar')
training_process(dataset_acoustic, 'Acoustic Performance', acoustic_hyperparameter_options, image='training_video_frames_Acoustic')
# ----------------------------------------------------------------------------------------------------------------------#

# ----------------------------------------------------------------------------------------------------------------------#
# Predicting with KAN Models
# ----------------------------------------------------------------------------------------------------------------------#
def predict_and_save_full_data(model, standardized_input_data, original_input_data, input_features, output_filename):
    model.eval()
    with torch.no_grad():
        y_prediction_tensor = model(standardized_input_data)
    y_prediction = y_prediction_tensor.cpu().numpy().flatten().round(2)
    df_predictions = pd.DataFrame(original_input_data, columns=input_features)
    prediction_column_name = output_filename.replace('_Predictions.csv', ' Predicted')
    df_predictions[prediction_column_name] = y_prediction
    df_predictions.to_csv(output_filename, index=False)
full_input_data_standardized = torch.tensor(scaler.transform(total_input_vectors), dtype=torch.float32, device=device)
predict_and_save_full_data(model_material, full_input_data_standardized, total_input_vectors, input_features, 'Material_Predictions.csv')
predict_and_save_full_data(model_solar, full_input_data_standardized, total_input_vectors, input_features, 'Solar_Predictions.csv')
predict_and_save_full_data(model_acoustic, full_input_data_standardized, total_input_vectors, input_features, 'Acoustic_Predictions.csv')
# ----------------------------------------------------------------------------------------------------------------------#
```

Inspired by:
- freeCodeCamp (2022, June 15). *Machine learning for everybody – Full course* [Video]. YouTube. https://www.youtube.com/watch?v=i_LwzRVP7bg
- TensorFlow (n.d.). *Multilayer perceptrons for digit recognition with Core APIs*. TensorFlow. https://www.tensorflow.org/guide/core/mlp_core
- DataScienceByExample (2023, June 24). *How to evaluate and visualize regression*. DataScienceByExample. https://www.datasciencebyexample.com/2023/06/24/how-to-evaluate-and-visualize-regression/

E

Multi-Layer Perceptron

```
# ------------------------------------------------------------------------------------------------------------------#
# Importing libraries
# ------------------------------------------------------------------------------------------------------------------#
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import tensorflow as tf
import random
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Preprocessing data
# ------------------------------------------------------------------------------------------------------------------#
random.seed(42)
geometry_vectors = pd.read_csv('Dataset/KAN_training_vectors_1-1500.csv', header=None, decimal=',')
geometry_vectors.columns = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
material_use = pd.read_csv('Dataset/1A material use (1-1500).txt', header=0)
solar_performance = pd.read_csv('Dataset/1B solar performance (1-1500).txt', header=0)
acoustic_performance = pd.read_csv('Dataset/1C acoustic performance (1-1500).txt', header=0)
min_length = min(len(geometry_vectors), len(material_use), len(solar_performance), len(acoustic_performance))
geometry_vectors = geometry_vectors.iloc[:min_length].reset_index(drop=True)
material_use = material_use.iloc[:min_length].reset_index(drop=True)
solar_performance = solar_performance.iloc[:min_length].reset_index(drop=True)
acoustic_performance = acoustic_performance.iloc[:min_length].reset_index(drop=True)
material_use.columns, solar_performance.columns, acoustic_performance.columns = ['Material'], ['Solar'],
['Acoustic']
performance_metrics = pd.concat([material_use, solar_performance, acoustic_performance], axis=1)
data = pd.concat([geometry_vectors, performance_metrics], axis=1)
input_features = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
standardscaler = StandardScaler()
scaled = pd.DataFrame(standardscaler.fit_transform(data[input_features]), columns=input_features)
scaled[['Material', 'Solar', 'Acoustic']] = data[['Material', 'Solar', 'Acoustic']]
data_array = scaled.to_numpy()
train, val, test = np.split(data_array[np.random.permutation(len(data_array))], [int(0.7 * len(data_array)),
int(0.85 * len(data_array))])
train = pd.DataFrame(train, columns=scaled.columns)
val = pd.DataFrame(val, columns=scaled.columns)
test = pd.DataFrame(test, columns=scaled.columns)
x_train, x_val, x_test = train[input_features].values, val[input_features].values, test[input_features].values
y_train_material, y_val_material, y_test_material = train['Material'].values, val['Material'].values,
test['Material'].values
y_train_solar, y_val_solar, y_test_solar = train['Solar'].values, val['Solar'].values, test['Solar'].values
y_train_acoustic, y_val_acoustic, y_test_acoustic = train['Acoustic'].values, val['Acoustic'].values,
test['Acoustic'].values

def plot_live_training_errors(history):
    plt.ion()
    fig, ax = plt.subplots()
    ax.set_title('Live Training Loss Over Epochs')
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Loss')
    ax.grid(True, linestyle='--', linewidth=0.5, color='grey', which='both')
    line1, = ax.plot([], [], label='Training Loss')
    line2, = ax.plot([], [], label='Validation Loss')
    ax.legend()
```

```
    for epoch in range(len(history.history['loss'])):
        line1.set_xdata(range(epoch + 1))
        line1.set_ydata(history.history['loss'][:epoch + 1])
        line2.set_xdata(range(epoch + 1))
        line2.set_ydata(history.history['val_loss'][:epoch + 1])
        ax.relim()
        ax.autoscale_view()
        plt.pause(0.1)
    plt.pause(1)
    plt.close(fig)
# ----------------------------------------------------------------------------------------------------------#

# ----------------------------------------------------------------------------------------------------------#
# Training Material MLP 01
# ----------------------------------------------------------------------------------------------------------#
callbacks_list = [
    tf.keras.callbacks.ModelCheckpoint("Acoustic_MLP_model", save_best_only=True, save_format="tf"),
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=50, verbose=1, restore_best_weights=True)]
def train_model_material(x_train, y_train, x_val, y_val, nodes_1, nodes_2, dropout_probability, learning_rate,
batch_size, epochs):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(nodes_1, activation='relu', input_shape=(len(input_features),)),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(nodes_2, activation='relu'),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(1, activation='linear')])
    model.compile(optimizer=tf.optimizers.Adam(learning_rate=learning_rate), loss='MSE')
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=0, validation_data=
(x_val, y_val), callbacks=callbacks_list)
    return model, history
least_validation_loss_material = float('inf')
best_model_material = None
best_hyperparameters_material = None
nodes_1 = 4
nodes_2 = 2
epochs = 500
for dropout_probability in [0.01]:
    for learning_rate in [0.0005]:
        for batch_size in [8]:
            model_material, history_material = train_model_material(x_train, y_train_material, x_val,
y_val_material, nodes_1, nodes_2, dropout_probability, learning_rate, batch_size, epochs)
            plot_live_training_errors(history_material)
            validation_loss_material = model_material.evaluate(x_val, y_val_material, verbose=0)
            if validation_loss_material < least_validation_loss_material:
                least_validation_loss_material = validation_loss_material
                best_model_material = model_material
                best_hyperparameters_material = (dropout_probability, learning_rate, batch_size, epochs)
y_predictions_material = best_model_material.predict(x_test)
recomputed_MSE_material = mean_squared_error(y_test_material, y_predictions_material.flatten())
recomputed_R2_material = r2_score(y_test_material, y_predictions_material.flatten())
# ----------------------------------------------------------------------------------------------------------#

# ----------------------------------------------------------------------------------------------------------#
# Training Solar MLP 01
# ----------------------------------------------------------------------------------------------------------#
callbacks_list = [
    tf.keras.callbacks.ModelCheckpoint("Acoustic_MLP_model", save_best_only=True, save_format="tf"),
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=50, verbose=1, restore_best_weights=True)]
def train_model_solar(x_train, y_train, x_val, y_val, nodes_1, nodes_2, dropout_probability, learning_rate,
batch_size, epochs):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(nodes_1, activation='relu', input_shape=(len(input_features),)),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(nodes_2, activation='relu'),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(1, activation='linear')])
    model.compile(optimizer=tf.optimizers.Adam(learning_rate=learning_rate), loss='MSE')
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=0, validation_data=
```

```
(x_val, y_val), callbacks=callbacks_list)
    return model, history
least_validation_loss_solar = float('inf')
best_model_solar = None
best_hyperparameters_solar = None
nodes_1 = 4
nodes_2 = 2
epochs = 1000
for dropout_probability in [0.001]:
    for learning_rate in [0.005]:
        for batch_size in [8]:
            model_solar, history_solar = train_model_solar(x_train, y_train_solar, x_val, y_val_solar, nodes_1, nodes_2, dropout_probability, learning_rate, batch_size, epochs)
            plot_live_training_errors(history_solar)
            validation_loss_solar = model_solar.evaluate(x_val, y_val_solar, verbose=0)
            if validation_loss_solar < least_validation_loss_solar:
                least_validation_loss_solar = validation_loss_solar
                best_model_solar = model_solar
                best_hyperparameters_solar = (dropout_probability, learning_rate, batch_size, epochs)
y_predictions_solar = best_model_solar.predict(x_test)
recomputed_MSE_solar = mean_squared_error(y_test_solar, y_predictions_solar.flatten())
recomputed_R2_solar = r2_score(y_test_solar, y_predictions_solar.flatten())
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Training Acoustic MLP 01
# ------------------------------------------------------------------------------------------------------------------#
callbacks_list = [
    tf.keras.callbacks.ModelCheckpoint("Acoustic_MLP_model", save_best_only=True, save_format="tf"),
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=50, verbose=1, restore_best_weights=True)]
def train_model_acoustic(x_train, y_train, x_val, y_val, nodes_1, nodes_2, dropout_probability, learning_rate, batch_size, epochs):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(nodes_1, activation='relu', input_shape=(len(input_features),)),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(nodes_2, activation='relu'),
        tf.keras.layers.Dropout(dropout_probability),
        tf.keras.layers.Dense(1, activation='linear')])
    model.compile(optimizer=tf.optimizers.Adam(learning_rate=learning_rate), loss='MSE')
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=0, validation_data=(x_val, y_val), callbacks=callbacks_list)
    return model, history
least_validation_loss_acoustic = float('inf')
best_model_acoustic = None
best_hyperparameters_acoustic = None
nodes_1 = 8
nodes_2 = 4
epochs = 1000
for dropout_probability in [0.00005]:
    for learning_rate in [0.001]:
        for batch_size in [32]:
            model_acoustic, history_acoustic = train_model_acoustic(x_train, y_train_acoustic, x_val, y_val_acoustic, nodes_1, nodes_2, dropout_probability, learning_rate, batch_size, epochs)
            plot_live_training_errors(history_acoustic)
            validation_loss_acoustic = model_acoustic.evaluate(x_val, y_val_acoustic, verbose=0)
            if validation_loss_acoustic < least_validation_loss_acoustic:
                least_validation_loss_acoustic = validation_loss_acoustic
                best_model_acoustic = model_acoustic
                best_hyperparameters_acoustic = (dropout_probability, learning_rate, batch_size, epochs)
y_predictions_acoustic = best_model_acoustic.predict(x_test)
recomputed_MSE_acoustic = mean_squared_error(y_test_acoustic, y_predictions_acoustic.flatten())
recomputed_R2_acoustic = r2_score(y_test_acoustic, y_predictions_acoustic.flatten())
# ------------------------------------------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------------------------------------------#
# Accuracy plots
# ------------------------------------------------------------------------------------------------------------------#
def plot_actual_vs_predicted_with_enhancements(y_testing, y_prediction, feature, accuracy, num_training,
```

```
num_total, unit):
    y_testing = y_testing.flatten()
    y_prediction = y_prediction.flatten()
    a, b = np.polyfit(y_testing, y_prediction, 1)
    fit_line = a * np.array(y_testing) + b
    computed_r2 = r2_score(y_testing, y_prediction)
    fig = plt.figure(figsize=(6, 6))
    min_validation = min(min(y_testing), min(y_prediction))
    max_validation = max(max(y_testing), max(y_prediction))
    plt.xlim(min_validation, max_validation)
    plt.ylim(min_validation, max_validation)
    scatter_color = 'gray'
    plt.scatter(y_testing, y_prediction, color=scatter_color, alpha=0.65, label='Datapoints', zorder=1)
    plt.plot([min_validation, max_validation], [min_validation, max_validation], color='darkblue', linestyle='--
', label='Best Fit', zorder=1)
    plt.plot(y_testing, fit_line, color='red', linestyle='-', label='Actual Fit', zorder=2)
    plt.xlabel(f'Actual {feature} ({unit})')
    plt.ylabel(f'Predicted {feature} ({unit})')
    title = f'MLP (128-64-32-16) - Accuracy Plot - {feature} (R²: {accuracy:.2f})'
    plt.title(title)
    percent_training = round((num_training / num_total) * 100)
    plt.text(0.97, 0.03, f'Training data: {num_training}\nDesign space: {num_total}\nTraining ratio:
{percent_training:.2f}%',
             fontsize=10, bbox=dict(facecolor='white', edgecolor='lightgray', alpha=1),
verticalalignment='bottom', horizontalalignment='right', transform=plt.gca().transAxes)
    plt.legend(loc='upper left', framealpha=1, edgecolor='lightgray', fancybox=False)
    plt.grid(True)
    save_filename = f'MLP_{feature.replace(" ", "_").lower()}_accuracy_plot.png'
    plt.savefig(save_filename, bbox_inches='tight', dpi=300)
    plt.show()
def get_predictions_and_plot(model, x_testing, y_testing, feature, num_training, num_total, unit):
    y_prediction = model.predict(x_testing).flatten()
    accuracy = r2_score(y_testing, y_prediction)
    plot_actual_vs_predicted_with_enhancements(y_testing, y_prediction, feature, accuracy, num_training,
num_total, unit)
get_predictions_and_plot(best_model_material, x_test, y_test_material, 'Material Use', num_training=6750,
num_total=27000, unit='m³')
get_predictions_and_plot(best_model_solar, x_test, y_test_solar, 'Solar Performance', num_training=6750,
num_total=27000, unit='kWh/m²')
get_predictions_and_plot(best_model_acoustic, x_test, y_test_acoustic, 'Acoustic', num_training=6750,
num_total=27000, unit='dB')
# ----------------------------------------------------------------------------------------------------------#
```

Inspired by:
- freeCodeCamp (2022, June 15). *Machine learning for everybody – Full course* [Video]. YouTube. https://www.youtube.com/watch?v=i_LwzRVP7bg
- Javatpoint (n.d.). *Sensitivity Analysis to Optimize Process Quality in Python*. Javatpoint. https://www.javatpoint.com/sensitivity-analysis-to-optimize-process-quality-in-python?utm_source
- Kundu, D. (2023, July 30). *Identifying important features using Python*. Medium. https://medium.com/@kundu.deepanjan/a-practical-guide-for-identifying-important-features-using-python-5448f7f99edd

F

Sensitivity analysis

```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import torch
import random

random.seed(42)
device = torch.device('succeeded' if torch.cuda.is_available() else 'failed')
geometry_vectors = pd.read_csv('fine-tuning/Fine-tuning_vectors_test.csv', header=None, decimal=',')
geometry_vectors.columns = ['Y1', 'Y2', 'Y3', 'X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
material_use = pd.read_csv('fine-tuning/Material use (216)_test.txt', header=0, decimal='.', names=['Material'])
solar_performance = pd.read_csv('fine-tuning/Solar performance (216)_test.txt', header=0, decimal='.', names=['Solar'])
acoustic_performance = pd.read_csv('fine-tuning/Acoustic performance (216)_test.txt', header=0, decimal='.', names=['Acoustic'])
performance_metrics = pd.concat([material_use, solar_performance, acoustic_performance], axis=1)
input_features = ['X2', 'W2', 'TS1', 'SCD1', 'BS2', 'SPD1']
data = pd.concat([geometry_vectors[input_features], performance_metrics], axis=1)
print("\nData without Y1, Y2, Y3:\n", data)
standardscaler = StandardScaler()
standardised = pd.DataFrame(standardscaler.fit_transform(data[input_features]), columns=input_features)
standardised[['Material', 'Solar', 'Acoustic']] = data[['Material', 'Solar', 'Acoustic']]
performance_metrics_names = ['Material', 'Solar', 'Acoustic']
print("\nStandardized Data with Performance Metrics:\n", standardised)

def sensitivity_analysis(metric_name):
    influence_percentages = {}
    performance_range = data[metric_name].max() - data[metric_name].min()
    for variable in input_features:
        influences = []
        for row in range(len(standardised)):
            for column in range(row + 1, len(standardised)):
                if (standardised.iloc[row][input_features].drop(variable) == standardised.iloc[column][input_features].drop(variable)).all() and standardised.iloc[row][variable] != standardised.iloc[column][variable]:
                    perf_change = data.iloc[column][metric_name] - data.iloc[row][metric_name]
                    influence = (perf_change / performance_range) * 100
                    influences.append(abs(influence))
        avg_influence = np.mean(influences) if influences else 0
        influence_percentages[variable] = avg_influence
    total_influence = sum(influence_percentages.values())
    for variable in influence_percentages:
        influence_percentages[variable] = (influence_percentages[variable] / total_influence * 100) if total_influence != 0 else 0
    return pd.DataFrame.from_dict(influence_percentages, orient='index', columns=[f'Influence_{metric_name}']).round(2)
for metric in performance_metrics_names:
    influence_df = sensitivity_analysis(metric)
    print(f"\nInfluence Percentages for {metric}:\n", influence_df)
    influence_df.to_csv(f'sensitivity_analysis_{metric}.csv')
```

G

Design exploration

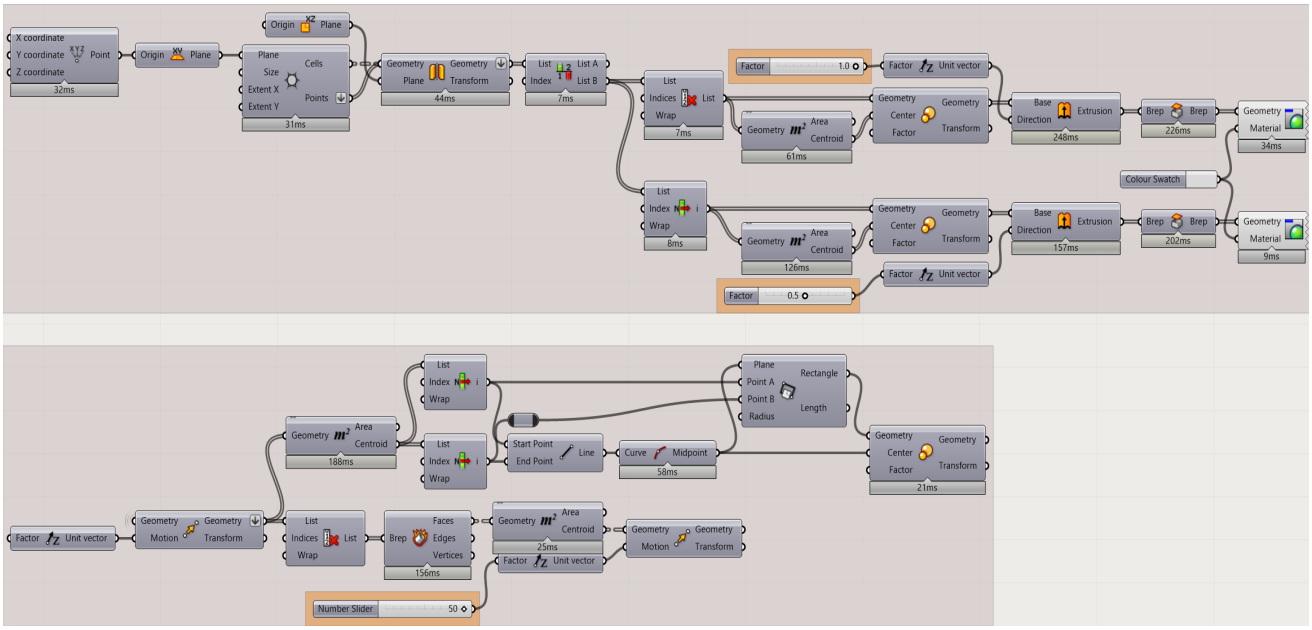


Figure G.1. The creation of the hexagonal grid, using the hexagonal, scale, and extrude component, with a different thickness for hexagons that contain facade designs.

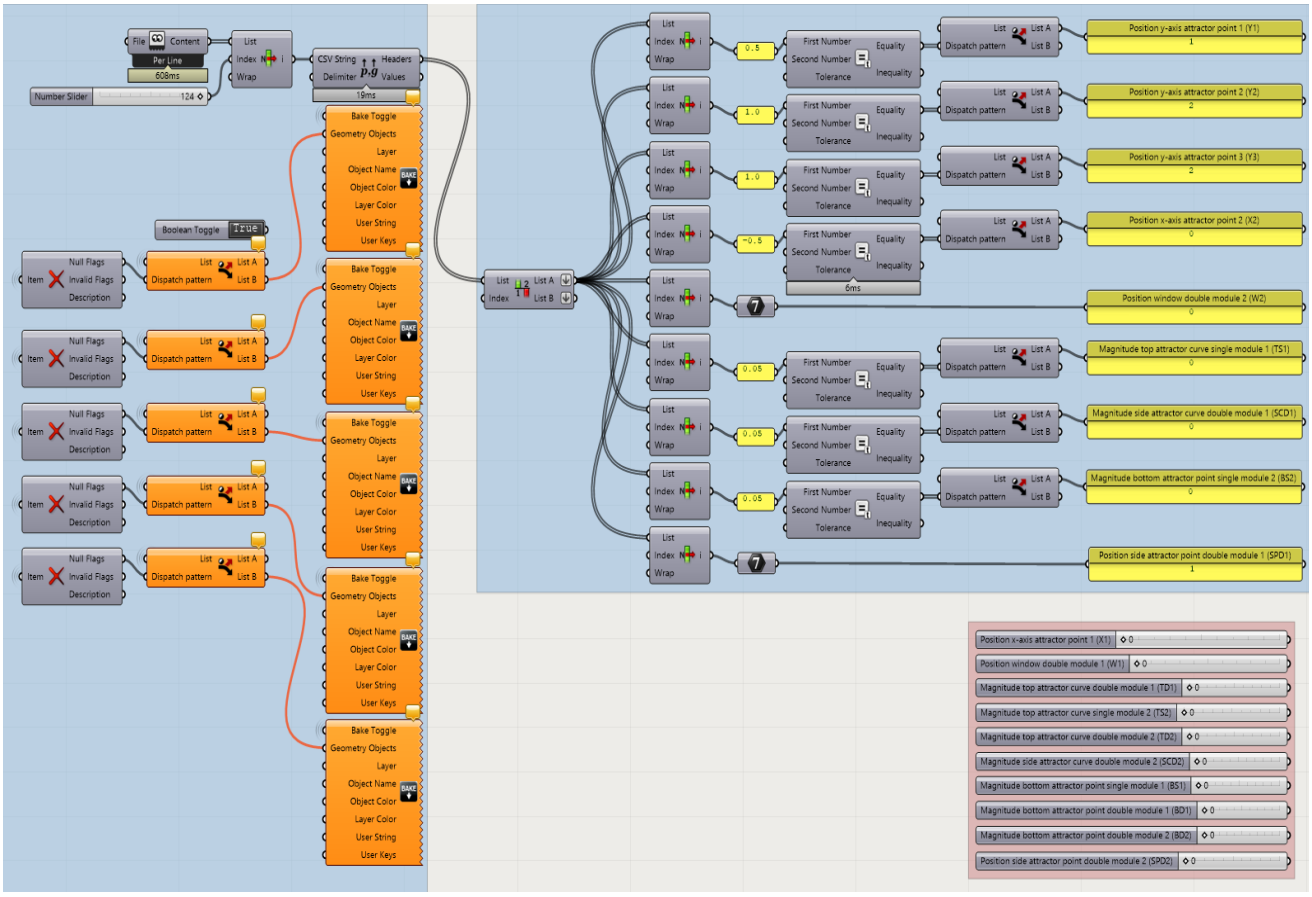


Figure G.2. Automatic baking and material allocation of node design geometries using the object bake component, and the design variables that are not used in exploration.

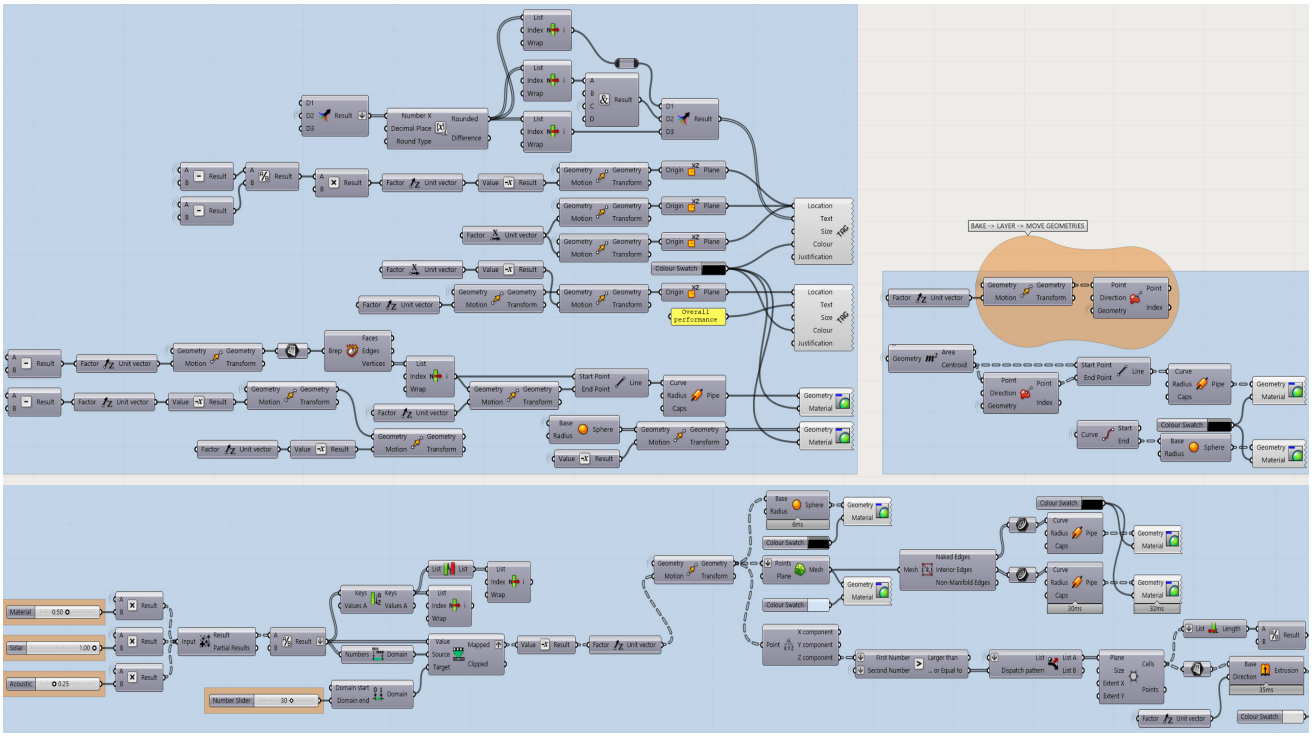


Figure G.3. The creation of a weighted performance surface using remap numbers and Delaunay mesh, a performance threshold plane, a legend, and lines to indicate designs.

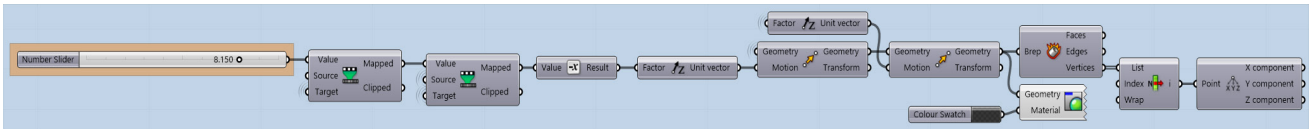


Figure G.4. Creating a slider that controls the height of the performance threshold plane, using the remap numbers component and the height of the performance surface.

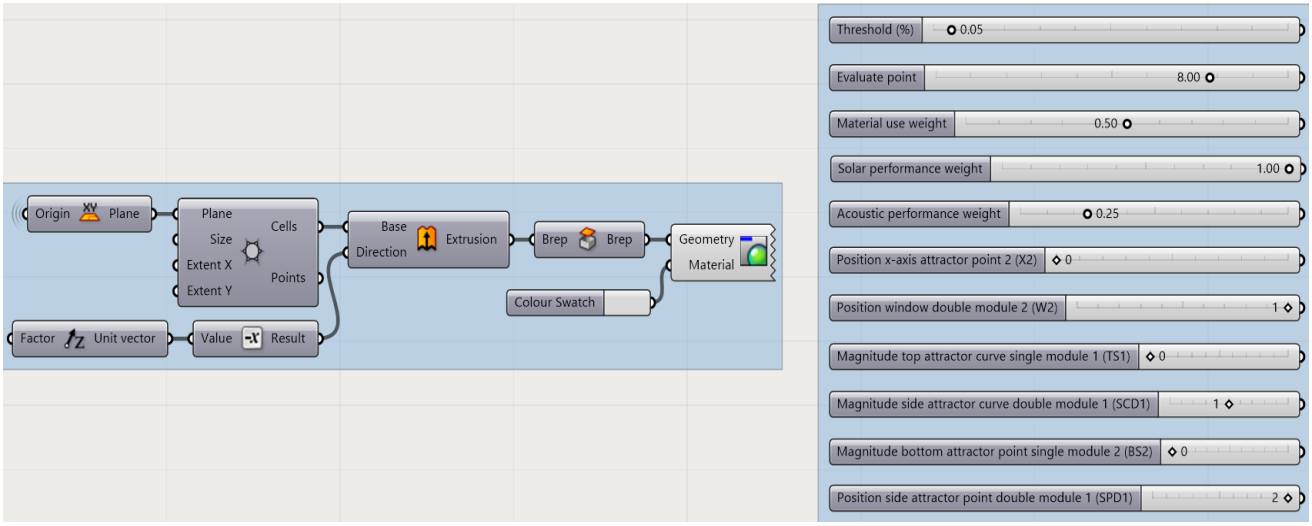


Figure G.5. The start of the design exploration's fine-tuning phase, with the creation of a single hexagonal base with the selected facade design, and the control panel.

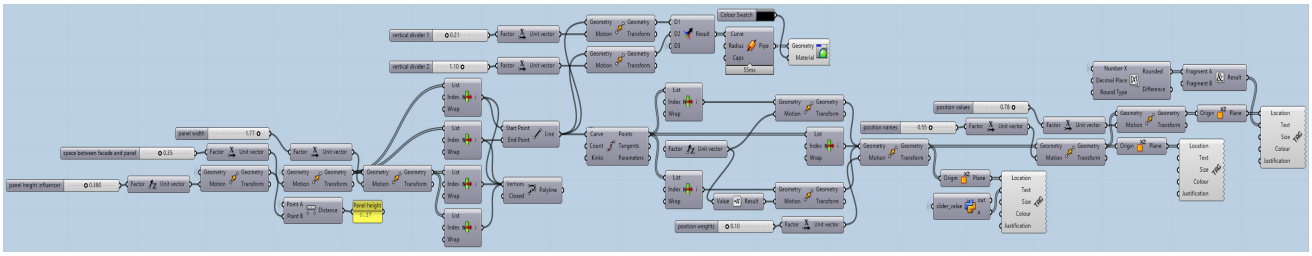


Figure G.6. Creating the top part of the three-dimensional information box, with the weights for each performance metric, their names, and their absolute performance values.

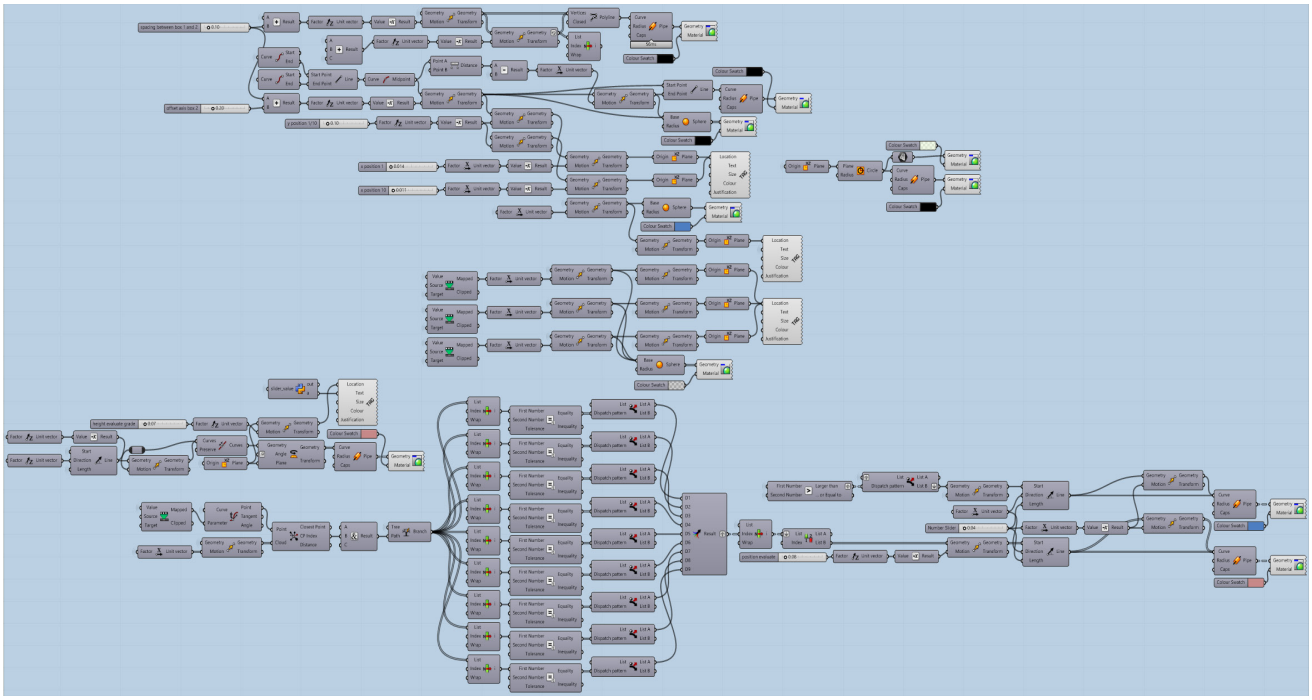


Figure G.7. Creating the middle part of the information box, with the performance score from 1 to 10 using remap numbers on the original domain of the performance values.

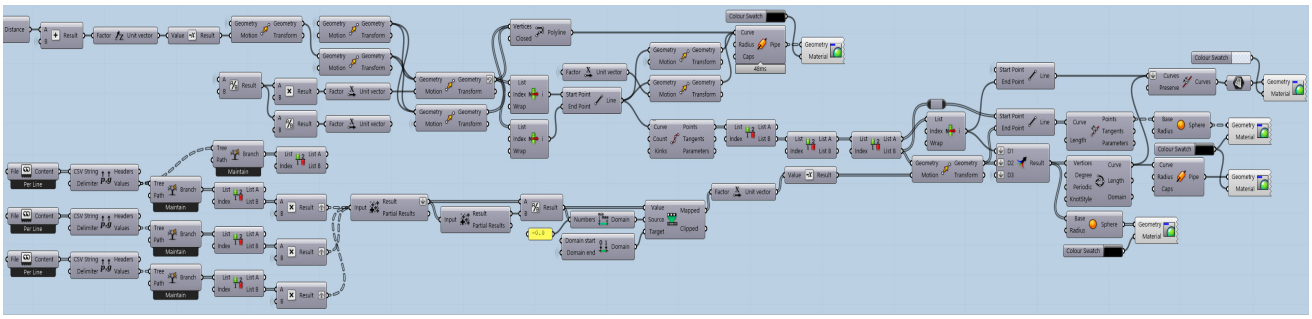


Figure G.8. Creating the last part of the information box, with the design variables' influence percentages using an interpolated curve which is filled with ruled surface.

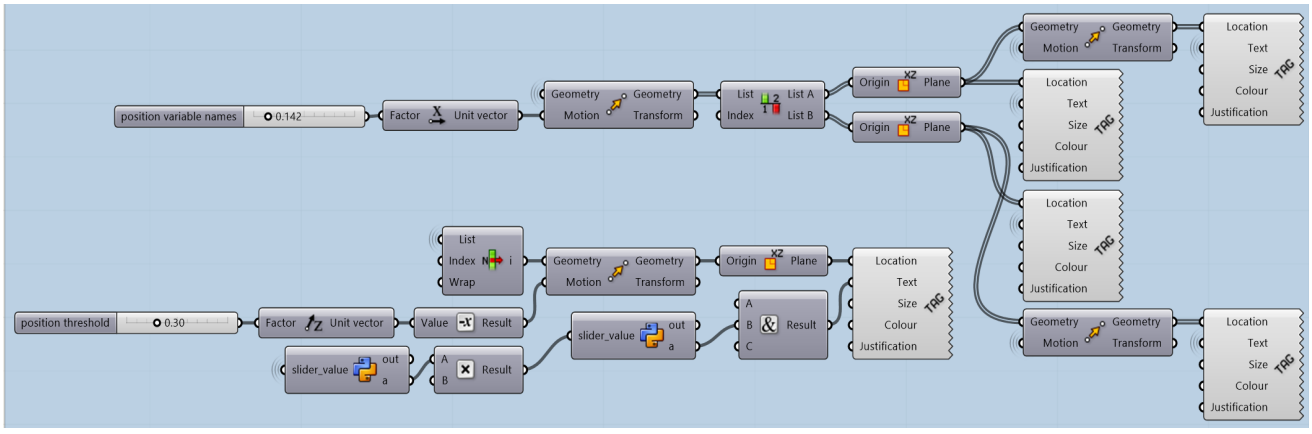
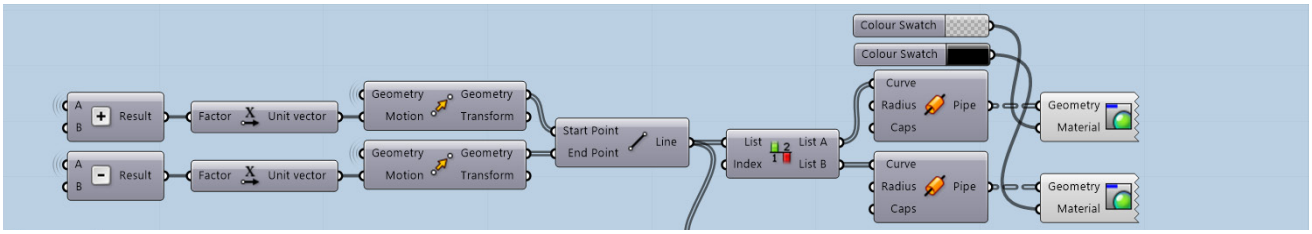


Figure G.9. Creating the design variable names and their absolute values using the text tag 3D component, and the influence threshold percentage below their names.



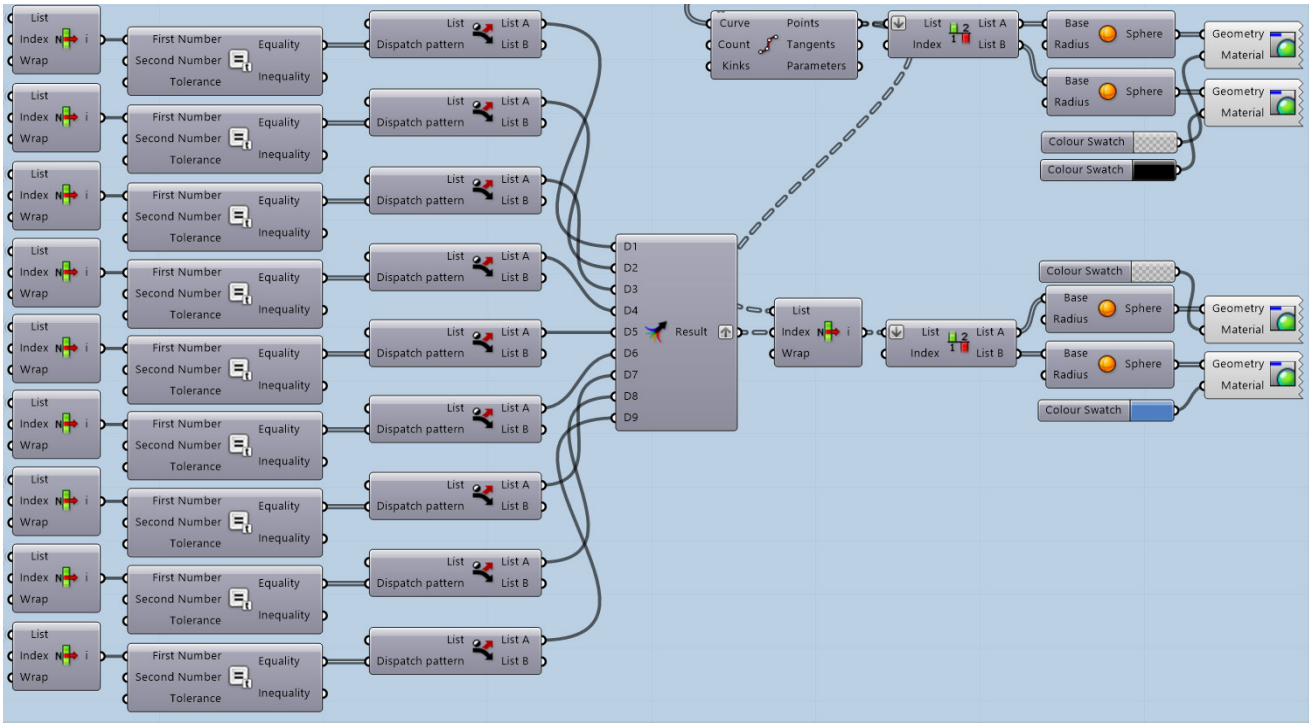


Figure G.10. Creating the design variables value range lines with points indicating the number of values in each range, and points indicating the current facade's setting.

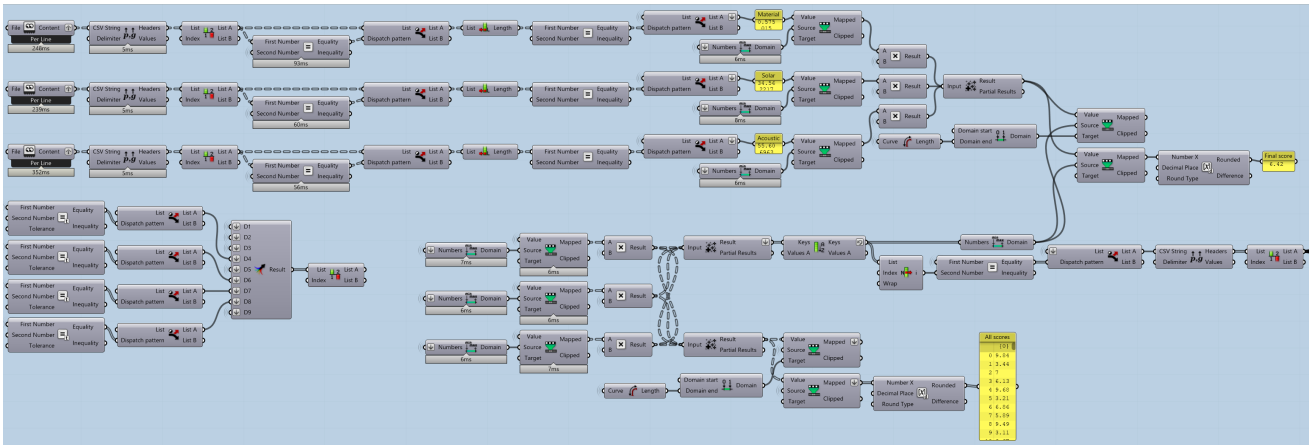


Figure G.11. Calculating the weighted performance score of the facade design, using the read csv file, remap numbers, mass addition, and multiplication components.

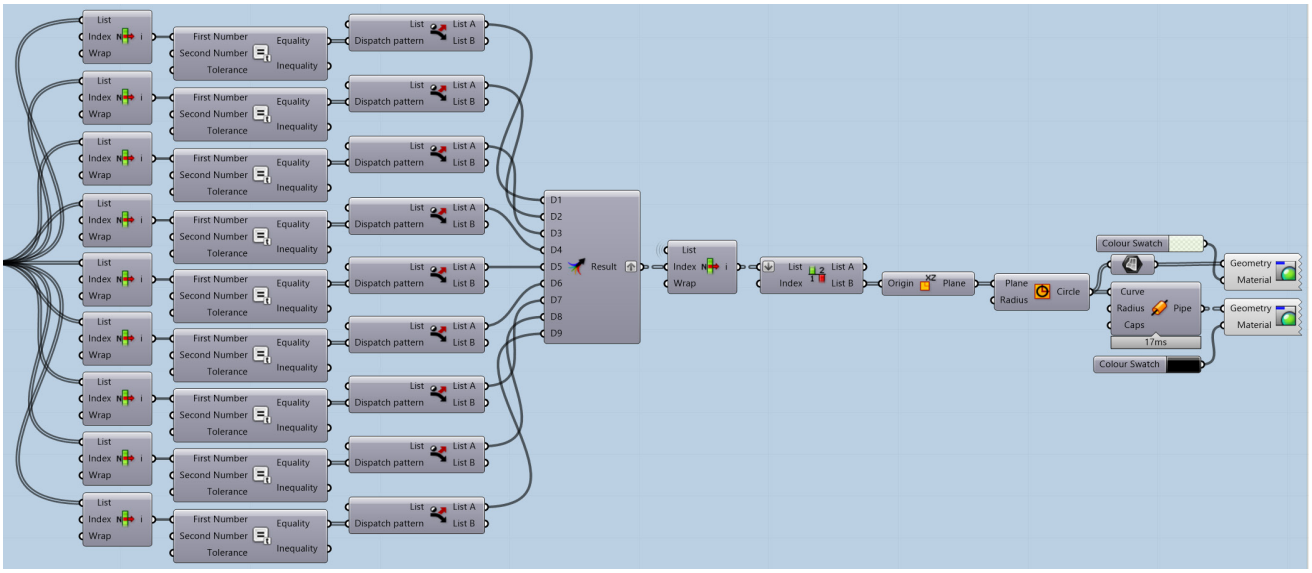


Figure G.12. Using the best-performing facade design to extract its design variable settings, and using these values to highlight these optimal settings in the information box.

H

Questionnaire

Dear participant, please take 5 to 10 minutes to complete this questionnaire. Your feedback will remain fully anonymous. Please focus on both frameworks as they were applied specifically for facade design.

If you have any questions left, please feel free to contact corresponding researcher Djani Cerneus at D.R.Cerneus@student.tudelft.nl or responsible researcher Alessandra Luna Navarro at A.LunaNavarro@tudelft.nl.

Thank you for your participation!

A. Participant Background

1. How familiar are you with AI-driven tools or Machine Learning?

☐ Not familiar ☐ Slightly familiar ☐ Neutral ☐ Familiar ☐ Very familiar

2. How familiar are you with Rhino 3D or Grasshopper?

☐ Not familiar ☐ Slightly familiar ☐ Neutral ☐ Familiar ☐ Very familiar

3. How experienced are you with facade design?

☐ No experience ☐ Limited experience ☐ Neutral ☐ Experienced ☐ Very experienced

4. How comfortable are you with using novel design frameworks?

☐ Uncomfortable ☐ Slightly comfortable ☐ Neutral ☐ Comfortable ☐ Very comfortable

B. Orientation Phase

5. How would you assess the overall impression of the orientation phase interface?

☐ Very poor ☐ Poor ☐ Neutral ☐ Good ☐ Excellent

6. How effective was the orientation phase in enabling fast design orientation?

☐ Very ineffective ☐ Ineffective ☐ Neutral ☐ Effective ☐ Very effective

7. How intuitive was design exploration during the orientation phase?

☐ Not intuitive ☐ Slightly intuitive ☐ Neutral ☐ Intuitive ☐ Very intuitive

8. How useful were the tools provided during the orientation phase (e.g. performance metric weights, performance threshold plane, and performance score line) in narrowing down design options?

☐ Not useful ☐ Slightly useful ☐ Neutral ☐ Useful ☐ Very useful

9. Did the orientation phase help you confidently identify the best design option?

☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly agree

10. How satisfied are you with the final design chosen during the orientation phase?

☐ Very dissatisfied ☐ Dissatisfied ☐ Neutral ☐ Satisfied ☐ Very satisfied

C. Fine-Tuning Phase

11. How would you assess the overall impression of the fine-tuning phase interface?

☐ Very poor ☐ Poor ☐ Neutral ☐ Good ☐ Excellent

12. How effective was the fine-tuning phase in refining your chosen design?

☐ Very ineffective ☐ Ineffective ☐ Neutral ☐ Effective ☐ Very effective

13. How intuitive was design exploration during the fine-tuning phase?

☐ Not intuitive ☐ Slightly intuitive ☐ Neutral ☐ Intuitive ☐ Very intuitive

14. How useful was the information panel for adjusting variables stratigically toward optimal performance?

☐ Not useful ☐ Slightly useful ☐ Neutral ☐ Useful ☐ Very useful

15. Did the fine-tuning phase help you confidently identify the best design option?

☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly agree

16. How satisfied are you with the final design chosen during the fine-tuning phase?

☐ Very dissatisfied ☐ Dissatisfied ☐ Neutral ☐ Satisfied ☐ Very satisfied

D. Overall Design Exploration

17. Did the overall framework provide a good balance between qualitative and quantitative performance?

☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly agree

18. How effective was the overall framework in exploring the design space?

☐ Very ineffective ☐ Ineffective ☐ Neutral ☐ Effective ☐ Very effective

E. Comparison With Traditional Methods

19. How would you compare the orientation phase of the novel framework with the traditional between-cluster exploration phase in terms of visualisation, speed, usability, and effectiveness in decision-making?

☐ Much worse ☐ Worse ☐ Neutral ☐ Better ☐ Much better

20. How would you compare the fine-tuning phase of the novel framework with the traditional inside-cluster exploration phase in terms of visualisation, speed, usability, and effectiveness in decision-making?

☐ Much worse ☐ Worse ☐ Neutral ☐ Better ☐ Much better

21. Which framework made you feel more confident in the final design choices?

☐ Traditional framework ☐ Neutral ☐ Novel framework

22. Which framework did you find more effective for facade design exploration overall?

☐ Traditional framework ☐ Neutral ☐ Novel framework