

MASTERARBEIT

zur Erlangung des akademischen Grades

„Master of Engineering“

im Fach Bauingenieurwesen

Untersuchungen zur Verwendung von künstlicher Intelligenz im Rahmen von statischen Berechnun- gen

**Investigations on the use of artificial intelligence in the context
of structural analysis**

Name

Yazeed Hazim Qasho

Matrikelnummer:	981345
Fachbereich:	Bauen + Leben
Fachrichtung:	Bauingenieurwesen
Abgabetermin:	13.05.2025
Aufgabensteller und Betreuer:	Prof. Dr. Yannick Broschart

Yazeed Hazim Qasho

Untersuchungen zur Verwendung von künstlicher Intelligenz im Rahmen von statischen Berechnungen

Inhaltsverzeichnis

1	Einleitung	3
1.1	Motivation der Arbeit	3
1.2	Zielsetzung der Arbeit	3
1.3	Aufbau der Arbeit	4
2	Grundlagen der Künstlichen Intelligenz	5
2.1	Allgemeine Funktionsweise von Künstlicher Intelligenz	5
2.3	Maschinelles Lernen und neuronale Netze	6
2.3.1	Grundlagen des Maschinellen Lernens	6
2.3.2	Aufbau und Funktionsweise neuronaler Netze	6
2.3.3	Einsatz bei statischen Berechnungen	7
2.4	Relevante Fachbegriffe	8
3	Künstliche Intelligenz in der Tragwerksanalyse	9
3.1	Überblick über bestehende Softwarelösungen	9
3.2	Rolle von KI in Finite-Elemente-Berechnungen	9
4	Codierungsstruktur und zentrale Begriffe	10
4.1	Prompt Verarbeitung durch das KI-System	10
4.2	Codegenerierung durch GPT	11
4.3	Aufbau der RFEM-API	12
4.4	Webservices und Datenkommunikation	13
4.5	Zentrale Befehle im Projekt	14
5	Struktur und Funktionsbereiche der RFEM-Python-Bibliothek	15
5.1	Übersicht der Module in Tabellenform mit Beschreibung	15
5.2	Modellzugriff und Berechnung in Python	16
5.3	Verwendung von Enums in RFEM	17
5.4	BasicObjects – Struktur und Aufruf der Objekte	18
	• Aufruf der Knoten	18
	• Aufruf der Linie	18
	• Aufruf des Materials	19
	• Aufruf des Querschnittes (Crosssection)	19
	• Aufruf eines Stabes (Member)	19
	• Aufruf einer Fläche (Surface)	20
5.5	Loads – Struktur und Aufruf der Lastobjekte	20

5.5.1	Aufruf einer Linienlast	21
5.5.2	Aufruf einer Flächenlast	21
5.5.3	Aufruf einer Stablast (MemberLoad)	21
5.5.4	Aufruf einer LinienSet-Last	21
5.5.5	Aufruf einer FlächenSet-Last	22
5.5.6	Aufruf einer StabSet-Last	22
5.6	Aufruf die LoadCasesAndCombinations – Lastfälle und Kombinationen in RFEM	22
5.6.1	Action – Definition von Lastarten	22
5.6.2	Aufruf ActionCombination – Verknüpfung von Lastarten	23
5.6.3	Aufruf CombinationWizard – Automatische Kombinationsgenerierung	23
5.6.4	Aufruf DesignSituation – Definition der Bemessungssituation	24
5.6.5	Aufruf ModalAnalysisSettings – Einstellungen zur Eigenwertanalyse	24
5.6.6	SpectralAnalysisSettings – Antwortspektrum für Erdbebenaufruf einer StabSet-Last	24
5.6.7	Aufruf stabilityAnalysisSettings	25
5.6.8	staticAnalysisSettings – Statische Solver konfiguration Aufruf einer StabSet-Last	25
5.6.9	Aufruf windSimulationAnalysisSetting – CFD-Windsimulation	25
5.7	Aufruf von TypesFor in RFEM	26
5.7.1	Aufruf Typdefinitionen für Linien (TypesForLines)	26
5.7.2	Aufruf Typdefinitionen für Knoten (TypesForNodes)	27
5.7.3	Aufruf Aufruf Typdefinitionen für Knoten (TypesForNodes)	28
5.8	Befehle Zusammenfassung	28
6	Notwendige Unterlagen zur Vorbereitung und Durchführung	29
6.1	Textdatei im .txt-Format	30
6.1.1	Introduction .txt-Format	30
6.1.2	Material.txt-Format	31
6.1.3	Geometry.txt-Format	31
6.1.4	Crosssection.txt-Format	33
6.1.5	Loads.txt-Format	33
6.1.6	Solver.txt-Format	34
6.1.7	Results.txt-Format	35
6.2	RFEM-Prompts im .py-Format	36
6.3	RFEM-Bot im .py-Format	38
6.4	Integration von Prompts, Templates und GPT in einem KI-gestützten Modellierungssystem	39

7	Erforderliche Dateien und Programme für die Systemvorbereitung und -ausführung	41
7.1	Erforderliche Programme	41
7.2	Erforderliche Dateien	42
8	Exemplarische Darstellung und methodisches Vorgehen	44
8.1	Herunterladen der RFEM-Python-Bibliothek	45
8.1.1	Einbindung der RFEM-Bibliothek in Visual Studio	45
8.2	Initialisierung des API-Schlüssels zur Nutzung der Programmierschnittstelle	46
8.3	Initiale Systemanpassung und – einrichtung	47
8.3.1	Speicherung des API-Schlüssels im System	48
8.3.2	Installation der Programmiersprache Python	48
8.3.3	Überprüfung der API-Schlüssel-Funktionalität in PowerShell	48
8.4	Erstellen und Bearbeiten von Skripten in Visual Studio Code	49
8.4.1	Entwicklung des Hauptskripts	49
8.4.2	Entwicklung des Skripts zur Dateneingabe	60
8.4.3	Entwicklung von Propmt	69
8.5	Durchführung und Berechnung der Aufgabenstellung	70
8.5.1	Beschreibung der Ausgangsparameter und Abmessungen	70
8.5.2	Abfrage der Parameter durch KI-generierten Python-Code	72
8.5.3	Start der Berechnung	82
8.5.4	Ergebnisse	83
8.5.5	Bewertung Der Ergebnisse	96
9	Typische Fehlermeldungen und deren Behebung	101
9.1	Fehlermeldungen	101
9.1.1	No surfaces generated – line loop is not closed	101
9.1.2	Member not assigned to any line	101
9.1.3	Invalid material or thickness ID	101
9.1.4	Load direction not recognized	102
9.1.5	API key not found or invalid	102
9.1.6	rfem_bot.py nicht gefunden oder falscher Pfad	102
8.6.7	RFEM API nicht aktiv	103
8.6.8	Skript nicht dauerhaft verfügbar	104
8.6.9	Number of mutations to keep not matching or invalid	104
8.6.10	Invalid or missing design situation in LoadCombination	105
8.6.11	WebFault: Globaler Parameter – Wert außerhalb des erlaubten Bereichs	105
8.6.12	SyntaxError: invalid syntax – ungültiger Import	106
9.2	Hinweis zur Nutzung von KI	107

10	Vorteile und Nachteile der KI-gestützten statischen Berechnung	109
10.1	Vorteile:	109
10.2	Nachteile:	110
11	Zusammenfassung und Ausblick	111
11.1	Zusammenfassung:	111
11.2	Ausblick:	111
11.3	Persönliche Meinung	112

Zusammenfassung

In der vorliegenden Masterarbeit wird ein innovativer, KI-gestützter Ansatz zur modellbasierten Tragwerksanalyse vorgestellt. Ziel ist es, den klassischen Prozess der statischen Modellierung und Berechnung durch intelligente Automatisierung zu optimieren und zukunftsorientiert weiterzuentwickeln. Der Fokus liegt dabei auf der automatisierten Erstellung, Berechnung und strukturierten Auswertung von statischen Modellen mithilfe der Finite-Elemente-Software Dlubal RFEM in Kombination mit einem eigens entwickelten Eingabemodul auf Basis künstlicher Intelligenz.

Die Besonderheit dieses Systems liegt in der Verwendung natürlicher Sprache als Ausgangspunkt für die Modellgenerierung. Nutzerinnen und Nutzer formulieren statische Problemstellungen in frei formulierter Alltagssprache. Ein auf GPT basierendes KI-Modell interpretiert diese Eingaben, extrahiert relevante technische Informationen und erzeugt automatisch ein ausführbares Python-Skript. Dieses Skript kommuniziert über die Webservices-Schnittstelle mit RFEM, erstellt das zugehörige Strukturmodell und initiiert die statische Berechnung. Dabei erkennt das System automatisch, ob beispielsweise Träger als Linienstäbe oder Flächen als Platten modelliert werden müssen, und ergänzt den notwendigen Code ohne manuelle Nachbearbeitung. Somit wird eine vollständig automatisierte Übersetzung von der Spracheingabe bis hin zur berechneten Struktur realisiert.

Ein weiterer zentraler Bestandteil dieser Arbeit ist die Entwicklung zusätzlicher Skripte zur Bewertung der ökologischen Auswirkungen der Tragwerksplanung. Insbesondere wurde ein Modul zur automatisierten Berechnung von CO₂-Emissionen integriert. Dabei werden verschiedene Parameter berücksichtigt, etwa die verwendeten Materialien, Bauteildicken, Flächeninhalte und die Bewehrung. Die daraus resultierenden Emissionen werden in strukturierter Form ausgewertet und ermöglichen eine fundierte ökologische Bewertung der entworfenen Tragwerke ein Aspekt, der in der modernen Baupraxis zunehmend an Bedeutung gewinnt.

Die Masterarbeit dokumentiert sowohl die allgemeinen Anwendungsmöglichkeiten des entwickelten Systems in der Praxis der Tragwerksplanung als auch die konkreten technischen Schritte, die zur Umsetzung geführt haben. Detailliert werden dabei die interne Systemarchitektur, die Verarbeitung natürlicher Sprache, die dynamische Auswahl von Codevorlagen, die API-Integration mit RFEM sowie die automatisierte Ergebnisanalyse erläutert. Zusätzlich wird der gesamte Entwicklungsprozess nachvollziehbar beschrieben von der Idee über die Programmierung bis zur Validierung am praktischen Beispiel.

Durch die Verbindung von KI, Ingenieursoftware und nachhaltiger Bewertung leistet diese Arbeit einen zukunftsweisenden Beitrag zur Digitalisierung und Automatisierung im Bauwesen. Sie eröffnet neue Perspektiven für interaktive, benutzerfreundliche Planungstools, die komplexe technische Prozesse vereinfachen und gleichzeitig ökologische Faktoren berücksichtigen. Die vorgestellte Lösung stellt damit nicht nur ein Werkzeug zur Effizienzsteigerung dar, sondern auch einen Impuls für eine nachhaltigere, intelligentere Baupraxis.

Abstract

This thesis presents an innovative, AI-driven approach to model-based structural analysis. The primary objective is to enhance and modernize the traditional process of structural modeling and analysis through intelligent automation, integrating artificial intelligence technologies with advanced engineering software to improve efficiency, accuracy, and environmental awareness in structural planning.

The core of this approach lies in the automated creation, calculation, and evaluation of structural models using the finite element software Dlubal RFEM, in combination with a custom-developed input module based on artificial intelligence. A key feature of the system is its use of natural language as the starting point for model generation. Users describe structural problems and design requirements in plain, everyday language. These inputs are then interpreted by a GPT-based AI model, which extracts relevant engineering information and automatically generates executable Python code.

This code communicates directly with RFEM via its Webservices API to create the corresponding structural model and perform the necessary calculations. One of the system's most powerful capabilities is its ability to automatically identify and define structural elements, such as distinguishing between beams that should be modeled as line members and surfaces that should be treated as slabs or plates. The system generates and completes the required code without any manual intervention, streamlining the entire modeling process.

In addition to automated model creation and analysis, this research also addresses the growing importance of sustainability in civil engineering. A dedicated script was developed to evaluate the environmental impact of structural models by calculating CO₂ emissions based on several parameters, including material selection, member thickness, surface areas, and reinforcement content. This automated CO₂ analysis enables a transparent and data-driven assessment of structural designs from both technical and ecological perspectives.

The thesis not only explores the general potential applications of this AI-based system within engineering practice but also provides a detailed technical explanation of the development process behind it. This includes the internal system architecture, the natural language processing workflow, the dynamic selection of code templates, the API integration with RFEM, and the automated analysis and reporting of structural results and emissions. Furthermore, the complete development journey is documented—from conceptualization and algorithm design to implementation and validation through practical case studies.

By combining artificial intelligence, structural engineering software, and environmental evaluation, this work contributes to the future of digitalized and automated engineering workflows. It demonstrates how intelligent tools can support engineers in simplifying complex tasks, improving decision-making, and promoting sustainable construction practices. The system presented here not only enhances technical efficiency but also lays the foundation for the next generation of user-friendly, interactive design tools in structural engineering.

1 Einleitung

1.1 Motivation der Arbeit

Künstliche Intelligenz spielt bereits heute eine bedeutende Rolle im Alltag und wird zukünftig auch im Bauwesen nicht mehr wegzudenken sein. Diese Entwicklung war ein zentraler Grund für die Wahl des Themas dieser Arbeit. Der Einsatz von KI in Verbindung mit ingenieurtechnischer Software wie Dlubal RFEM eröffnet neue Möglichkeiten zur Optimierung von Planungsprozessen.

Bereits im Vorfeld konnte ich praktische Erfahrungen mit RFEM und der Anwendung von ChatGPT sammeln, was mein Interesse an der Schnittstelle zwischen klassischen Ingenieurmethoden und modernen KI-Technologien zusätzlich gestärkt hat. Ziel war es, ein konkretes Problem – die umständliche und fehleranfällige manuelle Eingabe statischer Modelle – zu lösen. Dieses Problem konnte mithilfe eines automatisierten, KI-gestützten Systems erfolgreich adressiert werden.

Langfristig bietet die entwickelte Lösung Potenzial für eine praxisnahe Weiterentwicklung und könnte perspektivisch zu einem intuitiven Planungstool für Ingenieurinnen und Ingenieure ausgebaut werden.

1.2 Zielsetzung der Arbeit

Ziel dieser Masterarbeit ist die Konzeption, Entwicklung und exemplarische Anwendung eines KI-gestützten Assistenzsystems zur automatisierten Modellierung und Berechnung von Tragwerksmodellen in Dlubal RFEM. Im Mittelpunkt steht dabei die Idee, komplexe Modellierungsprozesse durch eine intuitive, textbasierte Benutzerschnittstelle zu vereinfachen. Anstelle der herkömmlichen manuellen Eingabe über die grafische Benutzeroberfläche soll es möglich sein, das gewünschte Tragwerk durch eine natürliche Spracheingabe zu definieren. Die künstliche Intelligenz interpretiert diese Eingaben, generiert daraus automatisch den entsprechenden Python-Code für die RFEM-API und führt sowohl die Modellierung als auch die anschließende Berechnung direkt aus.

Die Arbeit verfolgt das Ziel, die Potenziale eines solchen automatisierten Workflows aufzuzeigen und dabei insbesondere die Aspekte Benutzerfreundlichkeit, Effizienzsteigerung und Reduktion von Eingabebefehlern zu bewerten. Die technische Umsetzung erfolgt mithilfe einer Streamlit-basierten Weboberfläche, die als Brücke zwischen dem Nutzer, dem Sprachmodell und der RFEM-Umgebung dient. Abschließend wird das entwickelte System anhand eines prototypischen Anwendungsbeispiels getestet und hinsichtlich seiner Praxistauglichkeit, Leistungsfähigkeit und Grenzen bewertet.

1.3 Aufbau der Arbeit

Die Masterarbeit besteht aus elf Kapiteln, die jeweils ein eigenes Ziel verfolgen und einen spezifischen Beitrag zum Gesamtverständnis der entwickelten Methode leisten. Kapitel 1 liefert eine Einführung in das Thema. Es beschreibt die Motivation, den fachlichen Hintergrund und den strukturellen Aufbau der Arbeit. In Kapitel 2 wird der Stand der Technik analysiert. Dabei werden bestehende Systeme zur automatisierten Modellierung und Analyse vorgestellt sowie die Rolle künstlicher Intelligenz im Bauwesen und relevante Forschungsergebnisse eingeordnet.

Kapitel 3 befasst sich mit dem Einsatz künstlicher Intelligenz im Ingenieurwesen. Insbesondere wird erläutert, wie GPT-basierte Systeme den Prozess statischer Berechnungen unterstützen und automatisieren können. Kapitel 4 beschreibt die Struktur der Programmierung. Es wird aufgezeigt, welche Skripte verwendet oder entwickelt wurden, welche Funktionen sie erfüllen und wie typische Abläufe im KI-gestützten Modellierungsprozess ablaufen.

Kapitel 5 behandelt die allgemeinen Funktionen des Systems. Es erläutert die verfügbaren Funktionen, deren Bedeutung und Anwendung sowie ihr Zusammenspiel im Gesamtsystem. Eine zusammenfassende Übersicht aller Funktionen schließt das Kapitel ab. In Kapitel 6 werden die in RFEM vorhandenen Vorlagen und Bibliothekselemente beschrieben, die für die automatisierte Modellierung genutzt werden können – darunter Materialien, Querschnitte und Lastdefinitionen sowie deren Formatierung und Integration.

Kapitel 7 beschreibt die erforderlichen Programme und Dateien für den Betrieb des Systems. Dazu zählen unter anderem die Python-Umgebung, RFEM-Webservices, GPT-Zugänge sowie Konfigurations- und Template-Dateien. In Kapitel 8 beginnt der praktische Teil. Anhand eines einfachen Beispiels wird schrittweise erklärt, wie Skripte modelliert werden müssen. Anschließend wird ein realitätsnahes Beispiel analysiert, bei dem die GPT-Kommunikation im Fokus steht. Der Ablauf zwischen Nutzer und System wird detailliert dargestellt und die erzielten Ergebnisse bewertet.

Kapitel 9 dokumentiert typische Fehlermeldungen, die im Entwicklungsprozess aufgetreten sind, und bietet zu jeder eine Ursachenanalyse sowie funktionierende Lösungen. Kapitel 10 stellt die Vor- und Nachteile der entwickelten Methode gegenüber. Automatisierung, Effizienz und Benutzerfreundlichkeit werden dabei ebenso betrachtet wie bestehende Schwächen und Herausforderungen. Kapitel 11 fasst die zentralen Erkenntnisse der Arbeit zusammen, enthält eine persönliche Bewertung des entwickelten Ansatzes und gibt einen Ausblick auf mögliche Weiterentwicklungen.

2 Grundlagen der Künstlichen Intelligenz

2.1 Allgemeine Funktionsweise von Künstlicher Intelligenz

Künstliche Intelligenz (KI) ist ein interdisziplinäres Teilgebiet der Informatik, das sich mit der Automatisierung intelligenten Verhaltens befasst. Ziel ist es, Maschinen mit Fähigkeiten auszustatten, die beim Menschen als intelligent gelten – beispielsweise das Lernen aus Daten, das Erkennen von Mustern oder die Entscheidungsfindung auf Basis unsicherer Informationen. Zu den zentralen Technologien zählen wissensbasierte Systeme, maschinelles Lernen (ML) und künstliche neuronale Netze (KNN), wobei letztere insbesondere bei komplexen, nichtlinearen Problemstellungen Anwendung finden. [Babel, T. (2024)]

Ein typisches KI-System besteht aus mehreren Komponenten: der Datenerfassung (z. B. durch Sensoren oder Simulationen), der Datenverarbeitung und Mustererkennung sowie einer Entscheidungslogik, die auf den gelernten Informationen basiert. Besonders das maschinelle Lernen ermöglicht es Systemen, auf Grundlage vorhandener Daten Vorhersagen zu treffen oder Zusammenhänge zu erkennen, ohne explizit dafür programmiert worden zu sein. [Babel, T. (2024)]

Auch im Bauwesen gewinnt KI zunehmend an Bedeutung – insbesondere im Bereich der statischen Berechnungen. Hier kommen Verfahren wie künstliche neuronale Netze zum Einsatz, um Strukturverhalten zu modellieren oder bestehende Rechenprozesse zu beschleunigen. So kann z. B. bei der Modellierung nichtlinearer Tragstrukturen eine KI anstelle eines vollständigen physikalischen Modells verwendet werden, was eine deutliche Reduktion der Rechenzeiten bei gleichzeitig akzeptabler Genauigkeit erlaubt. Damit ergänzt die KI klassische numerische Methoden wie die Finite-Elemente-Methode (FEM) um datengetriebene Ansätze, die besonders bei großen Datenmengen oder in der Entwurfsphase effizient eingesetzt werden können. [Babel, T. (2024)]

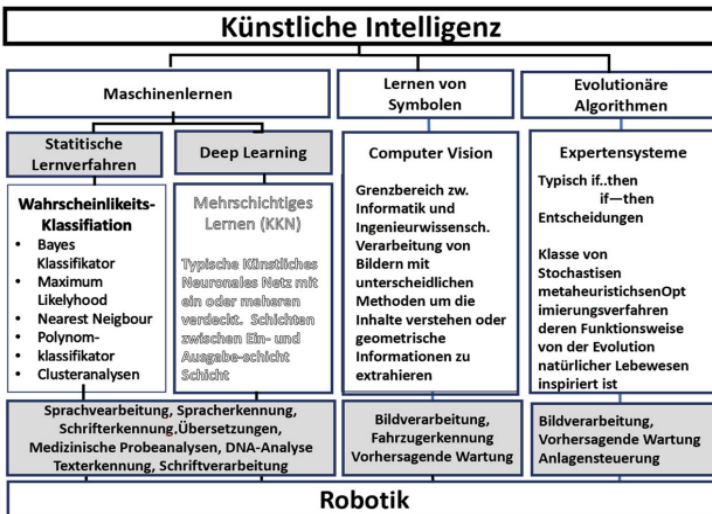


Abb.2.1: Struktur von KI [Babel, T. (2024)]

2.3 Maschinelles Lernen und neuronale Netze

2.3.1 Grundlagen des Maschinellen Lernens

Maschinelles Lernen (ML) ist ein Teilbereich der künstlichen Intelligenz, der es Computern ermöglicht, aus Daten zu lernen und darauf basierende Entscheidungen oder Vorhersagen zu treffen – ohne explizit programmiert zu sein. Grundlegend unterscheidet man zwischen überwachtem Lernen (z. B. für Regressions- oder Klassifikationsaufgaben), unüberwachtem Lernen (z. B. Clusteranalysen) und bestärkendem Lernen (z. B. Agentensysteme mit Belohnung).

Im Bauwesen wird ML eingesetzt, um große Datenmengen aus Simulationen, Messungen oder Sensoren auszuwerten und daraus z. B. Strukturverhalten vorherzusagen. [Babel, T. (2024), Koch, C. (2021)]

2.3.2 Aufbau und Funktionsweise neuronaler Netze

Neuronale Netze sind rechnergestützte Modelle, die vom biologischen Nervensystem inspiriert sind. Sie bestehen aus mehreren Schichten von künstlichen Neuronen: einer Eingabeschicht, einer oder mehreren versteckten Schichten (Hidden Layers) und einer Ausgabeschicht.

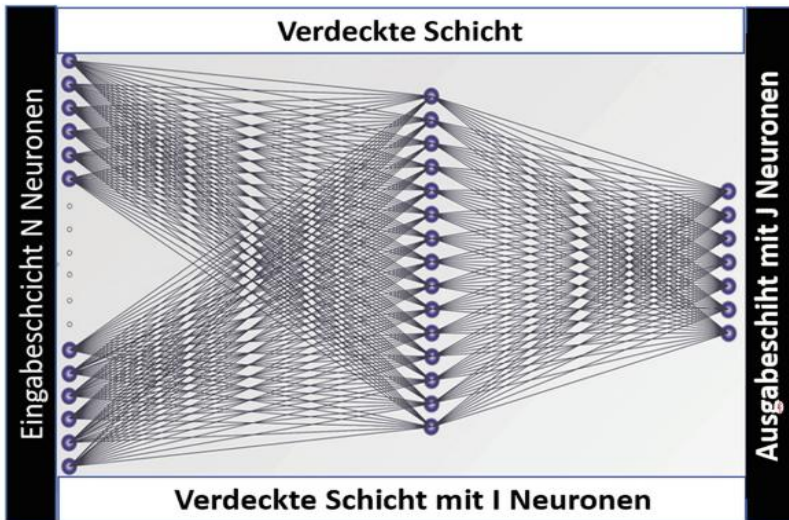


Abb.2.2: Neuronales Netzwerk mit N Eingangsneuronen, J Neuronen im verdeckten Layer und M Ausgangsneuronen [Babel, T. (2024)]

Jedes Neuron verarbeitet Eingaben mit Hilfe einer Aktivierungsfunktion (z. B. ReLU, Sigmoid) und leitet die Information an das nächste Neuron weiter.

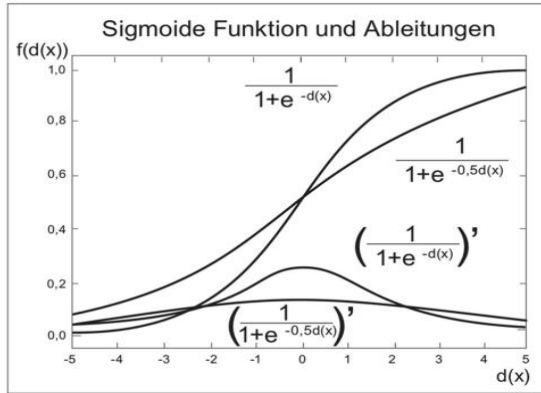


Abb.2.3: Schwellwertfunktion $d(x)$ – Sigmoide Funktion

Das Trai-

ning eines neuronalen Netzes erfolgt typischerweise mit dem Backpropagation-Verfahren und einem Optimierungsalgorithmus wie Stochastic Gradient Descent, um Fehler zu minimieren. [Babel, T. (2024)]

2.3.3 Einsatz bei statischen Berechnungen

Im Bauingenieurwesen, insbesondere im Bereich statischer Berechnungen und FEM-Simulationen, finden künstliche neuronale Netze zunehmend Anwendung. Sie werden beispielsweise eingesetzt, um komplexes Materialverhalten wie nichtlineare Spannungs-Dehnungs-Beziehungen näherungsweise zu modellieren. Darüber hinaus eignen sich neuronale Netze zur Vorhersage von FEM-Ergebnissen wie Spannungen, Verformungen oder Eigenfrequenzen auf Basis gegebener Eingangsdaten. Auch in der Optimierung von Tragwerksentwürfen können sie zur Effizienzsteigerung beitragen, indem sie Rechenprozesse deutlich beschleunigen. Zudem ermöglichen sie die Durchführung von Sensitivitätsanalysen oder parametrischen Studien mit reduziertem Rechenaufwand. Ein konkretes Beispiel ist der Einsatz eines trainierten neuronalen Netzes, das bei bekannter Last und Geometrie die maximal auftretenden Verformungen vorhersagen kann, ohne dass eine vollständige Finite-Elemente-Simulation durchgeführt werden muss. Darüber hinaus können solche Netze als Bestandteil intelligenter Assistenzsysteme in computergestützten Modellierungsprozessen dienen. So lassen sich etwa automatisch geeignete Modellparameter vorschlagen oder Plausibilitätsprüfungen durchführen, noch bevor eine detaillierte FE-Analyse gestartet wird. In Verbindung mit natürlichen Spracheingaben – etwa durch den Einsatz von GPT-basierten Modellen – entsteht dadurch ein vielversprechender Ansatz zur weitgehend automatisierten, KI-gestützten Tragwerksplanung. [Babel, T. (2024), Koch, C. (2021)]

2.4 Relevante Fachbegriffe

In diesem Kapitel werden zentrale Begriffe definiert, die für das Verständnis der weiteren Ausführungen zur Anwendung Künstlicher Intelligenz in der Tragwerksplanung wesentlich sind. Die Begriffe basieren auf Fachliteratur und orientieren sich an aktuellen Definitionen aus Wissenschaft und Praxis.

- Künstliche Intelligenz (KI)

KI ist ein Teilgebiet der Informatik, das sich mit der Automatisierung intelligenten Verhaltens befasst. [Babel 2024]

- Maschinelles Lernen (ML)

Maschinelles Lernen ist eine Methode, durch die ein System aus Daten lernt, ohne explizit programmiert zu sein. [Babel 2024, Koch 2021]

- Neuronales Netz (KNN oder ANN)

Neuronale Netze bestehen aus künstlichen Neuronen, die in Schichten organisiert sind. Sie sind in der Lage, komplexe Muster zu erkennen und Funktionen zu approximieren. [Babel 2024]

- Deep Learning

Deep Learning ist ein Teilbereich des maschinellen Lernens, bei dem tiefe neuronale Netze verwendet werden, um sehr komplexe Zusammenhänge zu modellieren. [Koch 2021]

- Finite-Elemente-Methode (FEM)

Die FEM ist ein numerisches Verfahren zur näherungsweise Lösung von Differentialgleichungen, die z. B. bei der Analyse von Spannungen und Verformungen in Tragwerken auftreten. [Koch 2021]

- Modellkalibrierung

Modellkalibrierung bezeichnet die Anpassung von Modellparametern an gemessene Daten, um ein realistisches Abbild des physikalischen Systems zu erhalten. [Babel 2024]

3 Künstliche Intelligenz in der Tragwerksanalyse

3.1 Überblick über bestehende Softwarelösungen

Im Bereich der Tragwerksanalyse existieren heute zahlreiche Softwarelösungen, die Methoden der klassischen Finite-Elemente-Methode (FEM) nutzen, jedoch zunehmend durch Elemente der Künstlichen Intelligenz (KI) ergänzt werden. Programme wie RFEM, SOFiSTiK oder SCIA Engineer bilden seit Jahren den Standard für die rechnergestützte Tragwerksplanung. Diese Werkzeuge ermöglichen die Durchführung komplexer Bauteilnachweise und statischer Berechnungen, sind jedoch in der Modellierung und Auswertung oftmals zeitintensiv und stark vom Erfahrungsschatz des Anwenders abhängig.

Moderne Entwicklungen setzen genau an diesem Punkt an. Die Firma SOFiSTiK etwa entwickelt im Rahmen ihres AI Labs KI-gestützte Funktionen, um Planungsprozesse zu automatisieren – etwa durch intelligente Vorschläge für Lagerbedingungen, Netzgenerierung oder Lastansätze. Auch Plattformen wie Autodesk BIM 360 integrieren KI zur Qualitätskontrolle von Modellierungsdaten, z. B. durch automatische Kollisionsprüfung und Regelüberprüfung. In der Forschung werden Python-Frameworks wie TensorFlow oder PyTorch eingesetzt, um KI-Modelle zu trainieren, die FEM-Ergebnisse (z. B. Verformungen, Spannungen oder Stabilitätsreserven) annähernd berechnen können. Diese Lösungen basieren auf zuvor simulierten Datensätzen, aus denen neuronale Netze lernen. Derartige Tools befinden sich teilweise noch im Forschungsstadium, zeigen jedoch bereits heute großes Potenzial in Bezug auf Effizienz und Automatisierung. (Babel 2024, Koch 2021)

3.2 Rolle von KI in Finite-Elemente-Berechnungen

Die Finite-Elemente-Methode ist das zentrale Verfahren in der numerischen Strukturanalyse. Aufgrund der gestiegenen Komplexität moderner Bauwerke und dem wachsenden Bedarf an Variantenstudien, Optimierungen und nichtlinearen Berechnungen, stoßen klassische FEM-Ansätze jedoch zunehmend an ihre Grenzen, insbesondere hinsichtlich Rechenzeit und Automatisierung. Hier setzt Künstliche Intelligenz an.

KI-Methoden, insbesondere künstliche neuronale Netze (KNN), können verwendet werden, um strukturmechanisches Verhalten zu erlernen und vorherzusagen. Dies geschieht, indem ein KI-Modell auf eine Vielzahl von zuvor berechneten FEM-Datensätzen trainiert wird. Anschließend kann das Netz auf neue Eingabedaten (z. B. Lasten oder Geometrien) reagieren und relevante Ausgabegrößen wie maximale Verformung, Spannung oder Eigenfrequenz approximieren. Diese Herangehensweise erlaubt eine erhebliche Reduktion der Rechenzeit, insbesondere bei wiederkehrenden Analysen oder parametrischen Studien.

In Kombination mit klassischen FEM-Systemen kann die KI auch zur Modellkalibrierung oder zur automatisierten Generierung von Eingabedaten verwendet werden. Damit eröffnet sich ein neues Paradigma der hybriden Simulation, bei dem KI und physikbasierte Modelle eng zusammenarbeiten.

4 Codierungsstruktur und zentrale Begriffe

4.1 Prompt Verarbeitung durch das KI-System

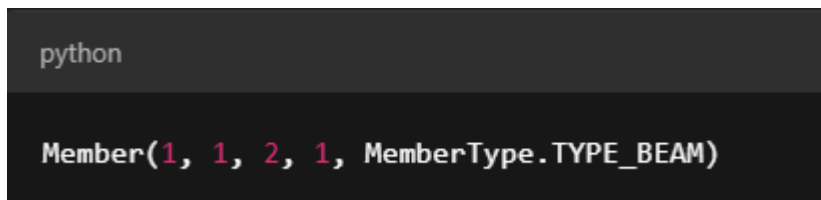
Ein zentraler Bestandteil des entwickelten Systems ist die Verwendung natürlicher Spracheingaben, sogenannter Prompts. Ein Prompt ist eine textbasierte Anweisung, die dem KI-Modell übergeben wird, um eine spezifische Reaktion – in diesem Fall einen entsprechenden Python-Code zur Modellierung in RFEM – zu erzeugen.

Im Rahmen dieses Projekts wurden Prompts wie:

„Erstelle einen Träger aus Beton mit einer Länge von 5 m und fester Einspannung an beiden Enden.“

verwendet, um automatisiert entsprechende Befehle zu generieren. Das KI-Modell analysiert die semantischen Bestandteile der Eingabe – etwa „Träger“, „Beton“, „5 m“, „Einspannung“ – und übersetzt diese in strukturierte Python-Befehle.

Ein Beispiel für die automatisch erzeugte Ausgabe wäre:



```
python
Member(1, 1, 2, 1, MemberType.TYPE_BEAM)
```

Abb.4.1: Erzeugte Ausgabe für einen Träger

Hier erkennt das Modell, dass ein linearer Träger zwischen zwei Knoten (1 und 2) benötigt wird, und erzeugt den notwendigen Code-Block, ohne dass der Nutzer direkte Programmierkenntnisse benötigt.

Die Generierung basiert auf einem sogenannten Template-Matching-Verfahren in Kombination mit einem GPT-basierten Sprachmodell. Template Matching bezeichnet eine Technik, bei der das System vordefinierte Textmuster (Templates) mit der Nutzereingabe vergleicht. Wird ein bekanntes Muster erkannt – z. B. „Träger mit x m Länge“ –, wird automatisch ein passendes Code-Template ausgewählt und mit den konkreten Werten (z. B. Länge = 5 m) befüllt.

„Füge einen 5 Meter langen Betonbalken hinzu.“

korrekt interpretieren und technisch umsetzen.

Die erzeugten Prompts werden im nächsten Schritt weiterverarbeitet und über eine API-Schnittstelle an die RFEM-Umgebung übergeben.

4.2 Codegenerierung durch GPT

Nach der semantischen Analyse des eingegebenen Prompts erfolgt die automatisierte Erzeugung von Python-Code durch das KI-Modell. Dabei handelt es sich um ein Sprachmodell auf Basis der GPT-Architektur (Generative Pre-trained Transformer), das auf technische Begriffe, Strukturvorgaben und Syntaxmuster trainiert wurde.

Ziel dieses Prozesses ist es, aus dem Prompt eine strukturierte Folge von Codebefehlen abzuleiten, die direkt in einer Python-Schnittstelle mit RFEM ausgeführt werden können. Dabei erkennt das Modell Schlüsselwörter wie „Träger“, „Knoten“, „Beton“, „Einspannung“, „Länge“, „Last“ und ordnet diesen Begriffen entsprechende Funktionen oder Objekte aus der RFEM-API zu.

Ein einfaches Beispiel:

Prompt: „Erstelle einen 5 m langen Träger mit beidseitiger Einspannung.“

wird von GPT analysiert und liefert etwa folgenden Python-Code:

```
python
Node(1, 0, 0, 0)
Node(2, 5, 0, 0)
Member(1, 1, 2, 1, MemberType.TYPE_BEAM)
```

Abb.4.2: Analyse eines Trägers durch ChatGPT – Umwandlung in Node- und Member-Elemente

In diesem Beispiel erkennt das Modell, dass zwei Knoten im Abstand von 5 m angelegt werden sollen. Der Befehl Member(...) erstellt anschließend einen linearen Träger (beam) zwischen den beiden Knoten.

Komplexere Prompts wie:

„Füge eine Streckenlast von 10 kN/m in z-Richtung auf den Balken hinzu.“

führen zur Erzeugung zusätzlicher Befehle wie:

```
python
MemberLoad(1, 1, LoadDirection.LOAD_DIRECTION_GLOBAL_Z, 10000)
```

Abb.4.3: Analyse einer Last mit Richtung durch ChatGPT

Die Fähigkeit zur Codegenerierung basiert auf vortrainierten Beispielen und strukturierten Regeln, welche durch eine Kombination aus KI-gesteuerter Textverarbeitung und Template-Systemen ergänzt werden. Auf diese Weise kann das System nicht nur einfache Eingaben verarbeiten, sondern auch komplexe Modellanweisungen in ausführbaren Python-Code umwandeln.

4.3 Aufbau der RFEM-API

Die RFEM-API von Dlubal stellt eine umfangreiche Schnittstelle zur Verfügung, mit der sich Modelle und Berechnungen programmgesteuert erstellen, bearbeiten und auswerten lassen. Sie basiert auf modernen Webservice-Technologien und kann über verschiedene Programmiersprachen, insbesondere Python und C#, angesprochen werden.

Im Kern ermöglicht die API den direkten Zugriff auf das RFEM-Modell – vergleichbar mit der Bedienung über die Benutzeroberfläche, jedoch automatisiert über Code. Dadurch lassen sich insbesondere wiederkehrende Modellierungsaufgaben, Variantenstudien oder komplexe Generierungen effizient umsetzen.

Tab.4.1: Bedeutung der RFEM-Objekte

Objekt	Bedeutung
Node	Ein geometrischer Punkt im Raum, an dem z. B. Linien beginnen oder Lasten angreifen.
Line	Verbindung zwischen zwei oder mehr Knoten – meist die Grundlage für Stäbe.
Member	Ein stabförmiges Tragelement, das entlang einer Linie definiert wird.
Material	Gibt die physikalischen Eigenschaften eines Elements an (z. B. Beton C25/30).
CrossSection	Beschreibt den geometrischen Querschnitt eines Stabes (z. B. IPE 300).
LoadCase	Stellt eine bestimmte Belastungssituation dar.
MemberLoad	Eine Belastung, die auf einen Stab wirkt (z. B. Streckenlast).

4.4 Webservices und Datenkommunikation

Die Kommunikation zwischen dem Python-Skript und RFEM erfolgt über sogenannte Webservices. Ein Webservice ist eine standardisierte Schnittstelle, über die Programme über das Internet oder lokale Netzwerke miteinander kommunizieren können unabhängig von Programmiersprache oder Plattform.

In diesem Projekt wird die RFEM Webservice-API verwendet, die auf dem SOAP-Standard basiert und über HTTP-Anfragen arbeitet. Dabei wird zwischen Request und Response unterschieden:

Ein Request ist die Anforderung des Python-Clients an RFEM – z. B. „Erstelle ein neues Member.“

Die Response ist die Antwort von RFEM – z. B. eine Bestätigung, dass das Element erstellt wurde, oder ein Fehlercode.

Die dabei übertragenen Daten sind in der Regel im JSON-Format strukturiert. JSON (JavaScript Object Notation) ist ein leichtgewichtiges, textbasiertes Datenformat, das Schlüssel-Wert-Paare enthält. Ein Beispiel für ein JSON-Objekt zur Definition eines Knotens könnte wie folgt aussehen:

```
json
{
  "no": 1,
  "x": 0.0,
  "y": 0.0,
  "z": 0.0
}
```

Abb.4.4: JavaScript nach der Kommunikation zwischen RFEM und dem Python-Skript

Dieser Eintrag beschreibt einen Knoten mit der Nummer 1 am Ursprung des Koordinatensystems.

Die Verbindung zwischen dem Python-Skript und RFEM erfolgt über eine lokale Instanz von RFEM, die als Server agiert. Das Skript stellt über einen Client (z. B. `Model.clientModel.service`) Anfragen an diesen Server. Dieser Vorgang läuft im Hintergrund automatisch ab, sobald das Python-Modul `rfem` korrekt initialisiert und die Verbindung geöffnet wurde.

Dank dieser Struktur ist es möglich, Modelle vollständig programmgesteuert zu definieren, zu ändern, zu berechnen und auszulesen – ohne die grafische Benutzeroberfläche zu verwenden.

4.5 Zentrale Befehle im Projekt

Zur Steuerung der Modellierungsprozesse in RFEM über Python kommen bestimmte zentrale Befehle zum Einsatz, die den Zugriff auf das Modell und dessen Bearbeitung ermöglichen. Diese Befehle stammen aus dem Modul rfem und nutzen die Funktionen der RFEM-Webservice-API.

Ein wesentliches Element der Modellbearbeitung ist die Verwendung von Modifikationsblöcken. Diese werden durch die Funktionen `begin_modification()` und `finish_modification()` gekennzeichnet. Innerhalb eines solchen Blocks können mehrere Modelländerungen durchgeführt werden, die anschließend gemeinsam gespeichert werden. Dies verbessert die Performance und stellt die Konsistenz des Modells sicher.

```
python

Model.clientModel.service.begin_modification()
Member(1, 1, 2, 1, MemberType.TYPE_BEAM)
Model.clientModel.service.finish_modification()
```

Abb.4.5: Änderungsdienste in der RFEM-Webservice-Schnittstelle

Die Änderungen werden durch `begin_modification()` eingeleitet und mit `finish_modification()` abgeschlossen.

Ein weiteres zentrales Element ist das Objekt `clientModel.service`. Darüber werden alle API-Aufrufe koordiniert. Es stellt die Verbindung zum RFEM-Modell her und ermöglicht das Hinzufügen, Bearbeiten und Löschen von Objekten wie Knoten, Linien, Materialien oder Stäben.

5 Struktur und Funktionsbereiche der RFEM-Python-Bibliothek

5.1 Übersicht der Module in Tabellenform mit Beschreibung

Die RFEM-Python-Bibliothek ist modular aufgebaut und in verschiedene Verzeichnisse unterteilt, die jeweils bestimmte Funktionsbereiche abbilden. Diese Struktur ermöglicht eine klare Trennung zwischen geometrischen Objekten, Belastungen, Auswertung, Bemessung und weiteren Hilfsfunktionen. Im Folgenden werden die wichtigsten Module und ihre Aufgabenbereiche erläutert:

Tabelle.5.1: Tabellarisches Modul mit Beschreibung

Modul / Verzeichnis	Beschreibung
initModel	Die Funktion <code>initModel()</code> wird verwendet, um ein neues Modell zu initialisieren
enums	Die Verwendung von Enums ist besonders wichtig, um spezifische Funktionen strukturiert und eindeutig abzubilden. Die Enums-Bibliothek in Python stellt verschiedene Möglichkeiten zur Verfügung, um solche Aufzählungen zu definieren und gezielt einzusetzen. Beispiele (<code>enums.SurfaceLoaddirection</code>) Für weitere Funktion in Rfem-Bibliothek
BasicObjects	Beinhaltet grundlegende Elemente wie Knoten (Node), Linien (Line), Materialien (Material), Querschnitte (CrossSection) und Stäbe (Member). Diese sind die Basis jedes Modells.
Loads	Enthält alle Arten von Lasten, z. B. Streckenlasten, Punktlasten oder Temperaturlasten auf Linien, Flächen oder Stäbe.
LoadCasesAndCombinations	Dient der Verwaltung von Lastfällen und deren Kombinationen nach Normvorgaben.
TypesFor (Auswahl)	Diese Ordner enthalten Konstanten, Enumerationen und Typdefinitionen für

	z. B. Querschnittstypen, Stabarten, Materialnormen oder Oberflächeneigenschaften.
Results	Ermöglicht den Zugriff auf Analyseergebnisse wie Verschiebungen, Schnittgrößen oder Spannungen.
SpecialObjects	Definiert spezielle Elemente wie Lager, Federverbindungen, Gelenke oder Freigaben.
Tools	Enthält Hilfsfunktionen zur Modellbearbeitung, Prüfung und Konvertierung von Einheiten oder Koordinaten.
GuideObjects	Unterstützt die Definition von Hilfsgeometrien oder Führungselementen im Modell.
ImportExport	Dient dem Datenimport und -export (z. B. DXF, IFC, Excel). Unterstützt den Austausch mit anderen Softwaretools.

5.2 Modellzugriff und Berechnung in Python

Die Kommunikation zwischen dem Python-Skript und RFEM erfolgt über eine zentrale Modellinstanz, die mit folgendem Befehl importiert wird:

```
python
from RFEM.initModel import Model, Calculate_all|
```

Abb.5.1: Aufruf der Initmodel

- Calculate_all() löst nach Abschluss der Modellierung die Berechnung des gesamten Modells aus.

Es ist der Programmäquivalent zum "Berechnen"-Button in der RFEM-Oberfläche.

```
python
Calculate_all()
```

Abb.5.2: Aufruf der Calculate_all

- ClientModel an RFEM-Model stellt die Verbindung zur aktiven RFEM-Modellinstanz her.

Darüber erfolgt der gesamte Zugriff auf das Modell – inklusive der Definition von Knoten, Linien, Stäben, Materialien, Lasten usw.

Alle Objekte werden intern über das sogenannte clientModel an RFEM übergeben:

```
python

Model.clientModel.service.begin_modification()
# ... Modellierungsbefehle ...
Model.clientModel.service.finish_modification()
```

Abb.5.3: Aufbau der clientModel-Struktur innerhalb des RFEM-Modells

5.3 Verwendung von Enums in RFEM

In der Python-Schnittstelle von RFEM werden sogenannte Enums verwendet, um vordefinierte Einstellungen, Richtungen, Typen oder Kategorien eindeutig zu benennen. Diese Enums sind notwendig, damit RFEM versteht, welche Eigenschaften einem Objekt zugewiesen werden sollen – beispielsweise ob eine Last in X- oder Z-Richtung wirkt, ob ein Lager fest oder gelenkig ist oder welcher Typ einem Stab zugeordnet werden soll.

Mit dem Importbefehl `from RFEM.enums import (...)` werden diese standardisierten Bezeichner in das Skript geladen. Sie ermöglichen es, normgerechte, eindeutige und vor allem fehlerfreie Angaben zu machen. Anstatt einen Wert frei einzugeben – was zu Missverständnissen oder Kompatibilitätsproblemen führen kann – wird eine feste, gültige Auswahl verwendet, die direkt mit den Vorgaben von RFEM übereinstimmt.

In der praktischen Anwendung ist dieser Import essenziell, da viele Befehle innerhalb des RFEM-Python-Workflows auf exakt definierte Werte angewiesen sind. Besonders beim Erstellen von Lasten, Querschnitten, Lagerbedingungen, Materialtypen oder bei der Definition von Bemessungssituationen kommen diese Enums zum Einsatz.

Der Einsatz von Enums sorgt somit für eine hohe Stabilität des Codes, eine bessere Lesbarkeit und eine direkte Kompatibilität mit den internen Datenstrukturen von RFEM. Gerade bei umfangreichen oder automatisierten Modellierungsprozessen, wie sie im Rahmen dieser Arbeit mit KI-basierten Systemen verfolgt werden, sind Enums ein unverzichtbarer Bestandteil der API-Kommunikation.

```
python

from RFEM.enums import (...)
```

Abb.5.4: Aufruf von Enums

5.4 BasicObjects – Struktur und Aufruf der Objekte

Damit die Grundelemente eines RFEM-Modells im Python-Skript verwendet werden können, müssen sie zunächst aus dem Modul BasicObjects importiert werden. In Python geschieht dies über sogenannte Importbefehle, mit denen einzelne Klassen aus spezifischen Unterdateien geladen werden.

Der Aufbau folgt dabei einem festen Schema:

```
python  
  
from rfem.basicObjects.<element> import <ElementName>
```

Abb.5.4: Aufruf der basicObjekte

- **Aufruf der Knoten**

Der Aufruf eines Knotens erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.node import Node
```

Abb.5.5: Aufruf der Knoten

- **Aufruf der Linie**

Der Aufruf der Linie erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.line import Line
```

Abb.5.6: Aufruf der Linie

- **Aufruf des Materials**

Der Aufruf des Materials erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.material import Material|
```

Abb.5.7: Aufruf des Materials

- **Aufruf des Querschnittes (Crossection)**

Der Aufruf eines Querschnittes erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.section import CrossSection
```

Abb.5.8: Aufruf des Crossection

- **Aufruf eines Stabes (Member)**

Der Aufruf eines Stabes erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.member import Member|
```

Abb.5.9: Aufruf des Stabes

- **Aufruf einer Fläche (Surface)**

Der Aufruf einer Fläche erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.surface import Surface
```

Abb.5.10: Aufruf der Fläche

- **Aufruf Einer Dicke (Thickness)**

Der Aufruf einer Dicke erfolgt mit dem Befehl:

```
python  
  
from rfem.basicObjects.thickness import Thickness
```

Abb.5.11: Aufruf der Dicke

5.5 Loads – Struktur und Aufruf der Lastobjekte

Das Modul Loads enthält alle Funktionen zur Definition verschiedener Lasttypen im RFEM-Modell. Jede Last Art ist in einer eigenen Python-Datei enthalten und kann gezielt über einen Importbefehl verwendet werden.

Der typische Aufbau des Importbefehls lautet:

```
python  
  
from rfem.Loads.<lastElement> import <LastElementName>
```

Abb.5.12: Aufruf der Lastart

5.5.1 Aufruf einer Linienlast

Importbefehl:

```
python  
  
from rfem.Loads.lineLoad import LineLoad
```

Abb.5.13: Aufruf der Linienlast

5.5.2 Aufruf einer Flächenlast

Importbefehl:

```
python  
  
from rfem.Loads.surfaceLoad import SurfaceLoad
```

Abb.5.14: Aufruf der Flächenlast

5.5.3 Aufruf einer Stablast (MemberLoad)

Importbefehl:

```
python  
  
from rfem.Loads.memberLoad import MemberLoad
```

Abb.5.15: Aufruf der Stablast

5.5.4 Aufruf einer LinienSet-Last

Importbefehl:

```
python  
  
from rfem.Loads.linesetLoad import LineSetLoad
```

Abb.5.16: Aufruf der LinienSet-last

5.5.5 Aufruf einer FlächenSet-Last

Importbefehl:

```
python  
  
from rfem.Loads.surfaceSetLoad import SurfaceSetLoad
```

Abb.5.17: Aufruf der Flächenset-Last

5.5.6 Aufruf einer StabSet-Last

Importbefehl:

```
python  
  
from rfem.Loads.memberSetLoad import MemberSetLoad
```

Abb.5.18: Aufruf der Stabset-Last

5.6 Aufruf die LoadCasesAndCombinations – Lastfälle und Kombinationen in RFEM

Das Modul LoadCasesAndCombinations dient der Definition von Lastfällen (LoadCases) sowie der automatischen oder manuellen Bildung von Kombinationen auf Basis normativer Vorgaben wie dem Eurocode. Es ermöglicht eine strukturierte Steuerung der Belastungsszenarien innerhalb eines RFEM-Modells.

Der Import erfolgt jeweils aus dem Unterverzeichnis:

```
python  
  
from rfem.LoadCasesAndCombinations.<element> import <Element>
```

Abb.5.19: Aufruf der Lastart

5.6.1 Action – Definition von Lastarten

Die Klasse Action definiert die Art der Belastung (ständig, veränderlich, außergewöhnlich).

Diese Kategorisierung ist entscheidend für die richtige Anwendung von Sicherheitsbeiwerten und die Bildung von Kombinationen nach Eurocode.

Beispielhafte Verwendung:

- Eigengewicht → ständige Last
- Schnee, Wind → veränderliche Last

Importbefehl:

```
python  
  
from rfem.LoadCasesAndCombinations.action import Action
```

Abb.5.20: Aufruf der Action

5.6.2 Aufruf ActionCombination – Verknüpfung von Lastarten

Diese Klasse verbindet mehrere Action-Objekte zu Kombinationsgruppen. Du legst hier ψ -Faktoren und Verhältnisse zwischen ständigen und veränderlichen Lasten fest.

Wichtig für:

- automatische Kombinationserstellung
- normgerechte Kombinationen nach EC0 / EC1

Importbefehl:

```
python  
  
from rfem.LoadCasesAndCombinations.actionCombination import ActionCombination
```

Abb.5.21: Aufruf der ActionCombination

5.6.3 Aufruf CombinationWizard – Automatische Kombinationsgenerierung

Mit dem CombinationWizard erzeugst du vollautomatisch gültige Lastkombinationen, z. B. für:

- Ständige + Verkehrslast
- Nutzlasten in verschiedenen Zonen
- Unterscheidung von führender und mitwirkender Last

Importbefehl:

```
python
from rfem.LoadCasesAndCombinations.combinationWizard import CombinationWizard
```

Abb.5.22: Aufruf der ActionCombinationWizard

5.6.4 Aufruf DesignSituation – Definition der Bemessungssituation

DesignSituation bestimmt, in welchem normativen Kontext deine Nachweise erfolgen:

- ULS – STR: Tragfähigkeit
- ULS – EQU: Gleichgewicht
- SLS – FREQ: Gebrauchstauglichkeit (z. B. Schwingungen)

Importbefehl:

```
python
from rfem.LoadCasesAndCombinations.designSituation import DesignSituation
```

Abb.5.23: Aufruf der DesignSituation

5.6.5 Aufruf ModalAnalysisSettings – Einstellungen zur Eigenwertanalyse

Diese Klasse definiert, wie RFEM die Modalanalyse durchführt:

- Anzahl der Eigenformen
- Berechnungsverfahren (Lanczos, Subspace, etc.)
- Massenquelle (Eigengewicht, benutzerdefiniert)

Importbefehl:

```
from rfem.LoadCasesAndCombinations.modalAnalysisSettings import ModalAnalysisSettings
```

Abb.5.24: Aufruf der ModalAnalysisSettings

5.6.6 SpectralAnalysisSettings – Antwortspektrum für Erdbebenaufruf einer StabSet-Last

Zur Durchführung einer Antwortspektrumanalyse nach DIN EN 1998-1 oder anderen Normen.

legt hier u. a. fest:

- Spektrum form

- Standortdaten
- Dämpfungsverhalten

Importbefehl:

```
from rfem.LoadCasesAndCombinations.spectralAnalysisSettings import SpectralAnalysisSettings
```

Abb.5.25: Aufruf der SpectralAnalysisSettings

5.6.7 Aufruf stabilityAnalysisSettings

Stabilitätsnachweise erfolgen z. B. durch:

- Berechnung kritischer Lasten
- Stabilitätsmodi definieren (lokal/global)

Importbefehl:

```
from rfem.LoadCasesAndCombinations.stabilityAnalysisSettings import StabilityAnalysisSettings
```

Abb.5.26: Aufruf der stabilityAnalysisSettings

5.6.8 staticAnalysisSettings – Statische Solver konfiguration Aufruf einer StabSet-Last

Mit dieser Klasse steuerst du, wie RFEM rechnet:

- linear vs. nichtlinear
- Iterationsverfahren
- Konvergenzkriterien

Besonders relevant bei großen oder komplexen FEM-Modellen.

Importbefehl:

```
from rfem.LoadCasesAndCombinations.staticAnalysisSettings import StaticAnalysisSettings
```

Abb.5.27: Aufruf der staticAnalysisSettings

5.6.9 Aufruf windSimulationAnalysisSetting – CFD-Windsimulation

Mit dieser Klasse werden Randbedingungen für die Windströmungssimulation definiert:

- Richtung, Geschwindigkeit

- CFD-Netz
- Gebäudeumgebung

Einsatz z. B. bei Hochhäusern, Membranbauten oder Stadiondächern.

Importbefehl:

```
from rfem.LoadCasesAndCombinations.windSimulationAnalysisSetting import WindSimulationAnalysisSetting
```

Abb.5.28: Aufruf der windSimulationAnalysisSetting

5.7 Aufruf von TypesFor in RFEM

In der RFEM-Python-Schnittstelle werden für viele Objektarten sogenannte Typdefinitionen benötigt. Diese befinden sich in eigenen Modulen mit dem Präfix TypesFor, wie zum Beispiel TypesForLines, TypesForNodes oder TypesForSurfaces.

Mit diesen Typdefinitionen kann exakt beschrieben werden, wie sich ein Objekt im Modell verhalten soll oder welche geometrischen und physikalischen Eigenschaften es besitzt.

Diese Typen sind notwendig, damit RFEM das Modell korrekt interpretieren, zeichnen und berechnen kann. Ohne diese Angaben würde das Verhalten der Objekte unvollständig oder unklar bleiben – was besonders bei automatisierten Modellen oder Sonderformen problematisch wäre.

Die Typdefinitionen werden in Python durch den folgenden Befehl importiert:

```
python
from RFEM.enums.TypesFor... import ...
```

Abb.5.29: Aufruf der Typesfor

5.7.1 Aufruf Typdefinitionen für Linien (TypesForLines)

Das Modul TypesForLines enthält die Typzuweisungen für geometrische Linien im RFEM-Modell.

Damit kann eine Linie nicht nur als reine Verbindung zweier Knoten betrachtet werden, sondern erhält eine eindeutige Formdefinition.

Zu den wichtigsten Typen gehören:

- gerade Linie
- Bogen (z. B. Kreisbogen oder Ellipsenbogen)

- Parabel
- Spline (frei geformte Kurve)

Diese Typangaben sind besonders wichtig für Modelle mit gebogenen Trägern, Kuppelkonstruktionen, wie sie in Fassaden und Brückenbauwerken vorkommen.

RFEM verwendet diese Information, um die Linien korrekt zu visualisieren und zu vernetzen, aber auch für die präzise Ermittlung von Längen, Tangenten oder Krümmungen

Importbefehl:

```
python
from RFEM.enums.TypesForLines import LineType
```

Abb.5.30: Aufruf der Typesforlines

5.7.2 Aufruf Typdefinitionen für Knoten (TypesForNodes)

Das Modul TypesForNodes ermöglicht eine kontextbezogene Klassifikation von Knotenpunkten.

Ein Knoten kann z. B. als rein geometrischer Punkt, als Lagerpunkt, als Lastpunkt oder als Verbindungspunkt zwischen Bauteilen auftreten.

Typische Anwendungsfälle:

- Ein Knoten ist Bestandteil einer Verbindung mehrerer Träger
- An einem Knoten wirken Einzelkräfte oder Momente
- Der Knoten ist Lagerort für eine Verschiebung oder Verdrehung

Durch die Typdefinition erkennt RFEM, welche Funktion der Knoten im Modell übernimmt. Dies ist entscheidend für die spätere Lastzuweisung, Bewegungsfreiheit oder Anschlusskonstruktionen.

Importbefehl:

```
python
from RFEM.enums.TypesForNodes import NodeType
```

Abb.5.31: Aufruf der Typesfornodes

5.7.3 Aufruf Aufruf Typdefinitionen für Knoten (TypesForNodes)

Mit dem Modul TypesForSurfaces kann genau festgelegt werden, wie sich eine Fläche mechanisch verhält.

Zu den wichtigsten Typen gehören:

- Standardfläche: normale Platte oder Schale mit Biegesteifigkeit
- Steifplatte: für nahezu starre Decken, Auflagerplatten oder Knotenverbindungen
- Membranfläche: flexibel, nur zugfähig, für Folien, Planen oder textile Bauteile
- Kontaktflächen: zur Simulation von Kontaktzonen in zusammengesetzten Systemen

Die Angabe des Flächentyps beeinflusst direkt die Berechnungsweise, Netze Zeugung und Physikalische Interpretation der Fläche Insbesondere bei leichten Konstruktionen, Geomembranen oder hybriden Tragwerken sind diese Typen unverzichtbar.

Importbefehl:

```
python  
  
from RFEM.enums.TypesForSurfaces import SurfaceType
```

Abb.5.31: Aufruf der TypesforSurfaces

5.8 Befehle Zusammenfassung

Das RFEM-Python-System ist modular aufgebaut und ermöglicht die automatisierte Modellierung, Lastdefinition, Bemessung und Ergebnisabfrage über Webservices. Die Datei BasicObjects enthält Befehle zur Definition elementarer Strukturobjekte wie Knoten, Linien, Flächen oder Stäbe. In den Lastmodulen werden Funktionen zur Erzeugung von Knoten-, Linien- oder Flächenlasten bereitgestellt. Das Modul für Lastfälle und Kombinationen steuert die Definition von Einwirkungen, Berechnungstypen und Kombinationen. Die Bemessungsmodule für Stahl, Beton, Holz und Aluminium bieten Funktionen zur Konfiguration nach Eurocodes. Über das Results-Modul können Ergebnisse automatisiert ausgelesen und strukturiert weiterverarbeitet werden. Die zentrale Datei enums.py definiert alle konstanten Steuergrößen wie Last- oder Richtungstypen, wodurch die Eingaben standardisiert und fehlersicher gestaltet werden können. Ergänzend dazu ermöglichen globale Parameter, Plausibilitätsprüfungen und Import-Export-Funktionen eine flexible und effiziente Steuerung des gesamten Modellierungsprozesses.

6 Notwendige Unterlagen zur Vorbereitung und Durchführung

Im Rahmen dieses Projekts kamen unterschiedliche Dokumente zum Einsatz, die miteinander verknüpft wurden, um eine strukturierte Darstellung der einzelnen Abschnitte zu ermöglichen. Dies erfolgte unter Verwendung gezielter Befehle, die eine systematische Bearbeitung und Visualisierung der jeweiligen Inhalte unterstützten.

Diese Unterlagen können über GitHub unter folgendem Link heruntergeladen werden: [dlupal-software/RFEM_Python_Client](https://github.com/dlupal-software/RFEM_Python_Client). Dort befindet sich die vollständige Bibliothek des RFEM_Python.Clients mit High-Level-Funktionen für RFEM 6 auf Basis von Web Services, SOAP und WSDL

📁 .github	workflow update upon release fail: Node.js 16 actions are de...
📁 .vscode	launch.json returned to previous state
📁 Examples	Bugfix
📁 RFEM	fix
📁 UnitTests	Revert "package number fix"
📁 assets	Added report from UT and settings
📁 img	logo update
📄 .gitconfig	New parameter on wrong place
📄 .gitignore	minor example fixes
📄 .gitmessage	New parameter on wrong place
📄 .nojekyll	moved nojekyll
📄 CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md
📄 CONTRIBUTING.md	Create CONTRIBUTING.md
📄 LICENSE	version and year update
📄 MANIFEST.in	incorrect slashes
📄 PythonWSClient.code-workspace	Example of export_to and save
📄 README.md	Update README.md
📄 REVIEW_BEST_PRACTICES.md	Review best practices added
📄 pull_request_template.md	Update pull_request_template.md
📄 pylintrc	Added emuns for Action Category and Design Situation.
📄 setup.py	fix

Abb.6.1: Herunterladen der RFEM_Python_Client-Bibliothek über GitHub

6.1 Textdatei im .txt-Format

6.1.1 Introduction .txt-Format

Diese Datei dient als Einstiegspunkt für das KI-gestützte System zur automatisierten Modellgenerierung im Finite-Elemente-Programm Dlubal RFEM. Nach dem Start erhält der Nutzer eine Begrüßungsnachricht mit der Aufforderung, eine allgemeine Beschreibung des gewünschten Modells einzugeben.

Die Datei definiert die Struktur und das Format des Python-Eingabeskripts, das in geordneter Reihenfolge die Module Material, Geometrie, Querschnitte, Lasten, Löser und Ergebnisse umfasst. Zudem enthält sie grundlegende Konventionen zur Modellinitialisierung und Eingabesyntax, um eine fehlerfreie und systematische Modellgenerierung zu gewährleisten.



Abb.6.2: Definition und Syntax eines Material-Templates

6.1.2 Material.txt-Format

Die Datei material.txt enthält allgemeine Befehle zur Definition von Materialien für das RFEM-Modell. Sie dient als Vorlage für die automatische Generierung von Materialien mithilfe eines KI-gestützten Eingabesystems.

Über einen Platzhalter ({placeholder}) kann flexibel ein Materialname oder eine Beschreibung eingefügt werden. Dieser Platzhalter wird im späteren Prozess durch die vom Nutzer eingegebene oder von der KI vorgeschlagene Materialbezeichnung ersetzt.

```

1 Now we define the Material section. {placeholder}
2 This is the syntax of the Material section:
3
4 Material(no: int = 1, name: str = 'S235', comment: str = '', params: dict = None, model = Model)
5
6 avoid to fill params when not requested by the user.
7
8
9 This is an example of the Material section:
10
11 # ---- Material Sec ----
12 Material(1, "S235")
    
```

no: Eindeutige Nummer des Materials im Modell.

name: Bezeichnung des Materials (z. B. "S235").

comment (optional): Freitext zur näheren Beschreibung oder Einordnung.

params (optional): Dict mit Materialparametern (z. B. E-Modul, Dichte)

model: Modellinstanz (Standard: Model).

Abb.6.3: Definition und Syntax eines Material-Templates

Hinweis: Es wird empfohlen, das params-Argument nur dann zu verwenden, wenn spezifische Materialeigenschaften gefordert sind. Ansonsten erfolgt die Auswahl direkt aus der RFEM-Materialbibliothek.

6.1.3 Geometry.txt-Format

Diese Datei enthält die grundlegenden Syntaxelemente zur Definition der Geometrie im RFEM-Modell. Dazu zählen Knoten, Linien (inkl. Kreis-, Bogen- und Splinelinien), Öffnungen sowie Linien- und Knoteneinspannungen.

Die Struktur basiert auf standardisierten Python-Funktionen, die in der richtigen Reihenfolge aufgerufen werden müssen. Wenn vom Nutzer keine Richtung angegeben

wird, wird die positive Höhe in negativer Z-Richtung angenommen. Ziel ist eine effiziente und automatisierte Erstellung der geometrischen Struktur mit nur den notwendigen Elementen.

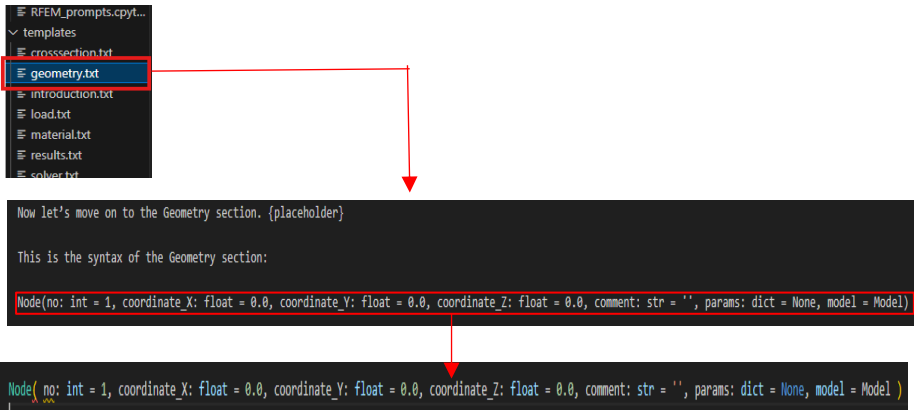


Abb.6.3: Definition und Syntax eines Geometry-Templates

Tabelle.6.1: Geometry Parameters Beschreibung und Bedeutung

Parameter	Bedeutung
no: int	Knotennummer – eine eindeutige Identifikationsnummer für den Knoten.
coordinate_X: float	X-Koordinate – Position des Knotens auf der X-Achse.
coordinate_Y: float	Y-Koordinate – Position des Knotens auf der Y-Achse.
coordinate_Z: float	Z-Koordinate – Position des Knotens auf der Z-Achse.
comment: str	Kommentar – optionaler Text zur Beschreibung oder Kennzeichnung.
params: dict	Zusätzliche Parameter – z. B. benutzerdefinierte Eigenschaften (optional).
model = Model	Modellreferenz – verweist auf das aktuelle RFEM-Modellobjekt.

6.1.4 Crossection.txt-Format

In dieser Datei werden die Querschnittsdefinitionen für das RFEM-Modell beschrieben. Dabei kommen sowohl klassische Stabquerschnitte als auch Dicken- und Flächendefinitionen für Platten und Schalen zum Einsatz. Die Python-Funktionen dienen der eindeutigen und parametrischen Beschreibung der jeweiligen Querschnittsformen.



Abb.6.4: Definition und Syntax eines Crossection-Templates

Tabelle.6.2: Crossection Funktionen Beschreibung und Bedeutung

Funktion	Beschreibung
Section(...)	Definiert einen Stabquerschnitt (z. B. IPE, HEB, Rechteck).
Thickness(...)	Legt eine Dicke für Flächen fest, z. B. für Platten oder Wände.
Surface.Standard(...)	Erstellt eine Fläche mit zugewiesener Geometrie (z. B. Ebene, Schale, Rotationsfläche)
Solid(...)	Definiert ein Volumenelement basierend auf vorhandenen Flächen.

6.1.5 Loads.txt-Format

In dieser Datei werden die Lasten definiert, die auf das RFEM-Modell einwirken. Die Struktur ermöglicht die automatische Erzeugung von Lastfällen sowie von Knoten-, Stab- und Flächenlasten. Standardmäßig wird das Eigengewicht des Modells berücksichtigt, sofern vom Benutzer nicht anders angegeben.

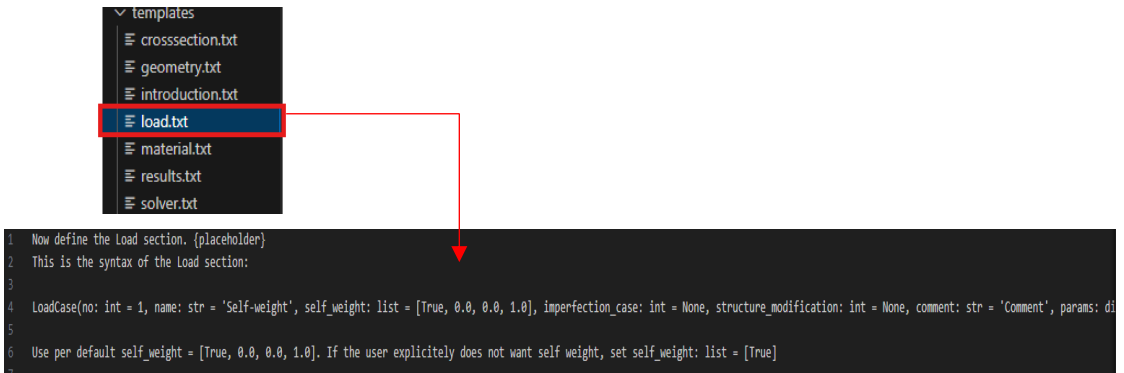


Abb.6.4: Definition und Syntax eines Loads-Templates

6.1.6 Solver.txt-Format

Die solver.txt-Datei beendet die Modellierungsphase und startet die Berechnung des vollständigen RFEM-Modells. Zunächst wird mit dem Befehl

```
Model.clientModel.service.finish_modification()
```

signalisiert, dass alle Eingaben abgeschlossen sind und keine weiteren Änderungen am Modell vorgenommen werden. Anschließend wird mit

```
Calculate_all()
```

die statische Berechnung für alle definierten Lastfälle durchgeführt. Das System analysiert dabei die globalen Gleichgewichtsbedingungen und erstellt die Grundlage für die anschließende Ergebnisauswertung im nächsten Abschnitt.

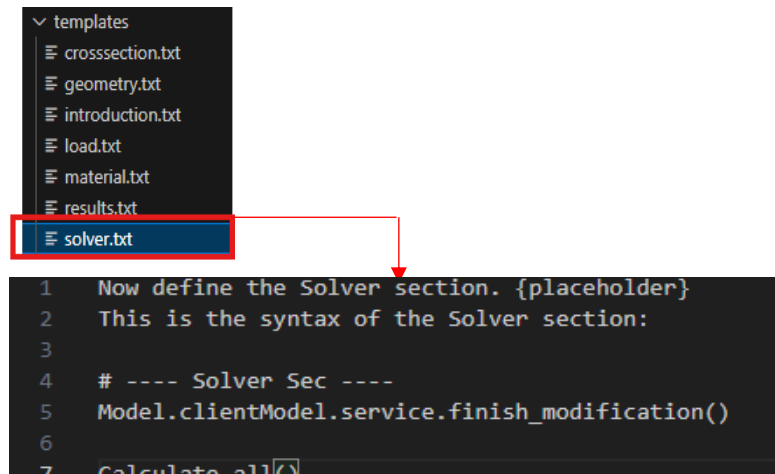


Abb.6.5: Definition und Syntax eines Solver-Templates

6.1.7 Results.txt-Format

Die results.txt-Datei dient der strukturierten Auswertung der Rechenergebnisse aus dem RFEM-Modell. Mit Hilfe verschiedener Funktionen können gezielt Spannungen, Verformungen oder sonstige Ausgabewerte abgefragt, verarbeitet und dokumentiert werden. Die Daten werden dabei zunächst in strukturierter Form gesammelt und anschließend analysiert.

```

Now define the Results section. {placeholder}
# ---- Results Sec ----

import json

# 1. Fläche aus RFEM-Ergebnissen lesen
surface_data = ResultTables.SurfacesGeometry()
total_area = SumValues(surface_data, 'surface_area') # in m²

# 2. Dicke annehmen oder aus Modell laden
thickness = 0.30 # [m] - alternativ: hole aus Modell, falls definiert

# 3. Betonvolumen berechnen
concrete_volume = total_area * thickness # [m³]

# 4. Bewehrungsanteil annehmen (z. B. 80 kg/m³)
reinforcement_kg_per_m3 = 80 # [kg/m³]
reinforcement_mass = concrete_volume * reinforcement_kg_per_m3 # [kg]

# 5. CO₂-Faktoren definieren
co2_per_m3_concrete = 720 # [kg CO₂ / m³]
co2_per_kg_steel = 1.8 # [kg CO₂ / kg]

# 6. CO₂ berechnen
concrete_co2 = concrete_volume * co2_per_m3_concrete
steel_co2 = reinforcement_mass * co2_per_kg_steel
total_co2 = concrete_co2 + steel_co2

# 7. Kosten (optional)
cost_concrete_per_m3 = 100 # €/m³ (z. B.)
cost_steel_per_kg = 2.5 # €/kg
total_cost = concrete_volume * cost_concrete_per_m3 + reinforcement_mass * cost_steel_per_kg

# 8. Ergebnisse anzeigen
print("Gesamtfläche:", round(total_area, 2), "m²")
print("Betonvolumen:", round(concrete_volume, 2), "m³")
print("Stahlmasse:", round(reinforcement_mass, 2), "kg")
print("Beton CO₂:", round(concrete_co2, 1), "kg")
print("Stahl CO₂:", round(steel_co2, 1), "kg")
print("Gesamt CO₂:", round(total_co2, 1), "kg")
print("Gesamtkosten:", round(total_cost, 2), "€")
    
```

Abb.6.6: Definition und Syntax eines Results-Templates

- 1. Flächenermittlung:** Über ResultTables.SurfacesGeometry() wird die Gesamtfläche der im Modell enthaltenen Flächen in Quadratmetern ausgelesen

→ **2. Dickenfestlegung:** Die Plattendicke wird angenommen (z. B. 30 cm) oder kann später aus dem Modell übernommen werden.

→ **3. Volumenberechnung:** Das Betonvolumen ergibt sich aus Fläche × Dicke und wird in Kubikmetern angegeben.

→ **4. Bewehrungsmasse:** Die Stahlmenge wird über einen typischen Bewehrungsanteil (z. B. 80 kg/m³) abgeschätzt.

→ **5. CO₂-Faktoren:** Für Beton und Stahl werden standardisierte Emissionsfaktoren definiert (z. B. 720 kg CO₂/m³ für Beton und 1,8 kg CO₂/kg für Stahl).

→ **6. CO₂-Bilanzierung:** Die Emissionen für Beton und Bewehrung werden separat berechnet und anschließend aufsummiert.

→ **7. Kostenschätzung:** Basierend auf üblichen Marktpreisen werden die Materialkosten für Beton und Bewehrung ermittelt.

→ **8. Ausgabe:** Die berechneten Kennwerte – Fläche, Volumen, Stahlmasse, CO₂ und Kosten – werden in der Konsole angezeigt.

6.2 RFEM-Prompts im .py-Format

Das folgende Python-Skript bildet den zentralen Mechanismus zur Verarbeitung nutzerdefinierter Prompts im KI-gestützten RFEM-Modellierungsprozess. Es verbindet Benutzereingaben mit vordefinierten Textvorlagen (Templates) und generiert daraus standardisierten Python-Code für RFEM. Der Aufbau des Moduls lässt sich wie folgt erklären:

1. Einlesen von Textdateien
2. Laden der Templates
3. Zuordnung von Schlüsselwörtern
4. Verarbeitung des Benutzerprompts

```
1 # Module for using prompt templates to define RFEM 0 models
2
3 def read_file(file_path):
4     with open(file_path, "r", encoding="utf-8") as file:
5         return file.read()
6
7 _INTRODUCTION_TPL = read_file("templates/introduction.txt")
8 _MATERIAL_TPL = read_file("templates/material.txt")
9 # Hinweis: Flächen-normals in GEOMETRY mit Surface(...) definieren
10 # Verwende Surface-Standard nur in CROSSSECTION
11 _GEOMETRY_TPL = read_file("templates/geometry.txt")
12
13 # Hinweis: Template angepasst
14 # Verwende in CROSSSECTION ausschließlich Surface-Standard(...) und keine keyword arguments wie thickness=...
15 _CROSSSECTION_TPL = read_file("templates/crosssection.txt")
16
17 # Hinweis: In RFEM zeigt die Z-Achse nach unten -> also 'nach unten' = positive Last in Z-Richtung
18 # GPT soll daher die Last als +Wert in GLOBAL_Z eintragen (z. B. +3000)
19 _LOAD_TPL = read_file("templates/load.txt")
20 _SOLVER_TPL = read_file("templates/solver.txt")
21 _RESULTS_TPL = read_file("templates/results.txt")
22
23 _keywords = {
24     'INTRO': _INTRODUCTION_TPL,
25     'MATERIAL': _MATERIAL_TPL,
26     'GEOMETRY': _GEOMETRY_TPL,
27     'CROSSSECTION': _CROSSSECTION_TPL,
28     'LOADS': _LOAD_TPL,
29     'SOLVER': _SOLVER_TPL,
30     'RESULTS': _RESULTS_TPL,
31 }
32
33 # Process prompt to use template if applicable
34 # The template keyword shall be specified at the beginning of the prompt with a semicolon
35 # Example:
36 # MATERIAL: My model...
37 def process_prompt(prompt):
38     if ';' not in prompt:
39         return prompt
40     key, value = prompt.split(':', 1)
41     if key not in _keywords:
42         return prompt
43     return _keywords[key].format(placeholder=value)
```

1. Einlesen von Textdateien

2. Laden der Templates

3. Zuordnung von Schlüsselwörtern

4. Verarbeitung des Benutzerprompts

Die Funktion verarbeitet die Eingabe des Nutzers in Form eines Prompts, zum Beispiel LOADS: Eine Flächenlast auf Fläche 1. Zuerst wird geprüft, ob ein Doppelpunkt enthalten ist und ob das Schlüsselwort vor dem Doppelpunkt im Wörterbuch `_keywords` vorhanden ist. Ist dies der Fall, wird die passende Vorlage geladen und der Platzhalter `{placeholder}` durch den eingegebenen Text ersetzt. Das Ergebnis ist ein vollständiger, automatisch generierter Python-Code für RFEM.

In diesem Dictionary werden sogenannte Schlüsselwörter (z. B. 'GEOMETRY', 'LOADS') mit den jeweils passenden Templates verknüpft. Dieses Mapping ermöglicht es dem System später, anhand eines Prompts wie GEOMETRY: Eine Fläche mit vier Knoten... automatisch das richtige Template auszuwählen.

Hier werden die sieben standardisierten Vorlagendateien (Templates) für die Modellierungsschritte in RFEM eingelesen. Jeder dieser Schritte hat eine eigene .txt-Datei, die die Code-Struktur vorgibt. Das Ergebnis ist sieben Variablen, in denen die vollständigen Inhalte der jeweiligen Textdateien gespeichert sind.

Diese Funktion liest eine externe Textdatei ein und gibt ihren gesamten Inhalt als Zeichenkette zurück. Die Datei wird im Lese-Modus ("r") mit UTF-8-Kodierung geöffnet. Das with-Statement sorgt dafür, dass die Datei nach dem Lesen automatisch geschlossen wird. Diese Funktion wird genutzt, um alle RFEM-spezifischen Templates zu laden.

Abb.6.7: Definition und Erklärung RFEM-Prompts

6.3 RFEM-Bot im .py-Format

Das folgende Skript stellt den interaktiven Ablauf dar, mit dem ein Nutzer mithilfe eines KI-Moduls (OpenAI GPT) schrittweise ein RFEM-Modell definieren kann. Die Kommunikation erfolgt über sogenannte **Prompts**, die vom Benutzer eingegeben, automatisch verarbeitet und von der KI interpretiert werden.

Zunächst wird die OpenAI-Bibliothek eingebunden und der benötigte API-Schlüssel aus der Systemumgebung (os.getenv) geladen. Anschließend wird eine leere Liste conversation erstellt, die den bisherigen Verlauf des Dialogs speichert.

Im ersten Schritt begrüßt das System den Benutzer mit einer farblich hervorgehobenen Startnachricht und fordert ihn auf, eine allgemeine Beschreibung des gewünschten RFEM-Modells einzugeben. Diese Eingabe wird durch die Funktion process_prompt(...) verarbeitet, also ggf. in ein passendes Template überführt, und als Systemnachricht in den Konversationsverlauf eingefügt.

Ab diesem Moment beginnt die eigentliche Schleife: Der Nutzer gibt weitere Prompts ein (z. B. zur Geometrie, den Materialien oder Lasten), die ebenfalls durch process_prompt(...) interpretiert werden. Jede Eingabe wird der conversation-Liste hinzugefügt. Anschließend wird die Anfrage an die OpenAI-API gesendet. Dabei wird das Modell "gpt-4o" verwendet. Die Antwort der KI wird ebenfalls ausgegeben und gespeichert – farblich hervorgehoben für bessere Lesbarkeit. Diese Schleife wird kontinuierlich wiederholt und erlaubt somit einen dialogbasierten, schrittweisen Aufbau eines vollständigen RFEM-Modells – von der ersten Beschreibung bis zur automatischen Generierung von Python-Code.

```

1 from RFEM_prompts import *
2 import openai
3 import os
4
5 openai.api_key = os.getenv('OPENAI_API_KEY')
6
7 # Initialize conversation
8 conversation = []
9
10 # Example OpenAI Python library request
11 MODEL = "gpt-4o"
12
13 # Conversation loop
14 while True:
15     # Get user message and process prompt
16     if len(conversation) == 0:
17         print('\n')
18         print("\033[34m I am an AI bot that will help you define your RFEM model. Please give me a general instruction and a description of the model to begin with. \033[0m")
19         general_instruction = input('\n')
20         processed_instruction = process_prompt(general_instruction)
21         conversation.append({"role": "system", "content": processed_instruction})
22         print()
23         print("\033[34m How you can continue defining the different modules of the RFEM model. \033[0m")
24         user_prompt = input('\n')
25         processed_prompt = process_prompt(user_prompt)
26         # Add message to conversation
27         conversation.append({"role": "user", "content": processed_prompt})
28         # Get chatGPT answer
29         response = openai.ChatCompletion.create(
30             model=MODEL,
31             messages=conversation,
32             temperature=0,
33         )
34         # Print answer
35         assistant_answer = response["choices"][0]["message"]["content"]
36         print()
37         print("\033[34m\033[34m[assistant_answer]\033[0m")
38         # Add answer to conversation
39         conversation.append({"role": "assistant", "content": assistant_answer})

```

Auswahl OpenAI Model

Die Begrüßungsnachricht der KI lautet

Die Modellierung erfolgt auf Basis vordefinierter Templates

Abb.6.8: Definition und Erklärung der RFEM-Bot

6.4 Integration von Prompts, Templates und GPT in einem KI-gestützten Modellierungssystem

Zur automatisierten Generierung von statischen Modellen (z. B. zylindrischen Tanks oder Fundamenten) wird ein modulares KI-gestütztes System entwickelt, das die Interaktion zwischen Nutzereingaben (Prompts), vordefinierten Templates (Vorlagen) und einem Sprachmodell wie GPT nutzt. Das Ziel dieser Struktur besteht darin, die Modellierung technischer Bauteile effizient, wiederverwendbar und intuitiv steuerbar zu machen.

1. Nutzereingabe (Prompt)

Der Einstieg in die Modellgenerierung erfolgt über einen natürlichsprachlichen Prompt, der vom Nutzer über eine Benutzerschnittstelle (z. B. Web-App) eingegeben wird. Dieser Prompt kann beispielsweise wie folgt lauten:

„Erstelle einen zylindrischen Stahltank mit Durchmesser 6 m und Höhe 10 m.“

Solche Eingaben enthalten technische Informationen, die vom System extrahiert und in strukturierte Parameter überführt werden müssen.

2. Prompt-Analyse und Modulzuordnung

Mithilfe des Sprachmodells GPT (z. B. GPT-4) wird der Prompt semantisch analysiert. Dabei erkennt das System relevante Schlüsselwörter wie „zylindrisch“, „Stahltank“, „Durchmesser“ oder „Höhe“ und ordnet diese einem geeigneten Modellierungsmodul zu. In diesem Fall wird der Prompt dem Modul *Tankmodell* zugewiesen.

Die GPT-gestützte Analyse ermöglicht eine flexible Interpretation der Nutzereingaben, auch bei abweichender oder unvollständiger Formulierung.

3. Template-Auswahl

Basierend auf der Modulzuweisung wird das passende Template aus einer Template-Bibliothek geladen. Ein Template ist eine strukturierte Text- oder Codevorlage mit definierten Platzhaltern (z. B. `{{diameter}}`, `{{height}}`, `{{material}}`), die später durch die erkannten Parameter ersetzt werden.

Beispiel eines Templates für die Geometrie eines zylindrischen Tanks:

```
# Template: template_tank_model.txt
model_name = "{{model_name}}"
create_cylindrical_tank(diameter={{diameter}}, height={{height}}, material="{{material}}")
```

Abb.6.9: Template Output

4. Parameterextraktion und Template-Füllung

Die im Prompt identifizierten Parameter werden nun extrahiert und in das Template eingefügt. Dabei wird automatisch eine Substitution der Platzhalter durchgeführt. Für obiges Beispiel ergibt sich folgender ausgefüllter Code:

```
model_name = "Tank_001"
create_cylindrical_tank(diameter=6.0, height=10.0, material="S235")
```

Abb.6.10: Template-Füllung

5. Codegenerierung und Ausgabe

Das vollständig befüllte Template stellt den finalen Modellierungscode dar, der direkt an eine externe Schnittstelle – beispielsweise die RFEM Webservices API – übergeben wird. Die Modellierung erfolgt damit automatisiert und ohne manuelle Eingriffe.

Visual Prompt Flow

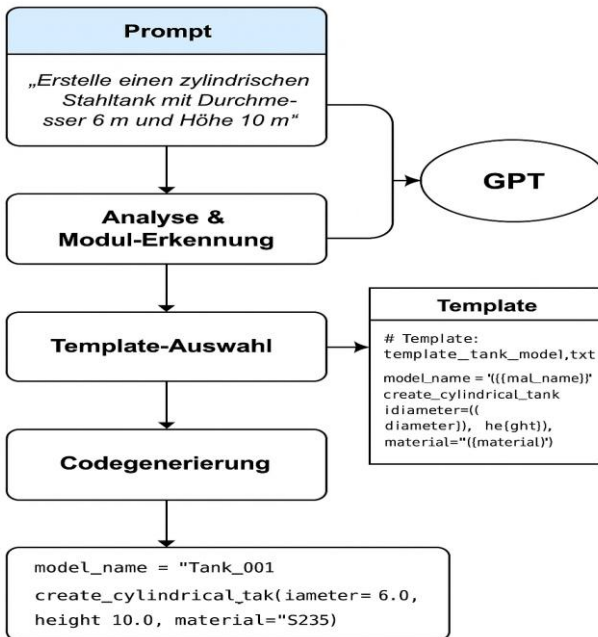


Abb.6.11: Systemablauf – Zusammenfassung

7 Erforderliche Dateien und Programme für die Systemvorbereitung und -ausführung

7.1 Erforderliche Programme

Für die Vorbereitung, Entwicklung und Ausführung des KI-gestützten Modellierungssystems werden mehrere Softwarekomponenten benötigt. Jedes dieser Programme erfüllt dabei eine spezifische Funktion im Gesamtworkflow:

1. GitHub

GitHub dient zur Versionsverwaltung und zentralen Ablage aller Projektdaten und Skripte. Es ermöglicht:

- die strukturierte Organisation des Codes (z. B. in Modulen und Templates),
- die Nachverfolgung von Änderungen im Verlauf der Entwicklung,
- die Zusammenarbeit mit anderen Beteiligten (z. B. durch Pull-Requests),
- sowie die Sicherung und Synchronisation des Projekts über verschiedene Geräte.



2. Visual Studio Code

Visual Studio Code (VS-Code) wird als integrierte Entwicklungsumgebung (IDE) verwendet. Es bietet eine benutzerfreundliche Oberfläche zur Bearbeitung und Ausführung von Python-Skripten sowie zur:

- Entwicklung und Testung der GPT-Integration,
- Verwaltung der Template-Logik,
- Programmierung der Verbindung zur RFEM-API,
- sowie zur Formatierung und Fehlersuche (Debugging).



3. RFEM – Dlubal

RFEM von Dlubal ist die Strukturanalyse-Software, in der die automatisch generierten Modelle berechnet und ausgewertet werden. In diesem Projekt wird RFEM über die Webservices API angesprochen, um:

- Bauteile automatisch zu generieren,
- Lasten und Randbedingungen zu definieren,
- Berechnungen durchzuführen,
- und anschließend Ergebnisse (z. B. Spannungen, Verschiebungen) auszulesen.



4.PowerShell

Die Windows PowerShell wird für die automatisierte Ausführung von Skripten und Kommandos verwendet, insbesondere zur:

- Steuerung des lokalen Servers bzw. der API-Verbindung zu RFEM,
- Ausführung von Setup-Skripten zur Initialisierung der Umgebung,
- Automatisierung von Dateioperationen (z. B. Ergebnisexporte, Log-Erstellung),
- sowie zur Steuerung der Kommunikation zwischen den einzelnen Modulen über CLI-Befehle.



7.2 Erforderliche Dateien

Um mit der RFEM Webservices API arbeiten zu können, ist ein gültiger API-Schlüssel erforderlich. Dieser Schlüssel authentifiziert den Zugriff auf die RFEM-Schnittstelle

Erforderliche Dateien und Programme für die Systemvorbereitung und -ausführung

und ermöglicht die Kommunikation zwischen dem Python-Client und der RFEM-Anwendung. Die Erstellung und Einbindung des API-Schlüssels erfolgt in mehreren Schritten

Schritte zur Erstellung und Einbindung eines API-Schlüssels:

1- API-Plattform öffnen:

Die Schlüsselgenerierung erfolgt in der Regel über die RFEM-API-Webplattform oder ein lokales Konfigurationstool, das die Webservices aktiviert.

2- API-Schlüssel erzeugen:

Nach dem Einloggen wird ein neuer API-Schlüssel generiert. Dieser besteht aus einem eindeutigen alphanumerischen Token, das zur Autorisierung dient.

3- API-Schlüssel kopieren:

Der erzeugte Schlüssel wird einmalig angezeigt und muss sicher kopiert und gespeichert werden.

4- Schlüssel lokal einfügen:

Der API-Schlüssel wird im Projektverzeichnis abgelegt, z. B.: als Umgebungsvariable (.env-Datei) oder direkt im Python-Skript (nur zu Testzwecken, nicht empfohlen für Produktivbetrieb).

• Verbindung testen:

Nach dem Einfügen kann die Verbindung zur RFEM API mit einem Testskript überprüft werden, um sicherzustellen, dass die Authentifizierung korrekt funktioniert.



Abb.7.1: Schritte – Zusammenfassung

8 Exemplarische Darstellung und methodisches Vorgehen

Ein L-förmiges Fundament dient als Referenzmodell zur Anwendung eines KI-gestützten Optimierungsverfahrens. Die Dicke der Platte sowie die Bewehrung werden als variable Eingangsgrößen definiert. Beide Parameter beeinflussen maßgeblich die CO_2 -Emissionen und die Kostenbewertung des Tragwerks.

Das Modell besteht aus zwei orthogonal zueinanderstehenden Platten – einer vertikalen und einer horizontalen Fläche – die geometrisch L-förmig verbunden sind. Es wirken sowohl das Eigengewicht als auch eine positionsabhängige variable Flächenlast auf der vertikalen Platte sowie eine konstante Flächenlast auf der horizontalen Platte. Alle Elemente und Lasten sind positionsbasiert definiert, sodass eine gezielte Analyse der Wirkungen auf CO_2 und Kosten ermöglicht wird.

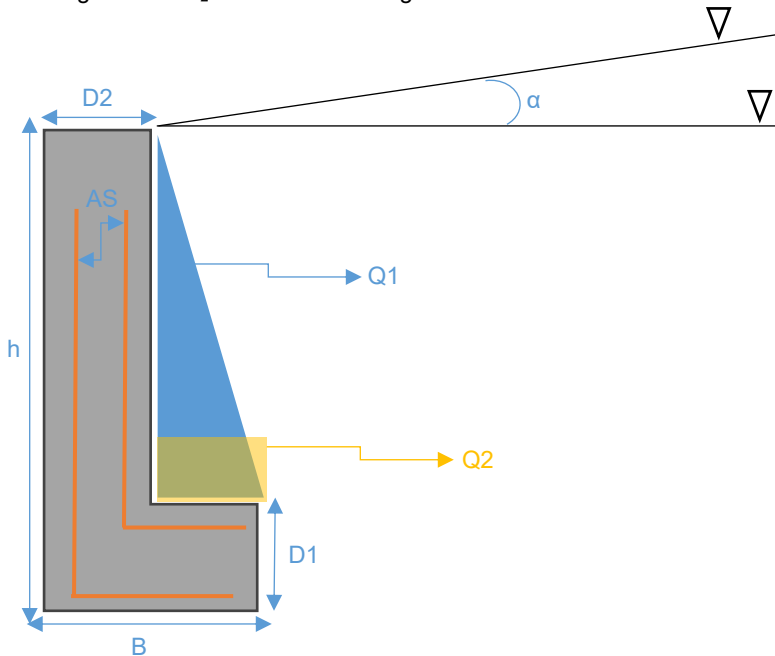


Abb.8.1: Beispielhafte Modellskizze

❖ Definition der Modellparameter:

H = Gesamthöhe des Fundamentsystems

B = Breite des Fundamentkörpers

D_1 = Dicke der horizontalen Fundamentplatte

D_2 = Dicke der vertikalen Wandplatte

Q_1 = Variable Flächenlast infolge des seitlich anstehenden Erdreichs

Q_2 = Flächenlast infolge des Eigengewichts und des aufstehenden Erdreichs

Zur systematischen Umsetzung erfolgt die Bearbeitung nachfolgender Arbeitsschrittfolge:

1. Herunterladen und Einrichten der RFEM-Python-Bibliothek
2. Erzeugen und Definieren des API-Schlüssels zur Schnittstellenkommunikation
3. Konfiguration und Anpassung des Systems
4. Programmierung und Skripterstellung mit Visual Studio Code
5. Modellbildung und Durchführung der Berechnung über die RFEM-API

8.1 Herunterladen der RFEM-Python-Bibliothek

Die benötigte Bibliothek ist auf GitHub verfügbar und kann über den folgenden Link heruntergeladen werden.

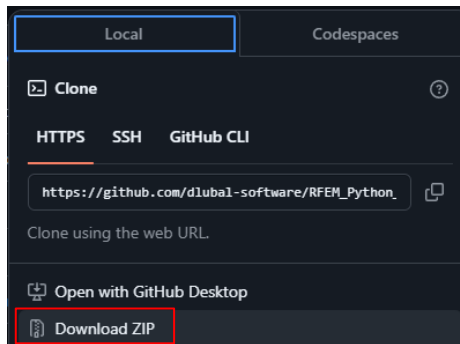


Abb.8.2: Herunterladen der RFEM-Bibliothek

Nach dem Herunterladen muss die Datei entpackt werden, um anschließend in Visual Studio in die weitere Bearbeitung eingebunden werden zu können.

8.1.1 Einbindung der RFEM-Bibliothek in Visual Studio

Die Bibliotheksdatei kann in Visual Studio eingebunden werden, um auf den grundlegenden Quellcode zur weiteren Bearbeitung zuzugreifen.

Hierzu ist zunächst die Option „Ordner öffnen“ (Open Folder) zu wählen, um das gesamte Projektverzeichnis zu laden. Anschließend kann eine spezifische Datei, wie beispielsweise ChatGPT_Tank.py, ausgewählt werden.

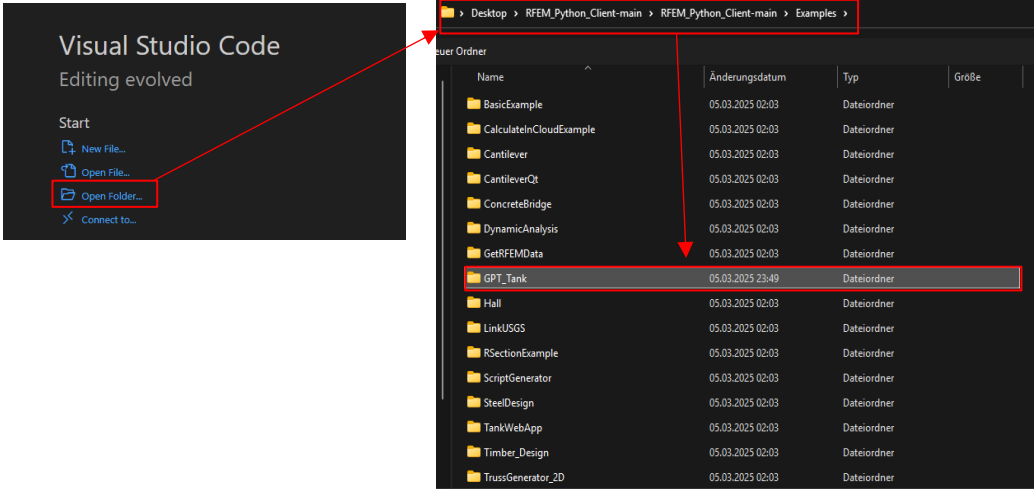


Abb.8.2: Konfiguration der RFEM-Schnittstelle in Visual Studio

Im geöffneten Projektverzeichnis sind bereits vorbereitete Strukturen erkennbar, darunter Template-Dateien, Prompt-Skripte sowie ein konfigurierter Chatbot-Bereich, die als Grundlage für die Weiterentwicklung dienen.

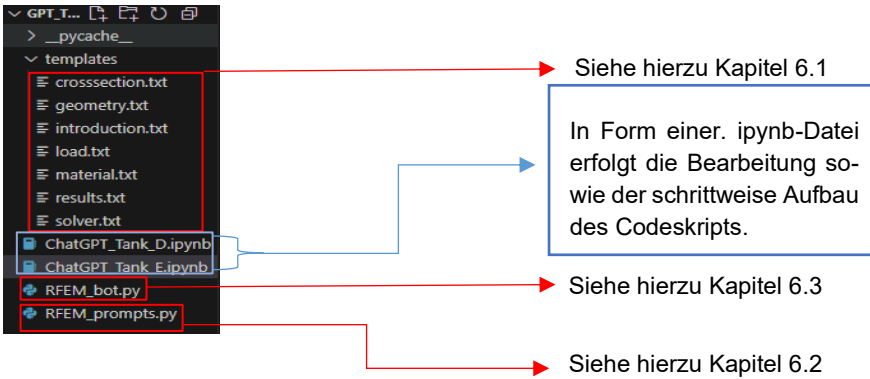


Abb.8.3: Bestandteile der RFEM-Python-Bibliothek

8.2 Initialisierung des API-Schlüssels zur Nutzung der Programmierschnittstelle

Zur erfolgreichen Kommunikation zwischen dem Nutzer und dem GPT-Modell ist die Erstellung eines API-Schlüssels erforderlich. Dieser Schlüssel dient als Authentifizierungsmerkmal und ermöglicht den Zugriff auf die entsprechenden Dienste. Die Generierung des API-Schlüssels erfolgt ausschließlich über die jeweilige API-Plattform.

Über die entsprechende API-Plattform kann ein individueller API-Schlüssel erstellt werden. Unter dem angegebenen Link besteht die Möglichkeit zur Anmeldung, woraufhin der Nutzer Zugriff auf den Bereich zur Schlüsselverwaltung erhält.

<https://platform.openai.com/api-keys>

Abb.8.4: API- Plattform Link

Nach erfolgreicher Registrierung bzw. Anmeldung ist der Bereich „API Keys“ auszuwählen. Dort kann ein neuer Schlüssel über die Funktion „Create New Key“ generiert werden.

Im Anschluss besteht die Möglichkeit, einen Namen sowie einen zugehörigen Projektnamen zu vergeben.

Der generierte Schlüssel sollte **unmittelbar** nach der Erstellung gesichert werden, da eine spätere Anzeige nicht möglich ist.

Systemanpassung und – einrichtung

Die erste Maßnahme ist der API-Schlüssel im System zu hinterlegen, damit dieser bei der Kommunikation mit der Programmierschnittstelle verwendet werden kann. Darüber hinaus sind vorbereitende Maßnahmen wie die Installation der Programmiersprache Python notwendig, um das System betriebsbereit zu machen.

8.3.1 Speicherung des API-Schlüssels im System

Zum Speichern des API-Schlüssels auf dem lokalen Rechner kann PowerShell verwendet werden. Nach dem Öffnen der PowerShell-Konsole ist folgender Befehl einzugeben. Anstelle des Platzhalters ist der persönliche API-Schlüssel von der entsprechenden Plattform einzusetzen:

```
$env:API_KEY = "dein_api_schlüssel"
```

Abb.8.6: Befehl zur lokalen Speicherung des API-Schlüssels im System

8.3.2 Installation der Programmiersprache Python

Um Python auf dem System zu installieren, ist in PowerShell der nachstehende Befehl auszuführen. Dadurch wird die aktuelle Version automatisch heruntergeladen und eingerichtet.

```
winget install Python.Python.3
```

Abb.8.7: Befehl zur Python Installation

8.3.3 Überprüfung der API-Schlüssel-Funktionalität in PowerShell

In PowerShell kann der folgende Befehl eingegeben werden:

```
python rfem_bot.py
```

Abb.8.8: Befehl zur Überprüfung von RFEM_bot.py

Nach dem Ausführen über die Eingabetaste wird das Skript *rfem_bot.py* gestartet. Dadurch wird der zugehörige Ordner angesprochen und der KI-Bot initialisiert.

Erscheint anschließend die folgende Meldung:

„I am an AI bot that will help you define your RFEM model. Please give me a general instruction and a description of the model to begin with.“

so zeigt dies an, dass die Verbindung zur API erfolgreich hergestellt wurde und der Bot ordnungsgemäß funktioniert.

```
PS C:\Users\qasho\Desktop\VM.Sc\Untersuchung von KI\RFEM_Python_Client-main\RFEM_Python_Client-main\Examples\GPT_Tank> py  
thon rfem_bot.py  
✔ API-Schlüssel erfolgreich geladen!  
I am an AI bot that will help you define your RFEM model. Please give me a general instruction and a description of the  
model to begin with.
```

Abb.8.9: Prüfergebnisse der API-Schlüssel-Initialisierung

8.4 Erstellen und Bearbeiten von Skripten in Visual Studio Code

Mit Visual Studio Code können Skripte aus der RFEM-Bibliothek angezeigt und bearbeitet sowie neue Skripte erstellt werden. Innerhalb der RFEM-Bibliothek befindet sich ein spezieller Ordner mit dem Namen „chat Gpt D“, in dem zusätzliche oder neue Codeabschnitte eingefügt werden können.

8.4.1 Entwicklung des Hauptskripts

In diesem Abschnitt befinden sich die allgemeinen Basisbefehle zur Initialisierung des RFEM-Modells. Wir möchten zusätzlich spezifische Befehle einfügen, die auf unseren jeweiligen Berechnungszielen basieren. Im konkreten Beispiel – der Bemessung der erforderlichen Bewehrung (As) und Plattendicke – ist es notwendig, bestimmte Add-Ons zu aktivieren und gezielte Einstellungen vorzunehmen.

Dazu gehören insbesondere die Aktivierung der Betonbemessung, CO₂-Bewertung, Optimierung sowie die Definition relevanter Globalparameter.

```
import os  
import sys  
  
from RFEM.enums import LineSupportType, SurfaceGeometry, SurfaceLoadDirection, SurfaceLoadDistribution, ActionCategoryType  
from RFEM.initModel import calculate_all, Model  
from RFEM.BasicObjects.node import Node  
from RFEM.BasicObjects.line import Line  
from RFEM.BasicObjects.material import Material  
from RFEM.BasicObjects.thickness import Thickness  
from RFEM.BasicObjects.surface import Surface  
from RFEM.TypesForLines.lineSupport import LineSupport  
from RFEM.TypesForNodes.nodalSupport import NodalSupport  
from RFEM.Results.resultTables import GetMaxValue, ResultTables  
from RFEM.Loads.surfaceLoad import SurfaceLoad  
from RFEM.LoadCasesAndCombinations.loadCase import LoadCase  
from RFEM.ImportExport.exports import ExportTo  
  
Model(True, 'ChatGPT_Tank.rf6')  
Model.clientModel.service.begin_modification()
```

Alle wesentlichen Ergänzungen zur Modellerstellung und Berechnung sind hier zu implementieren.

Abb.8.10: Hauptmodul der RFEM-Bibliothek

8.4.1.1 Entwicklung von Basisangaben

in diesem Abschnitt müssen zwei Optionen aktiviert werden: die Betonbemessung und die CO₂-Bewertung.

Dafür werden zwei spezifische Befehle benötigt, um die entsprechenden Add-Ons in RFEM zu aktivieren.

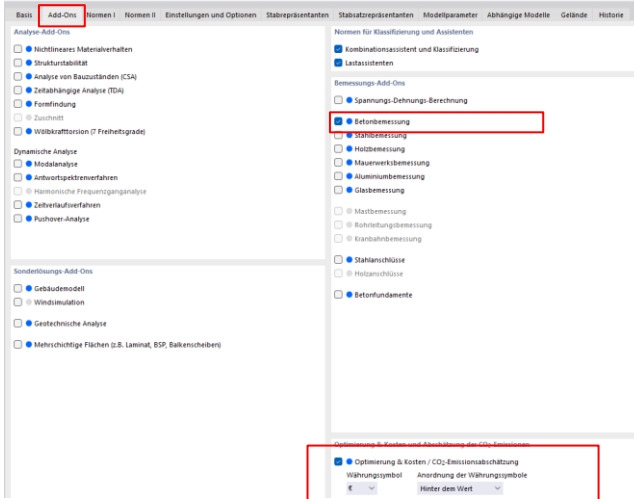


Abb.8.11: RFEM-Dlubal Basisangaben Fenster, add-ons

Damit die Betonbemessung im Rahmen des automatisierten Modellierungs- und Bemessungsprozesses durchgeführt werden kann, ist es erforderlich, das entsprechende Add-On innerhalb der RFEM-Umgebung zu aktivieren.

Dies geschieht durch die gezielte Einbindung eines Befehls, der dafür sorgt, dass die Funktionalität zur Betonbemessung gemäß den gültigen Normen (z. B. Eurocode 2) zur Verfügung steht. um die Betonbemessung zu aktivieren und die nachfolgenden Rechenschritte vorzubereiten:

```
SetAddonStatus(Model.clientModel,addOn=Addon.concrete_design_active, status=True)
```

Abb.8.12: Betonbemessung Befehl in RFEM-Bibliothek

Neben der Betonbemessung ist im Rahmen des nachhaltigen Planungsansatzes auch die Bewertung der CO₂-Emissionen ein zentraler Bestandteil des Optimierungsprozesses.

Um diese Funktionalität im Skript zu aktivieren, muss das entsprechende Add-On zur CO₂-Berechnung in RFEM aktiviert werden.

Zu diesem Zweck wird der folgende Befehl in das Skript integriert:

```
SetAddonStatus(Model.clientModel,addOn=Addon.cost_estimation_active, status=True)
```

Abb.8.13: CO2 Befehl in RFEM-Bibliothek

8.4.1.3 Entwicklung von Lastfällen und Kombinationen

In diesem Abschnitt wird die Lastkombination definiert, wie sie auch in der Benutzeroberfläche von Dlubal RFEM dargestellt wird.

Die Definition erfolgt scriptbasiert, entspricht jedoch inhaltlich der manuellen Eingabe über die grafische Oberfläche von RFEM.

Ziel ist es, die maßgebende Kombination aller relevanten Einwirkungen so zu konfigurieren, dass sie für die spätere Bemessung und Optimierung verwendet werden kann.

Zusätzlich zur Definition der Lastkombination ist es erforderlich, eine Bemessungssituation (Design Situation) festzulegen. Diese dient dazu, die gültigen Kombinationen entsprechend der Nachweisform (z. B. Grenzzustand der Tragfähigkeit – ULS oder Gebrauchstauglichkeit – SLS) einzuordnen.

Die Bemessungssituation legt fest, welche Kombinationen für welche Art von Nachweis relevant sind und wird somit zur Grundlage für die automatisierte Betonbemessung sowie für die spätere CO₂-Optimierung.

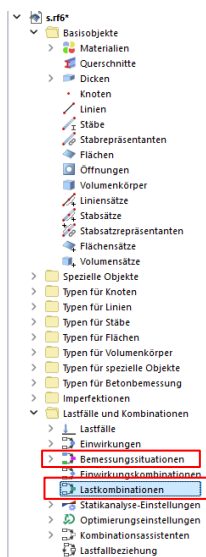


Abb.8.14: RFEM-Dlubal Lastfälle und Kombinationen

Um die Bemessungssituation scriptbasiert in das Modell einzufügen, wird der folgende Befehl verwendet. Damit lässt sich direkt im Python-Skript festlegen, unter welchen Randbedingungen die Nachweise erfolgen sollen – z. B. für den Grenzzustand der Tragfähigkeit (ULS) oder die Gebrauchstauglichkeit (SLS). Die Definition entspricht der manuellen Eingabe in der RFEM-Oberfläche und ist notwendig für die nachfolgende Betonbemessung und Optimierung.

```
● from RFEM.LoadCasesAndCombinations.designSituation import DesignSituation
```

Abb.8.15: Lasfällen und Kombinationen Befehl in RFEM-Bibliothek

Um die Lastkombination scriptbasiert in das Modell einzufügen, wird der folgende Befehl verwendet.

Dieser ermöglicht es, die gewünschte Kombination direkt im Python-Skript zu definieren und somit automatisiert in das RFEM-Modell zu integrieren.

Die Funktion entspricht der manuellen Eingabe über die grafische Benutzeroberfläche von RFEM und erlaubt eine präzise Steuerung der Kombinationsparameter:

```
● from RFEM.LoadCasesAndCombinations.loadCombination import LoadCombination
```

Abb.8.16: Lasfällen und Kombinationen Befehl in RFEM-Bibliothek

8.4.1.4 Entwicklung von FlächenLager

In diesem Abschnitt wird die Definition der Flächenlager behandelt, die in RFEM – wie auch in der grafischen Benutzeroberfläche sichtbar – für die Auflagerung von Flächen erforderlich sind.

Flächenlager stellen die Verbindung zwischen dem Modell und dem Untergrund dar und sind essenziell für eine realistische Abbildung der statischen Randbedingungen. Sie regeln, in welchen Richtungen eine Fläche gehalten oder verschieblich gelagert ist, z. B. in Z-Richtung (vertikale Lagerung) oder auch in den X- und Y-Richtungen bei horizontalen Lagerbedingungen.

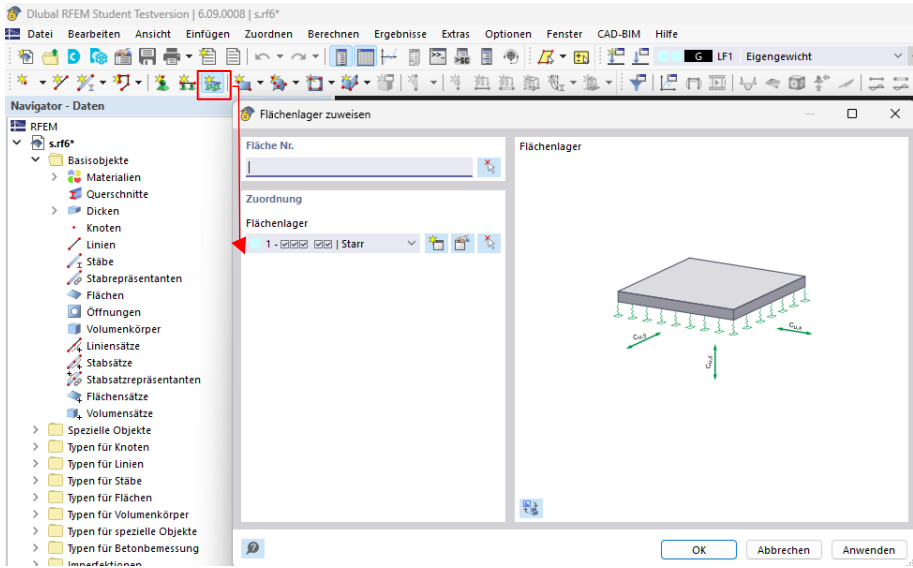


Abb.8.17: RFEM-Dlubal Flächenlager

Und jetzt um den Flächenlager einzufügen Um das Flächenlager in das Modell einzufügen, kann der folgende Befehl verwendet werden.

Dieser ermöglicht die scriptbasierte Definition der Lagerbedingungen für die ausgewählten Flächen – analog zur manuellen Eingabe in der Dlubal-Oberfläche.

Durch die Verwendung dieses Befehls wird das Flächenlager korrekt im Modell platziert und mit den gewünschten Lagerparametern versehen:

```
from RFEM.TypesForSurfaces.surfaceSupport import SurfaceSupport
```

Abb.8.18: Flächenlager Befehl in RFEM-Bibliothek

8.4.1.5 Entwicklung von Typen für Betonbemessung

In diesem Abschnitt sollen die relevanten Typen für die Betonbemessung entwickelt und im Skript abgebildet werden.

Dabei handelt es sich um drei zentrale Kategorien, die auch in der grafischen Benutzeroberfläche von RFEM dargestellt werden:

Betondauerhaftigkeiten, Bewehrungsrichtungen und Flächenbewehrung.

Diese Einstellungen sind essenziell, um eine normgerechte und realistische Bemessung von Stahlbetonbauteilen durchzuführen.

Ziel ist es, die entsprechenden Befehle zu identifizieren und zu integrieren, mit denen diese drei Betonbemessungstypen scriptbasiert definiert werden können.

Im Folgenden soll daher untersucht werden, welche Funktionen und Befehle notwendig sind, um diese Bemessungstypen im Modell automatisiert zu erstellen.

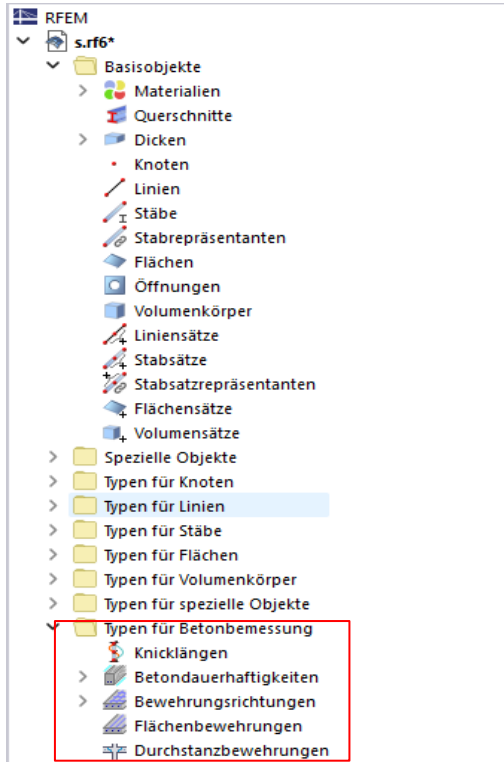


Abb.8.19: RFEM-Dlubal Typen für Betonbemessung

Für die Betondauerhaftigkeit muss eine entsprechende Einstellung im Modell vorgenommen werden. Diese legt die Expositionsklassen sowie zulässige Abweichungen und Randbedingungen fest, die für die Bemessung relevant sind. Die Definition erfolgt scriptbasiert mit folgendem Befehl, der die Dauerhaftigkeitsparameter gemäß den normativen Vorgaben im Modell verankert:

```
from RFEM.TypesforConcreteDesign.ConcreteDurability import ConcreteDurability
```

Abb.8.20: Betondauerhaftigkeit Befehl in RFEM-Bibliothek

Für die Bewehrungsrichtung muss ebenfalls eine gezielte Einstellung im Modell vorgenommen werden. Diese legt fest, in welcher Richtung die Hauptbewehrung innerhalb der Fläche verlaufen soll – z. B. entlang der X- oder Y-Achse oder angepasst an die Hauptspannungsrichtung. Die Definition erfolgt scriptbasiert mit folgendem Befehl,

der die Ausrichtung der Bewehrung gemäß den Bemessungsvorgaben im Modell verankert:

```
from RFEM.TypesforConcreteDesign.ConcreteReinforcementDirections import ConcreteReinforcementDirection
```

Abb.8.21: Bewehrungsrichtung Befehl in RFEM-Bibliothek

Für die Flächenbewehrung muss eine detaillierte Vorgabe im Modell definiert werden. Diese Einstellung bestimmt die Bewehrungsart, ihre Lage (oben oder unten) sowie den konkreten Bewehrungstyp (z. B. Stabstahl, Gittermatten). Die Definition erfolgt scriptbasiert mit folgendem Befehl, der die gewünschte Flächenbewehrung normgerecht im Modell zuweist:

```
from RFEM.TypesforConcreteDesign.ConcreteSurfaceReinforcements import ConcreteSurfaceReinforcements
```

Abb.8.22: Flächenbewehrung Befehl in RFEM-Bibliothek

8.4.1.6 Entwicklung von Globalparameters

In diesem Abschnitt müssen die Globalparameter definiert werden, die im weiteren Verlauf des Skripts für die Modellierung, Bemessung und Optimierung verwendet werden. Die Definition dieser Parameter erfolgt analog zur Darstellung in der grafischen Benutzeroberfläche von Dlubal RFEM, wie im zugehörigen Bild ersichtlich. Globalparameter ermöglichen eine flexible und zentral gesteuerte Eingabe von Werten wie Plattendicke, Bewehrungsquerschnitten oder Materialeigenschaften, die an mehreren Stellen im Modell referenziert werden können. Dadurch lässt sich das Modell effizienter strukturieren, parametrisieren und für Optimierungsprozesse vorbereiten.

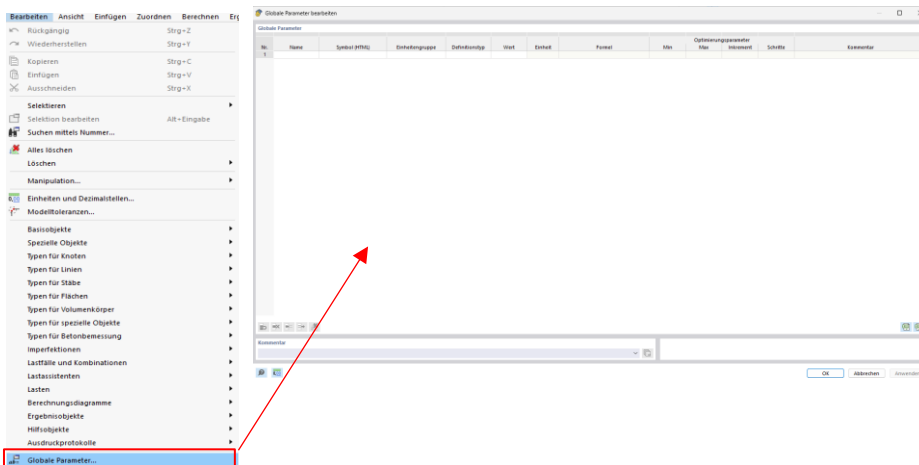


Abb.8.23: RFEM-Dlubal Typen für Globalparameter

Für die Globalparameter muss eine gezielte Definition im Modell vorgenommen werden. Diese Parameter ermöglichen es, zentrale Werte – wie z. B. Bauteildicken, Bewehrungsflächen oder CO₂-Faktoren – an einer Stelle zu definieren und im gesamten

Modell flexibel zu referenzieren. Dadurch wird das Skript übersichtlicher, leichter wartbar und optimal auf Optimierungsprozesse vorbereitet. Die Definition erfolgt scriptbasiert mit folgendem Befehl, der den jeweiligen Parameter normgerecht und modellweit verfügbar macht:

```
from RFEM.globalParameter import GlobalParameter
```

Abb.8.24: Globalparamets Befehl in RFEM-Bibliothek

8.4.1.7 Entwicklung von Betonbemessung

In diesem Abschnitt wird die Konfiguration für die Tragfähigkeitsbemessung (ULS) scriptbasiert definiert. Dazu wird die Funktion ConcreteUltimateConfiguration verwendet, mit der die Bemessung nach dem Grenzzustand der Tragfähigkeit gemäß den Normvorgaben eingerichtet wird. Diese Einstellung ist notwendig, um die ULS-Nachweise im Modell durchzuführen und entspricht der Konfiguration in der grafischen Benutzeroberfläche von RFEM.

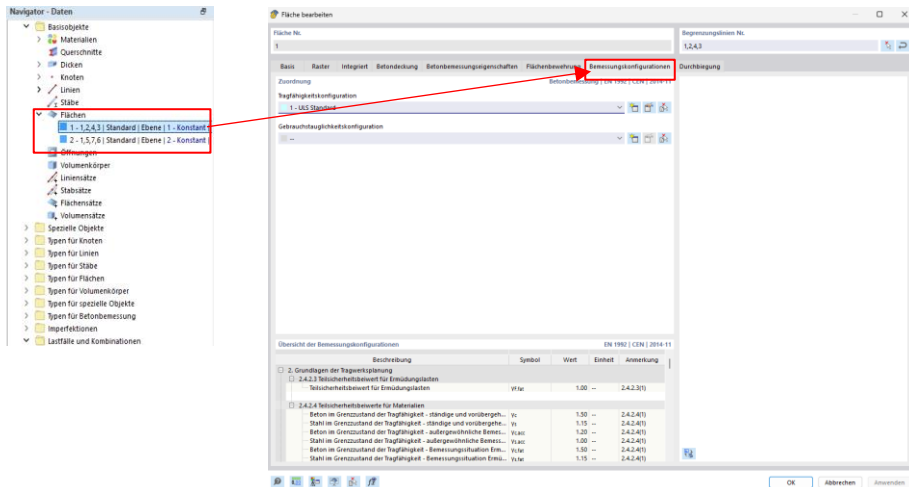


Abb.8.23: RFEM-Dlupal Bemessungskonfigurationen

Für die Tragfähigkeitsbemessung (ULS) muss eine gezielte Konfiguration im Modell vorgenommen werden. Diese Einstellung legt fest, dass die Nachweise nach dem Grenzzustand der Tragfähigkeit durchgeführt werden und welche Bauteile (z. B. Flächen oder Stäbe) darin enthalten sind.

Die Definition erfolgt scriptbasiert mit folgendem Befehl, der die entsprechende Bemessungskonfiguration normgerecht im Modell aktiviert und verfügbar macht:

```
from RFEM.ConcreteDesign.ConcreteUltimateConfigurations import ConcreteUltimateConfiguration
```

Abb.8.24: ConcreteUltimateConfiguration Befehl in RFEM-Bibliothek

8.4.1.8 Entwicklung von Optimierungseinstellungen

In diesem Abschnitt muss ein Optimierungsfall definiert werden, der eine zentrale Rolle im weiteren Ablauf spielt. Diese Definition ist besonders wichtig, um im späteren Verlauf eine CO₂-basierte Bewertung und Minimierung durchzuführen – wie auch in der Benutzeroberfläche von RFEM ersichtlich. Durch die Einrichtung eines Optimierungsfalls kann gezielt gesteuert werden, welche Parameter beeinflusst werden dürfen und welches Ziel, z. B. die Minimierung der CO₂-Emissionen, verfolgt wird.

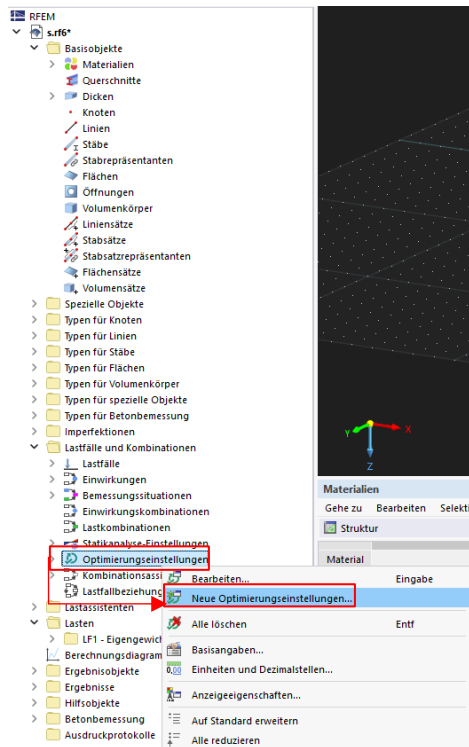


Abb.8.25: RFEM-Dlubal Optimierungseinstellungen, Neue

Um einen neuen Optimierungsfall in das Modell einzufügen, kann der folgende Befehl verwendet werden.

Damit wird die Optimierung im Skript aktiviert und entsprechend konfiguriert, sodass z. B. eine CO₂-Minimierung als Zielsetzung berücksichtigt werden kann:

```
from RFEM.Calculate.optimizationSettings import OptimizerType, OptimizationTargetValueType, OptimizationSettings
```

Abb.8.26: Optimierungseinstellungen Befehl in RFEM-Bibliothek

8.4.1.9 Entwicklung von Enums Bibliothek

In diesem Abschnitt müssen alle Funktionen definiert werden, die in den Abschnitten 8.4.1.1 bis 8.4.1.7 beschrieben wurden.

Diese Funktionen bilden die Grundlage für die automatisierte Erstellung und Berechnung des RFEM-Modells und umfassen unter anderem die Definition von Lastfällen, Bemessungssituationen, Bewehrungstypen, Optimierungseinstellungen und Globalparametern.

Eine zentrale Rolle spielt dabei die Enums-Bibliothek, da sie sämtliche notwendigen Konstanten und Typdefinitionen bereitstellt, die für die richtige Zuweisung von Eigenschaften innerhalb des Modells erforderlich sind.

Dazu zählen unter anderem Lagerarten, Bewehrungsrichtungen, Materialdefinitionen, Bemessungskonfigurationen sowie Zielgrößen für Optimierungen.

Ohne die Einbindung dieser Enums wäre es nicht möglich, die einzelnen Modellfunktionen eindeutig und normgerecht zu parametrisieren – was wiederum zu Fehlermeldungen oder fehlerhaften Berechnungsergebnissen führen würde.

Die korrekte Definition und Verwendung dieser Elemente ist daher eine zwingende Voraussetzung für die erfolgreiche Durchführung aller nachfolgenden Berechnungs-, Bemessungs- und Optimierungsschritte im automatisierten RFEM-Skript.

```
from RFEM.enums import (  
    NodalSupportType, SurfaceLoadDirection,  
    SurfaceLoadDistribution, ActionCategoryType, LineSupportType,  
    MemberLoadDirection, MemberLoadDistribution, MemberType,  
    CaseObjectType, NodalLoadDirection, AddOn, DesignSituationType, SurfaceReinforcementLocationType,  
    SurfaceReinforcementType, SurfaceReinforcementDirectionType, SurfaceReinforcementDesignDirection,  
    DurabilityStructuralClassType, DurabilityAllowanceDeviationType, ReinforcementDirectionType, OptimizerType,  
    OptimizationTargetValueType, GlobalParameterUnitGroup, GlobalParameterDefinitionType, ObjectTypes, ActionCategoryType, ActionType, ObjectTypes, MaterialDefinitionType,  
    MaterialType, MaterialModel, MaterialStiffnessModificationType, PoissonRatioEditableGroupType,  
)
```

Abb.8.27: Enums Befehl in RFEM-Bibliothek

8.4.1.10 Zusammengefasste alle Befehle in Hauptskript

```

from RFEM.iniModel import Model, calculate_all, insertSpaces, SetAddonStatus, clearAttributes, deleteEmptyAttributes, ConvertToDString, ConvertToDString
from RFEM.enums import C
Model.supportType, SurfaceLoadDirection,
SurfaceLoadDistribution, ActionCategoryType, LineSupportType,
MemberLoadDirection, MemberLoadDistribution, MemberType,
CaseObjectType, ModelLoadDirection, Addon, DesignSituationType, SurfaceReinforcementLocationType,
SurfaceReinforcementType, SurfaceReinforcementDirectionType, SurfaceReinforcementDesignDirection,
DurabilityStructuralClassType, DurabilityAllowanceCoefficientType, ReinforcementDirectionType, OptimizerType,
OptimizationTargetValueType, GlobalParameterUnitGroup, GlobalParameterDefinitionType, ObjectTypes, ActionCategoryType, ActionType, ObjectTypes, MaterialDefinitionType,
MaterialType, MaterialModel, MaterialStiffnessModificationType, PoissonRatioEtableGroupType,

from RFEM.BasicObjects.node import Node
from RFEM.BasicObjects.line import Line
from RFEM.BasicObjects.material import Material
from RFEM.BasicObjects.thickness import Thickness
from RFEM.BasicObjects.surface import Surface
from RFEM.BasicObjects.member import Member
from RFEM.BasicObjects.section import Section
from RFEM.TypesForLines.lineSupport import LineSupport
from RFEM.TypesForNodes.nodalSupport import NodalSupport
from RFEM.LoadCasesAndCombinations.loadCase import LoadCase
from RFEM.LoadCasesAndCombinations.staticAnalysisSettings import StaticAnalysisSettings
from RFEM.LoadCasesAndCombinations.loadCasesAndCombinations import LoadCasesAndCombinations
from RFEM.Loads.surfaceLoad import SurfaceLoad
from RFEM.Loads.memberLoad import MemberLoad
from RFEM.Loads.lineLoad import LineLoad
from RFEM.Loads.nodalLoad import NodalLoad
from RFEM.LoadCasesAndCombinations.loadCombination import LoadCombination
from RFEM.LoadCasesAndCombinations.designSituation import DesignSituation
from RFEM.TypesForSurfaces.surfaceSupport import SurfaceSupport
from RFEM.TypesForConcreteDesign.concreteSurfaceReinforcements import ConcreteSurfaceReinforcements
from RFEM.TypesForConcreteDesign.concreteDurability import ConcreteDurability
from RFEM.TypesForConcreteDesign.concreteReinforcementDirections import ConcreteReinforcementDirection
from RFEM.GlobalParameter import GlobalParameter
from RFEM.ConcreteDesign.concreteUltimateConfigurations import ConcreteUltimateConfiguration
from RFEM.Calculate.optimizationSettings import OptimizerType, OptimizationTargetValueType, OptimizationSettings
from RFEM.ImportExport.exports import ExportTo
from RFEM.Results.resultTables import ResultTables, GetMaxValue, GetMinValue, GetResultTableParameters
from RFEM.dataTypes import inf
Model(True, "MeinTestModell")
SetAddonStatus(Model.clientModel,addon=Addon.modal_active, status=True)
SetAddonStatus(Model.clientModel,addon=Addon.conc_optimization_active, status=True)
SetAddonStatus(Model.clientModel,addon=Addon.concrete_design_active, status=True)
Model.clientModel.service.delete_all() # Alles löschen!
Model.clientModel.service.begin_modification()
    
```

In diesem Abschnitt werden die Add-Ons für Betonbemessung und CO₂-Bewertung aktiviert, um normgerechte Nachweise und eine nachhaltige Optimierung zu ermöglichen. Beide Funktionen sind essenziell für die automatisierte Modellberechnung in RFEM und werden skriptbasiert durch gezielte Befehle eingebunden. (Siehe Abschnitt 8.4.1.1)

Die Tragfähigkeitsbemessung (ULS) wird skriptbasiert mit ConcreteUltimateConfiguration definiert, um normgerechte Nachweise im Modell zu ermöglichen. (Siehe Abschnitt 8.4.1.6)

Globalparameter werden skriptbasiert definiert, um zentrale Werte wie Dicke oder Bewehrung flexibel und modellweit für Berechnung und Optimierung nutzbar zu machen. (Siehe Abschnitt 8.4.1.5)

In diesem Abschnitt werden die Betonbemessungstypen Betondauerhaftigkeit, Bewehrungsrichtung und Flächenbewehrung skriptbasiert definiert, um eine normgerechte und automatisierte Bemessung im Modell zu ermöglichen. (Siehe Abschnitt 8.4.1.4)

Flächenlager werden skriptbasiert definiert, um die Lagerbedingungen der Flächen realitätsnah gemäß den statischen Randbedingungen im Modell festzulegen. (Siehe Abschnitt 8.4.1.3)

Lastkombination und Bemessungssituation werden skriptbasiert definiert, um maßgebende Einwirkungen normgerecht zu kombinieren und für die Betonbemessung sowie CO₂-Optimierung nutzbar zu machen. (Siehe Abschnitt 8.4.1.2)

Diese Importe laden die grundlegenden Objekte wie Knoten, Linien, Materialien, Flächen, Stäbe und Lager für die Modellierung in RFEM.

Dieser Import lädt alle benötigten Enums und Konstanten, um z. B. Lagerarten, Lastrichtungen, Materialtypen, Optimierungsziele und Bemessungseinstellungen in RFEM eindeutig und normgerecht zu definieren.

Abb.8.28: Zusammenfassung der zentralen Struktur und Funktionalität des Hauptskripts

8.4.2 Entwicklung des Skripts zur Dateneingabe

In diesem Abschnitt soll das bestehende Eingabeskript innerhalb der RFEM-Bibliothek weiterentwickelt und strukturell erweitert werden.

Bisher wurde das Skript bis zum Bereich „Loads“ umgesetzt. Ziel ist es nun, den Funktionsumfang auszubauen und alle zusätzlichen Funktionen abzubilden, die in Abschnitt 8.4.1 beschrieben wurden.

Dafür sollen entsprechende Bausteine als Templates erstellt und als Eingabeelemente implementiert werden, sodass sie automatisiert und scriptgesteuert in das Modell eingebunden werden können.

Die folgende Abbildung veranschaulicht die grundlegende Struktur des Dateneingabeskripts innerhalb der RFEM-Bibliothek.

Aufbauend darauf werden im nächsten Schritt die nachfolgenden Funktionen ergänzt:

1. Definition der Bemessungssituation (DesignSituation)
2. Eingabe der Betondauerhaftigkeit (ConcreteDurability)
3. Festlegung der Bewehrungsrichtung (ConcreteReinforcementDirection)
4. Definition der Flächenbewehrung (ConcreteSurfaceReinforcements)
5. Konfiguration der Tragfähigkeitsnachweise (ConcreteUltimateConfiguration)
6. Zentrale Verwaltung von Globalparametern (GlobalParameters)
7. Festlegung von Optimierungszielen und -werten

Diese Erweiterung ermöglicht eine vollständig scriptbasierte Steuerung der Modellierung, Bemessung und Optimierung und bildet somit die Grundlage für eine flexible und wiederverwendbare Eingabesystematik.

Ziel dieser Arbeit ist es, für jede Funktion ein eigenes Template zu entwickeln, dieses im RFEM-Prompt zu definieren und anschließend eine Eingabezelle bereitzustellen, in der der jeweilige Code in Python-Sprache eingegeben werden kann.



```

Allgemeiner PROMPT

MATERIAL

:RFEM_BasicObject_Material_Material at 802551865438

GEOMETRY

:RFEM_TypeOfForm_Line_LineSupport_LineSupport at 8025518657798

CROSSSECTION

LOADS

:RFEM_LoadCaseAndCombination_LoadCase_LoadCase at 8025518651798

SOLVER AUFRUFEN

P ..... solver_start .....
Model<class Model>.service_Finish_optimization()
Calculate_all()

result_solved_prompt_calculation_result at 80255186586608

ERGEBNISSE
    
```

Alle genannten Funktionen stehen einsatzbereit zur Verfügung.

Abb.8.29: Dateneingabe in RFEM- Bibliothek

8.4.2.1 Entwicklung des Templates für DesignSituationsen

Im Rahmen der Entwicklung der DesignSituation müssen zwei Schritte umgesetzt werden: Zunächst ist ein entsprechendes Template zu erstellen, das die Struktur der Eingabedaten vorgibt. Anschließend muss dieses Template im Prompt-Skript verankert werden, sodass eine Eingabemaske bereitgestellt wird, in der die erforderlichen Daten als Python-Datei eingegeben und verarbeitet werden können.

➤ Template zur Dateneingabe erstellen

In diesem Abschnitt muss die Syntaxstruktur der Funktion DesignSituation definiert und dokumentiert werden. Zusätzlich ist ein konkretes Beispiel bereitzustellen, das der KI als Vorlage dient, um zu erkennen, wie diese Funktion korrekt abgebildet und interpretiert werden soll.

The image shows a file explorer on the left with a 'New File...' menu open. A red arrow points from the 'New File...' option to a text box that says: 'Durch einen Klick auf „New File“ kann an dieser Stelle ein Ordner bzw. eine Datei im .txt-Format eingefügt werden, was für den weiteren Ablauf von großer Bedeutung ist.'

Below this, a code editor shows the Python code for the DesignSituation function. A red arrow points from the function signature to a text box: 'Dies ist die Syntaxstruktur der Funktion DesignSituation'. Another red arrow points from the function body to a text box: 'Beispiel für eine DesignSituation'. The code includes comments and a detailed example of a DesignSituation section with parameters like 'no = 1', 'design_situation_type', 'active = True', and 'name'.

Abb.8.30: DesignSituation Erstellung Schritte in RFEM- Bibliothek

➤ Eingabestruktur als Datei-Template definieren

An dieser Stelle sollen zwei Elemente erstellt werden: eine Eingabezelle zur Namensvergabe und eine weitere Zelle zur Eingabe des zugehörigen Codes.

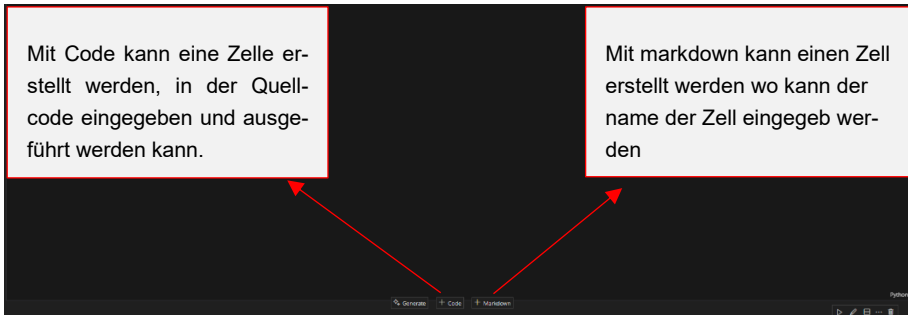


Abb.8.31: Eingabezelle zur Definition der DesignSituation in der RFEM-Bibliothek

8.4.2.2 Entwicklung des Templates für Betondauhaftigkeiten

Im Rahmen der Entwicklung der ConcreteDurability-Funktion müssen ebenfalls zwei Schritte umgesetzt werden: Zunächst ist ein passendes Template zu erstellen, das die Struktur der erforderlichen Eingabedaten vorgibt.

Anschließend muss dieses Template im Prompt-Skript eingebunden werden, sodass eine Eingabemaske zur Verfügung steht, über die die notwendigen Informationen in Form einer Python-Datei eingegeben und verarbeitet werden können.

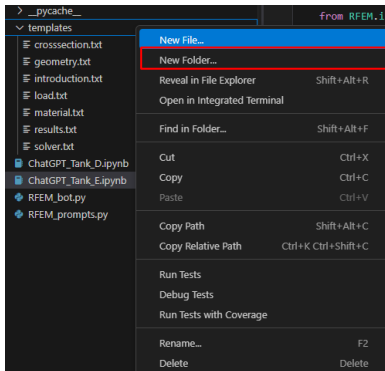
➤ Template zur Dateneingabe erstellen – ConcreteDurability

In diesem Abschnitt soll die Funktion ConcreteDurability im Detail betrachtet werden.

Ziel ist es, die Syntaxstruktur dieser Funktion vollständig zu definieren und zu dokumentieren, um eine normgerechte und automatisierte Eingabe im Rahmen der Betonbemessung zu ermöglichen. Dabei werden insbesondere die relevanten Parameter – wie z. B. Expositionsklassen, Bauteilzuordnung und zulässige Abweichungen – strukturiert erfasst.

Zusätzlich wird ein konkretes Beispiel bereitgestellt, das der KI als Vorlage dient, um zu erkennen, wie diese Funktion im Skript korrekt aufgebaut ist und welche Werte in welchem Format erwartet werden. Dieses Beispiel bildet die Grundlage für die spätere Generierung oder Verarbeitung entsprechender Eingabedateien im Rahmen des automatisierten Planungs- und Bemessungsprozesses.

Exemplarische Darstellung und methodisches Vorgehen



Durch einen Klick auf „New File“ kann an dieser Stelle ein Ordner bzw. eine Datei im .txt-Format eingefügt werden, was für den weiteren Ablauf von großer Bedeutung ist.

Dies ist die Syntaxstruktur der Funktion ConcreteDurability

```

1 Now define the Load section. [placeholder]
2 This is the syntax of the ConcreteDurability:
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Beispiel für ConcreteDurability

Mit Code kann eine Zelle erstellt werden, in der Quellcode eingegeben und ausgeführt werden kann.

Mit markdown kann einen Zell erstellt werden wo kann der name der Zell eingegeb werden

Abb.8.32: Eingabezelle zur Definition der Betondauerhaftigkeiten in der RFEM-Bibliothek

8.4.2.3 Entwicklung des Templates für Bewehrungsrichtung

Im Rahmen der Entwicklung der ConcreteReinforcementDirection-Funktion müssen ebenfalls zwei Schritte umgesetzt werden:

Zunächst ist ein passendes Template zu erstellen, das die Struktur der erforderlichen Eingabedaten vorgibt.

Anschließend muss dieses Template im Prompt-Skript eingebunden werden, sodass eine Eingabemaske zur Verfügung steht, über die die notwendigen Informationen in Form einer Python-Datei eingegeben und verarbeitet werden können.

➤ Template zur Dateneingabe erstellen – ConcreteReinforcementDirection

In diesem Abschnitt wird die Funktion ConcreteReinforcementDirection detailliert beschrieben. Ziel ist es, die Syntaxstruktur vollständig zu definieren und zu dokumentieren, um die automatisierte Festlegung der Bewehrungsrichtung im Rahmen der Betonbemessung zu ermöglichen. Dabei werden Parameter wie Richtungsverlauf (X-/Y-Achse), Orientierung zur Hauptspannung und die Bemessungsrichtung strukturiert erfasst. Ein konkretes Beispiel zeigt der KI, wie die Funktion korrekt aufgebaut und interpretiert werden soll und dient als Grundlage für die spätere automatische Verarbeitung von Eingabedateien.

Durch einen Klick auf „New File“ kann an dieser Stelle ein Ordner bzw. eine Datei im .txt-Format eingefügt werden, was für den weiteren Ablauf von großer Bedeutung ist.

Dies ist die Syntaxstruktur der Funktion ConcreteReinforcementDirection

```

1 New define the load section. (placeholder)
2 This is the syntax of the ConcreteReinforcementDirection:
3
4
5 ConcreteReinforcementDirection(
6     no=1,
7     name= "RD 1",
8     surfaces= "1",
9     reinforcement_direction_type = ReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_FIRST_REINFORCEMENT_IN_Y,
10    rotation_parameters: list = None,
11    comment= str = "",
12    params= dict = None,
13    model= Model)
14
15
16
17 This is an example of the ConcreteReinforcementDirection section:
18
19 # ---- ConcreteReinforcementDirection Sec ----
20
21 # ---- ConcreteReinforcementDirection Section ----
22
23 ConcreteReinforcementDirection(
24     no=1,
25     name= "RD 1",
26     surfaces= "1 2", # Surfaces 1 and 2
27     reinforcement_direction_type=ReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_FIRST_REINFORCEMENT_IN_Y, # Reinforcement in Y-direction
28     rotation_parameters=[False, 0.0], # No manual rotation angle
29     comment="Bewehrung in Y-Richtung für Fläche 1 und 2", # Comment for the reinforcement direction
30     params=None, # Additional parameters if needed
31     model=Model) # The model to which this reinforcement direction belongs
32
33
    
```

Beispiel für ConcreteReinforcementDirection

Abb.8.34: Eingabemaske zur Definition der ConcreteReinforcementDirection

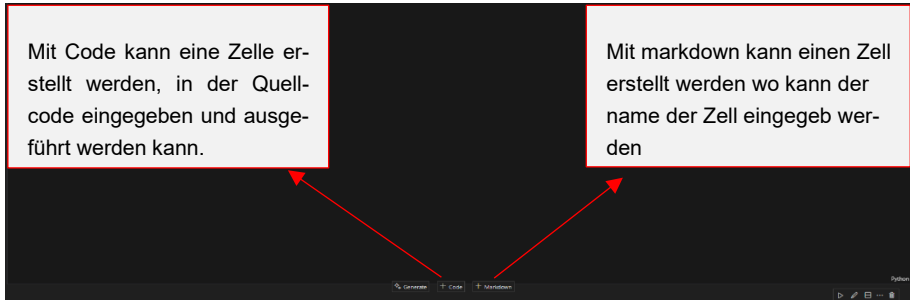


Abb.8.35: Eingabezelle zur Definition der ConcreteReinforcementDirection

8.4.2.4 Entwicklung des Templates für Flächenbewehrung

Für die Funktion ConcreteSurfaceReinforcements sind zwei Schritte erforderlich:

Zuerst wird ein Template zur Eingabestruktur erstellt, anschließend im Prompt-Skript eingebunden, um die Eingabe als Python-Datei zu ermöglichen.

- Template zur Dateneingabe erstellen – ConcreteSurfaceReinforcements

In diesem Abschnitt wird die Funktion ConcreteSurfaceReinforcements beschrieben.

Ziel ist die Definition der Syntax zur automatisierten Eingabe von Flächenbewehrung, einschließlich Lage, Typ und Richtung der Bewehrung.

Durch einen Klick auf „New File“ kann an dieser Stelle ein Ordner bzw. eine Datei im .txt-Format eingefügt werden, was für den weiteren Ablauf von großer Bedeutung ist.

Dies ist die Syntaxstruktur der Funktion Flächenbewehrung

Beispiel für Flächenbewehrung

```

Now define the load section. (placeholder)
This is the syntax of the ConcreteSurfaceReinforcements:

ConcreteSurfaceReinforcements(
    name: str = "RD 1",
    surfaces: list = [],
    location_type = SurfaceReinforcementLocationType.LOCATION_TYPE_ON_SURFACE,
    reinforcement_type = SurfaceReinforcementType.REINFORCEMENT_TYPE_BESB,
    reinforcement_type_parameters: list = [0.01, 0.15, False],
    cover_offset: list = [True, True, 0, 0],
    reinforcement_direction = SurfaceReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_IN_DESIGN_REINFORCEMENT_DIRECTION,
    reinforcement_direction_parameters: list = [SurfaceReinforcementDesignDirection.DESIGN_REINFORCEMENT_DIRECTION_A_5_1],
    reinforcement_location_list = None,
    reinforcement_acting_region: list = None,
    comment: str = "",
    params: dict = None,
    model = None)

This is an example of the ConcreteSurfaceReinforcements section:

---- ConcreteSurfaceReinforcements Sec ----

ConcreteSurfaceReinforcements(
    name: "OBERBEMEHRUNG",
    surfaces: [1, 2], # Surfaces 1 and 2
    material: "2", # Concrete material Number 2
    location_type: SurfaceReinforcementLocationType.LOCATION_TYPE_ON_SURFACE, # Reinforcement on the surface
    reinforcement_type: SurfaceReinforcementType.REINFORCEMENT_TYPE_BESB, # Rebar type reinforcement
    reinforcement_type_parameters: [None, 0.15, False], # If no diameter, 15 cm spacing, no stirrups
    cover_offset: [True, True, 0.03, 0.03], # Cover offset 3 cm top and bottom
    reinforcement_direction: SurfaceReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_IN_DESIGN_REINFORCEMENT_DIRECTION, # In design direction A5
    reinforcement_direction_parameters: [SurfaceReinforcementDesignDirection.DESIGN_REINFORCEMENT_DIRECTION_A_5_1],
    comment: "User's comment (Block 1)", # Comment for the reinforcement
    params: None, # Additional parameters if needed
    model: None) # The model to which this reinforcement belongs
    
```

Abb.8.36: Eingabezeile zur Definition der Flächenbewehrung

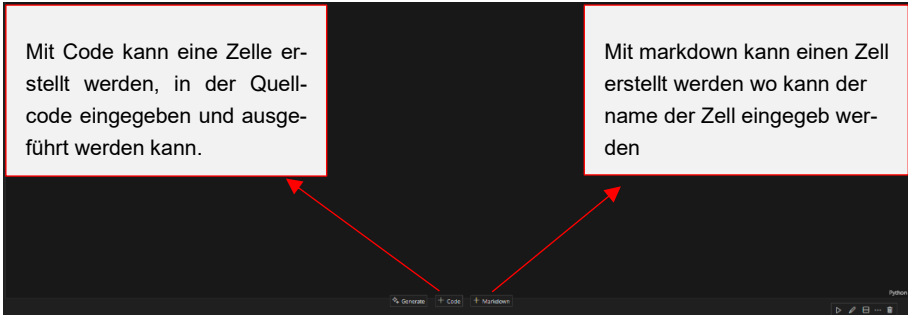


Abb.8.37: Eingabezeile zur Definition der Flächenbewehrung

8.4.2.5 Entwicklung des Templates für Beton Bemessung

Für die Funktion ConcreteUltimateConfiguration sind zwei Schritte erforderlich: Zuerst wird ein Template zur Eingabestruktur erstellt, anschließend im Prompt-Skript eingebunden, um die Eingabe als Python-Datei zu ermöglichen.

- Template zur Dateneingabe erstellen – ConcreteUltimateConfiguration

In diesem Abschnitt wird die Funktion ConcreteUltimateConfiguration beschrieben. Ziel ist die Definition der Syntax zur automatisierten Festlegung der Betonbemessung im Grenzzustand der Tragfähigkeit, einschließlich der Zuweisung zu Stäben und/oder Flächen.

Durch einen Klick auf „New File“ kann an dieser Stelle ein Ordner bzw. eine Datei im .txt-Format eingefügt werden, was für den weiteren Ablauf von großer Bedeutung ist.

Dies ist die Syntaxstruktur der Funktion ConcreteUltimateConfiguration

Beispiel für ConcreteUltimateConfiguration

```

1 Now define the load section. (placeholder)
2 This is the syntax of the ConcreteUltimateConfiguration:
3
4
5
6 ConcreteUltimateConfiguration(
7     no: int = 1,
8     name: str = 'ULS',
9     members: str = 'All',
10    member_sets: str = 'All',
11    surfaces: str = 'All',
12    surface_sets: str = 'All',
13    nodes: str = '',
14    comment: str = '',
15    params: dict = None,
16    model = Model)
17
18
19 This is an example of the ConcreteUltimateConfiguration section:
20 #----- ConcreteUltimateConfiguration.Sec-----
21
22 ConcreteUltimateConfiguration(
23     no = 1,
24     name = 'ULS Standard',
25     members = '1 2 3', # oder 'All'
26     member_sets = 'All',
27     surfaces = '1 2',
28     surface_sets = '1 2',
29     nodes = '1, 2, 3, 4, 5, 6',
30     comment = 'Standard ULS für Platten und Träger',
31     model = Model
32 )
33
34
    
```

Abb.8.38: Eingabezeile zur Definition der ConcreteUltimateConfiguration



Abb.8.39: Eingabezelle zur Definition der ConcreteUltimateConf...

8.4.2.6 Entwicklung des Templates für Globalparameters

Für die Funktion GlobalParameter sind zwei Schritte erforderlich:

Zuerst wird ein Template zur Eingabestruktur erstellt, anschließend im Prompt-Skript eingebunden, um die Eingabe als Python-Datei zu ermöglichen.

- Template zur Dateneingabe erstellen – GlobalParameter

In diesem Abschnitt wird die Funktion GlobalParameter beschrieben.

Ziel ist die Definition der Syntax zur automatisierten Eingabe globaler Modellparameter, z. B. für Dicken, Bewehrungswerte oder CO₂-Faktoren, die an mehreren Stellen im Modell verwendet werden können.

Dies ist die Syntaxstruktur der Funktion GlobalParameter

Beispiel für GlobalParameter

```

OptimizationSettings(
    no: int = 1,
    number_of_mutations_to_keep: int = 20,
    target_value_type: enum = OptimizationTargetValueType.MIN_TOTAL_WEIGHT,
    optimizer_type: enum = OptimizerType.ALL_MUTATIONS,
    percent_of_mutations: float = 0.1,
    optimization_values_table = None,
    total_number_of_mutations: int = 40,
    active: bool = True,
    name: str = None,
    comment: str = '',
    params: dict = None,
    model = Model)

This is an example of the OptimizationSettings section:
# ---- OptimizationSettings Sec ----

OptimizationSettings(
    no=1,
    number_of_mutations_to_keep=None,
    target_value_type=OptimizationTargetValueType.MIN_CO2_EMISSIONS, # << CO2-Optimierung!
    optimizer_type=OptimizerType.PARTICLE_SWARM,
    percent_of_mutations=0.2,
    optimization_values_table = None,
    total_number_of_mutations=1,
    active=True,
    name="CO2-Optimierung",
    comment="Ziel: Minimale CO2-Emissionen",
    )
    
```

Abb.8.40: Eingabezelle zur Definition der GlobalParameter



Abb.8.41: Eingabezelle zur Definition der GlobalParameter

8.4.2.7 Entwicklung des Templates für Optimierungseinstellungen

Für die Funktion OptimizationSettings sind zwei Schritte erforderlich:

Zuerst wird ein Template zur Eingabestruktur erstellt, anschließend im Prompt-Skript eingebunden, um die Eingabe als Python-Datei zu ermöglichen.

- Template zur Dateneingabe erstellen – OptimizationSettings

In diesem Abschnitt wird die Funktion OptimizationSettings beschrieben.

Ziel ist die Definition der Syntax zur automatisierten Festlegung von Optimierungszielen, wie z. B. die Minimierung von CO₂-Emissionen, sowie die Konfiguration der entsprechenden Rechenoptionen.

Dies ist die Syntaxstruktur der Funktion OptimizationSettings

Beispiel für OptimizationSettings

Abb.8.42: Eingabezelle zur Definition der Optimierungseinstellungen

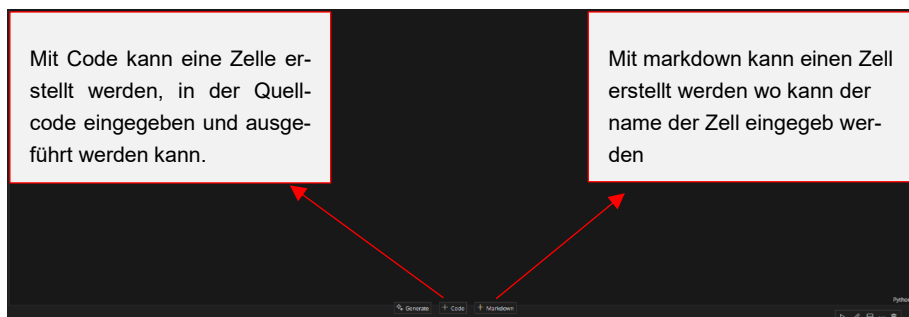


Abb.8.43: Eingabezelle zur Definition der Optimierungseinstellungen

8.4.3 Entwicklung von Propmt

In diesem Abschnitt soll ein Prompt entwickelt werden, der alle in Abschnitt 8.4.2 erstellten Templates integriert. Ziel ist es, der KI die Möglichkeit zu geben, gezielt auf die angefragte Funktion zuzugreifen und den dazugehörigen Python-Code automatisch zu generieren. Dafür sind zwei zentrale Bestandteile umzusetzen:

Erstens muss eine Definition für jedes Template erstellt werden, was mit folgendem Befehl realisiert werden kann:

```
template = read_file(f"templates/{function_name}.txt")
```

Abb.8.44: Befehl für Definition eines Templates

Zweitens ist ein passender „Keyword“-Mechanismus zu entwickeln, über den die KI erkennt, welches Template angesprochen werden soll. Auch dies kann mithilfe des folgenden Befehls umgesetzt werden:

```
'<Funktionsname>': <TemplateVariable>,
```

Abb.8.45: Befehl für Keyword eines Templates

Die folgende Darstellung zeigt den allgemeinen Aufbau des Prompts, in dem alle Funktionen mit ihren zugehörigen Templates und Keywords eingebunden sind.

Gleichzeitig veranschaulicht das Bild den Ablauf des Systems sowie die Verbindung zwischen den einzelnen Elementen, die zur automatisierten Ausführung der gewünschten Funktion beitragen.

```

1 # Module for using prompt templates to define RFEH 6 models
2
3 Tabnine | Edit | Test | Explain | Document
4 def read_file(file_path):
5     with open(file_path, "r", encoding="utf-8") as file:
6         return file.read()
7
8 INTRODUCTION_TPL = read_file("templates/introduction.txt")
9 MATERIAL_TPL = read_file("templates/Material.txt")
10 Geometry_TPL = read_file("templates/Geometry.txt")
11 CrossSection_TPL = read_file("templates/crosssection.txt")
12 LOAD_TPL = read_file("templates/load.txt")
13 SOLVER_TPL = read_file("templates/solver.txt")
14 RESULTS_TPL = read_file("templates/results.txt")
15 DesignSituation = read_file("templates/DesignSituation.txt")
16 ConcreteDurability = read_file("templates/ConcreteDurability.txt")
17 ConcreteReinforcementDirection = read_file("templates/ConcreteReinforcementDirection.txt")
18 ConcreteSurfaceReinforcements = read_file("templates/ConcreteSurfaceReinforcements.txt")
19 ConcreteUltimateConfiguration = read_file("templates/ConcreteUltimateConfiguration.txt")
20 GlobalParameter = read_file("templates/GlobalParameter.txt")
21 OptimizationSettings = read_file("templates/OptimizationSettings.txt")
22
23 keywords = {}
24 'INTRO': INTRODUCTION_TPL,
25 'MATERIAL': MATERIAL_TPL,
26 'GEOMETRY': Geometry_TPL,
27 'CROSSSECTION': CrossSection_TPL,
28 'LOADS': LOAD_TPL,
29 'SOLVER': SOLVER_TPL,
30 'RESULTS': RESULTS_TPL,
31 'DesignSituation': DesignSituation,
32 'ConcreteDurability': ConcreteDurability,
33 'ConcreteReinforcementDirection': ConcreteReinforcementDirection,
34 'ConcreteSurfaceReinforcements': ConcreteSurfaceReinforcements,
35 'ConcreteUltimateConfiguration': ConcreteUltimateConfiguration,
36 'GlobalParameter': GlobalParameter,
37 'OptimizationSettings': OptimizationSettings,
38
39
40
41
42 # Process prompt to use template if applicable
43 # The template keyword shall be specified at the beginning of the prompt with a semicolon
44 # Example:
45 # MATERIAL: My model...
46
47 Tabnine | Edit | Test | Explain | Document
48 def process_prompt(prompt):
49     if ':' not in prompt:
50         return prompt
51     key, value = prompt.split(':', 1)
52     if key not in keywords:
53         return prompt
54     return keywords[key].format(placeholder =value)

```

Abb.8.46: Basiscode des vollständigen Prompt-Skripts

8.5 Durchführung und Berechnung der Aufgabenstellung

In diesem Abschnitt wird ein vollständiges Beispiel durchgerechnet, wobei die Ergebnisse auf Grundlage des Basis-Codes ermittelt werden, der in den vorherigen Abschnitten dargestellt wurde.

8.5.1 Beschreibung der Ausgangsparameter und Abmessungen

Ein L-förmiges Betonfundament soll unter statischen Lasten hinsichtlich seiner CO₂-Emissionen und Wirtschaftlichkeit optimiert werden. Das Modell besteht aus zwei orthogonal zueinanderstehenden Platten (eine vertikale Wandplatte, eine horizontale Fundamentplatte), die gemeinsam ein geometrisch L-förmiges System bilden.

- **Geometrie und Randbedingungen:**

Höhe der vertikalen Platte: 1,00 m

Breite der horizontalen Platte: 0,75 m

Variable Dicke (Optimierungsparameter) für beide Platten

- **Materialien nach DIN EN 1992-1-1:**

Beton: C20/25

Bewehrungsstahl: B500A

Expositionsklasse: XC1

- **Lastansätze:**

Eigengewicht wirkt im gesamten System automatisch

Vertikale variable Flächenlast auf der Wandplatte: linear von 0 bis 50 kN/m^2 über die Höhe

Konstante Flächenlast auf der Fundamentplatte: 100 kN/m^2

- **Gesucht:**

Ermittlung einer optimalen Kombination aus Plattendicke [m], Bewehrungsgehalt A_s [cm^2/m] unter der Zielsetzung Minimierung der CO_2 -Emissionen unter Einhaltung der statischen Anforderungen

- **Skizze**

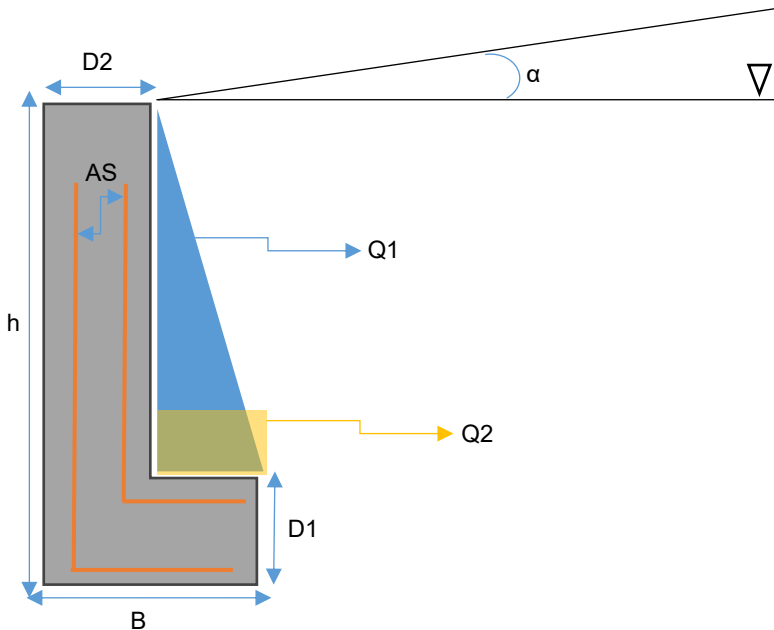


Abb.8.47: Skizze der Aufgabestellung

8.5.2 Abfrage der Parameter durch KI-generierten Python-Code

In diesem Abschnitt werden mithilfe von ChatGPT alle erforderlichen Parameter abgefragt. Die KI liefert daraufhin den passenden Python-Code, der anschließend an den jeweiligen Stellen im Skript gezielt eingefügt werden kann. Durch die Ausführung von gpt_bot.py können gezielt Fragen gestellt werden – wie in der Abbildung dargestellt.

Hinweis: Die Funktion muss im folgenden Format aufgerufen werden: „Funktion:“

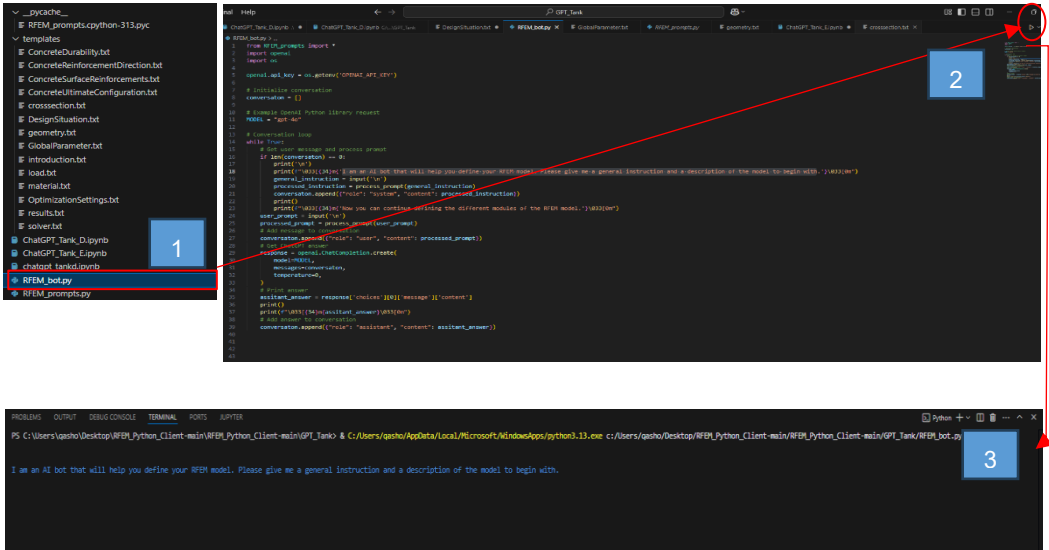


Abb.8.48: Chat-GPT Aktivieren

8.5.2.1 Allgemeiner Prompt und Funktionsbeschreibung

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Modul abgefragt, und zwar in folgender Form:

„Das Modell besteht aus L-Fundament: eine vertikale und eine horizontale, L-förmig verbunden. Es wirken Eigengewicht, eine variable Flächenlast auf der vertikalen und eine konstante Last auf der horizontalen Platte. Alle Elemente und Lasten sind positionsbasiert definiert“.

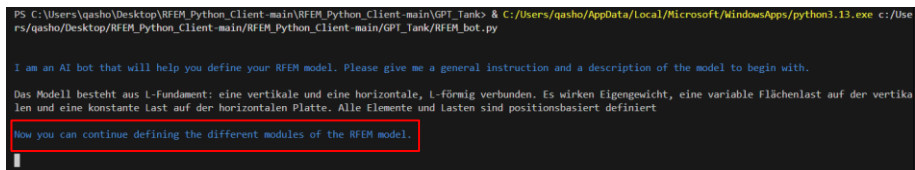


Abb.8.49: Allgemeiner Prompt für GPT eingeben

8.5.2.2 Materialabfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Material abgefragt, und zwar in folgender Form:

„MATERIAL: mit Deutschland Region und Norm nach erstell C20/25 und B550S(A)“

The image shows a GPT chat interface. At the top, the user asks: "MATERIAL: mit Deutschland Region und Norm nach erstell C20/25 und B550S(A)". Below this, the GPT response is shown, starting with "Based on the provided syntax and example, here is how you can define the Material section for the given requirements:". The response includes a code block for defining two materials in a Python-like syntax. A red box highlights this code block. A yellow callout box with a red arrow pointing to the code contains the text: "Der folgende Code muss kopiert und in die Materialeingabedatei eingefügt werden." Below the callout, the code is shown again in a larger font, with a red arrow pointing from the callout to it.

```
MATERIAL: mit Deutschland Region und Norm nach erstell C20/25 und B550S(A)

Based on the provided syntax and example, here is how you can define the Material section for the given requirements:

```python
---- Material Sec ----
Material(
 no=1,
 name='C20/25 | DIN EN 1992-1-1/NA/A1:2015-12',
 comment='C20/25 mit deutschem NA',
 model=Model
)

Material(
 no=2,
 name='B550S(A)',
 model=Model
)
```
```

Der folgende Code muss kopiert und in die Materialeingabedatei eingefügt werden.

```
MATERIAL

# ---- Material Sec ----
Material(
    no=1,
    name='C20/25 | DIN EN 1992-1-1/NA/A1:2015-12',
    comment='C20/25 mit deutschem NA',
    model=Model
)

Material(
    no=2,
    name='B550S(A)',
    model=Model
)
```

Abb.8.50: Verlauf Materialabfrage an GPT

8.5.2.3 Geometryabfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Geometry abgefragt, und zwar in folgender Form:

„GEOMETRY: Modellieren Sie ein L-förmiges Betonfundament bestehend aus zwei rechtwinklig verbundenen Platten: Eine vertikale Platte in der X-Z-Ebene, 0.75 m breit (X-Richtung) und 1.0 m hoch (nach unten in Z-Richtung), beginnend bei Z = 0. Eine horizontale Platte in der X-Y-Ebene, ebenfalls 0.75 m breit (X-Richtung) und 1.0 m lang (Y-Richtung), beginnend bei Y = 0. Beide Platten sind an ihrer gemeinsamen Kante zwischen Punkt (0, 0, 0) und (0.75, 0, 0) verbunden. Verwenden Sie keine doppelten Knoten oder Linien – die Verbindungslinie zwischen den Platten darf nur einmal definiert werden.“



Abb.8.51: Verlauf Geometryabfrage an GPT

8.5.2.4 Crosssectionabfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Crosssection abgefragt, und zwar in folgender Form:

„CROSSECTION: Erstelle Thickness(...), Surface(...) und SurfaceSupport(...) ausschließlich mit Positionsparametern. Verwende in Vertikale Richtung und Horizontal als Namen. Die Flächen sollen über Linien als String definiert werden. Verwende nicht SurfaceGeometry.“

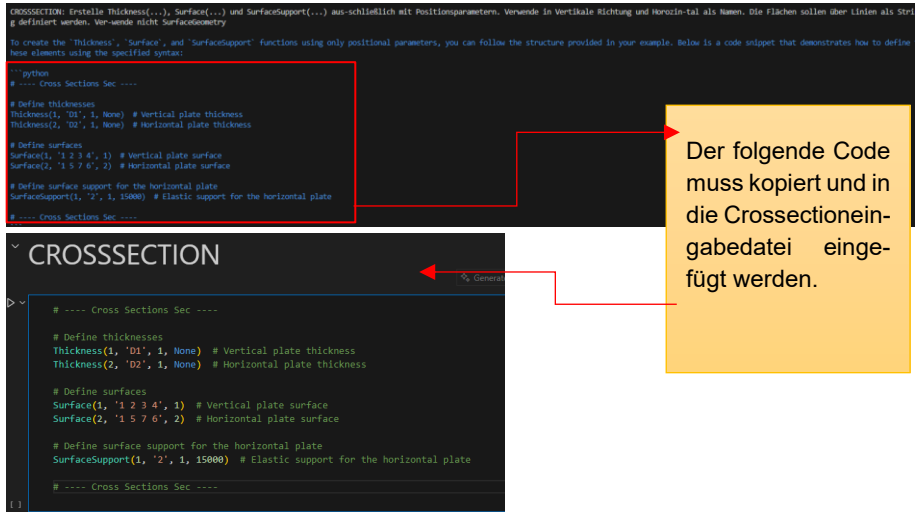


Abb.8.52: Verlauf Crosssectionabfrage an GPT

8.5.2.5 Loadsabfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Lasten abgefragt, und zwar in folgender Form:

„LOADS: Es sollen zwei Lastfälle erstellt werden: Lastfall 1 enthält das Eigengewicht und ist aktiviert. Lastfall 2 ist eine „Erddrucklast nach Befüllung“. In Lastfall 2 soll eine linear veränderliche Flächenlast in lokaler Z-Richtung auf Fläche 1 (vertikale Platte) definiert werden. Die Platte verläuft in negativer Z-Richtung von $Z = -1$ bis $Z = 0$. Die Last beginnt mit 0 kN/m^2 oben ($Z = -1$) und steigt auf 50 kN/m^2 unten ($Z = 0$). Zusätzlich erhält Fläche 2 (horizontale Platte) in Lastfall 2 eine konstante Flächenlast von 100 kN/m^2 in lokaler Z-Richtung. Beide Flächenlasten sind mit `SurfaceLoad.Force` einzugeben. Die Flächennummern sollen als Strings übergeben werden. Die Einheiten sind in N/m^2 umzuwandeln.“

LOADS: Es sollen zwei Lastfälle erstellt werden: Lastfall 1 enthält das Eigengewicht und ist aktiviert. Lastfall 2 ist eine „Erddrucklast nach Befüllung“. In Lastfall 2 soll eine linear veränderliche Flächenlast in lokaler Z-Richtung auf Fläche 1 (vertikale Platte) definiert werden. Die Platte verläuft in negativer Z-Richtung von $Z = -1$ bis $Z = 0$. Die Last beginnt mit 0 kN/m^2 oben ($Z = -1$) und steigt auf 50 kN/m^2 unten ($Z = 0$). Zusätzlich erhält Fläche 2 (horizontale Platte) in Lastfall 2 eine konstante Flächenlast von 100 kN/m^2 in lokaler Z-Richtung. Beide Flächenlasten sind mit `SurfaceLoad.Force` einzugeben. Die Flächennummern sollen als Strings übergeben werden. Die Einheiten sind in N/m^2 umzuwandeln.

Below is the code snippet for defining the load section according to your specifications. It includes two load cases: one for self-weight and another for earth pressure after filling. The surface loads are defined as follows. The appropriate parameters and units converted to N/m^2 .

```

python
# ---- Loads Sec ----
# ---- Lastfälle ----
loadCase(1, "Self-weight", [True, 0.0, 0.0, 1.0])
loadCase(2, "Erddrucklast nach Befüllung")
# ---- Loads Sec ----

# Define surface loads
# linear varying surface load on the vertical plate
SurfaceLoad.Force(
    1, # Load number
    2, # Load case number
    '1', # Surface number as string
    SurfaceLoadDirection.LOCAL_Z, # Load direction
    SurfaceLoadDistribution.LOCAL_DISTRIBUTION_VARYING_IN_Z, # Load distribution
    [-1, 0], [0, 50 * 1000] # Load parameters: start and end values in N/m^2
)

# Constant surface load on the horizontal plate
SurfaceLoad.Force(
    2, # Load number
    2, # Load case number
    '2', # Surface number as string
    SurfaceLoadDirection.LOCAL_Z, # Load direction
    SurfaceLoadDistribution.LOCAL_DISTRIBUTION_UNIFORM, # Load distribution
    [100 * 1000] # Load magnitude in N/m^2
)
    
```

Der folgende Code muss kopiert und in die Loadseingabedatei eingefügt werden.

```

LOADS

# ---- Loads Sec ----

# ---- Lastfälle ----
loadCase(1, "Self-weight", [True, 0.0, 0.0, 1.0])
loadCase(2, "Erddrucklast nach Befüllung")

# ---- Loads Sec ----

# Define surface loads
# linear varying surface load on the vertical plate
SurfaceLoad.Force(
    1, # Load number
    2, # Load case number
    '1', # Surface number as string
    SurfaceLoadDirection.LOCAL_Z, # Load direction
    SurfaceLoadDistribution.LOCAL_DISTRIBUTION_VARYING_IN_Z, # Load distribution
    [-1, 0], [0, 50 * 1000] # Load parameters: start and end values in N/m^2
)

# Constant surface load on the horizontal plate
SurfaceLoad.Force(
    2, # Load number
    2, # Load case number
    '2', # Surface number as string
    SurfaceLoadDirection.LOCAL_Z, # Load direction
    SurfaceLoadDistribution.LOCAL_DISTRIBUTION_UNIFORM, # Load distribution
    [100 * 1000] # Load magnitude in N/m^2
)
    
```

Abb.8.53: Verlauf Loadsabfrage an GPT

8.5.2.6 Disgnsituation Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Designsituatuion abgefragt, und zwar in folgender Form:

„DesignSituation: Erstelle eine GZT-Designsituation nach DIN EN 1990 und eine Lastkombination (ULS) mit einer statischen Analyse. In der Kombination sollen Lastfall 1 (Eigengewicht) m und Lastfall 2 (Nutzlast) berücksichtigt werden.“

DesignSituation: Erstelle eine GZT-Designsituation nach DIN EN 1990, und eine Lastkombination (ULS) mit einer statischen Analyse. In der Kombination sollen Lastfall 1 (Eigengewicht) m und Lastfall 2 (Nutzlast) berücksichtigt werden.

Below is the code snippet for defining the design situation and load combination according to your specifications. It includes a GZT design situation following DIN EN 1990 and a ULS load combination with a static analysis that considers both the self-weight and live load.

```
python
# ---- DesignSituation Sec ----
DesignSituation(
    no=1,
    design_situation_type=DesignSituationType.DESIGN_SITUATION_TYPE_STR_PERMANENT_AND_TRANSIENT_6_10,
    active=True,
    name="GZT Ständig und vorübergehend (6.10)",
    params=None
)

LoadCombination(
    no=1,
    analysis_type=AnalysisType.ANALYSIS_TYPE_STATIC,
    name="ULS Kombination",
    static_analysis_settings=1,
    to_solve=True,
    design_situation=1,
    combination_items=[
        [1.35, 1, 0, True], # Load factor for self-weight (Load Case 1)
        [1.5, 2, 0, True] # Load factor for live load (Load Case 2)
    ]
)

```

Der folgende Code muss kopiert und in die Designsituationeneingabedatei eingefügt werden.

Designsituation

```
# ---- DesignSituation Sec ----
DesignSituation(
    no=1,
    design_situation_type=DesignSituationType.DESIGN_SITUATION_TYPE_STR_PERMANENT_AND_TRANSIENT_6_10,
    active=True,
    name="GZT Ständig und vorübergehend (6.10)",
    params=None
)

LoadCombination(
    no=1,
    analysis_type=AnalysisType.ANALYSIS_TYPE_STATIC,
    name="ULS Kombination",
    static_analysis_settings=1,
    to_solve=True,
    design_situation=1,
    combination_items=[
        [1.35, 1, 0, True], # Load factor for self-weight (Load Case 1)
        [1.5, 2, 0, True] # Load factor for live load (Load Case 2)
    ]
)

```

Abb.8.54: Verlauf Disgnsituationabfrage an GPT

8.5.2.7 Betondauerhaftigkeiten Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Betondauerhaftigkeiten abgefragt, und zwar in folgender Form:

„ConcreteDurability: Lege eine Dauerhaftigkeitsdefinition für Betonbauteile an. Die Expositionsklasse soll XC1 sein. für beide Platten“



Abb.8.55: Verlauf Betondauerhaftigkeitenabfrage an GPT

8.5.2.8 Bewehrungsrichtung Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Bewehrungsrichtung abgefragt, und zwar in folgender Form:

„ConcreteReinforcementDirection: Erzeuge eine Bewehrungsrichtung für beide Platten Die Bewehrung soll in Y-Richtung verlaufen, ohne Rotationswinkel.“

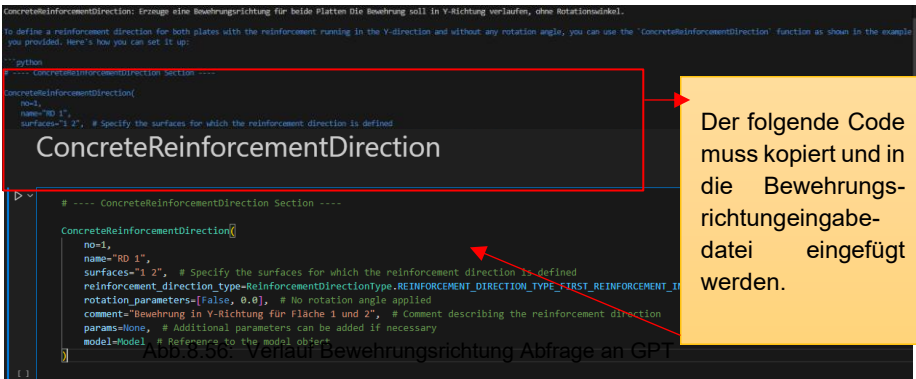


Abb.8.56: Verlauf Bewehrungsrichtung Abfrage an GPT

8.5.2.9 Flächenbewehrung Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Flächenbewehrung abgefragt, und zwar in folgender Form:

„ConcreteSurfaceReinforcements: Erzeuge eine obere Flächenbewehrung mit dem Namen „OBERBEWEHRUNG“ auf den Flächen 1 und 2. Verwende Betonmaterial Nummer 2. Die Bewehrung soll auf der Oberfläche liegen, ohne Bügel. Der Betondeckungsrandabstand soll oben und unten jeweils 3 cm betragen. Die Bewehrung soll in Entwurfsrichtung As1 verlaufen.“

```
ConcreteSurfaceReinforcements: Erzeuge eine obere Flächenbewehrung mit dem Namen OBERBEWEHRUNG auf den Flächen 1 und 2. Verwende Betonmaterial Nummer 2. Die Bewehrung soll auf der Oberfläche liegen, ohne Bügel. Der Betondeckungsrandabstand soll oben und unten jeweils 3 cm betragen. Die Bewehrung soll in Entwurfsrichtung As1 verlaufen.

To define the load section for creating an upper surface reinforcement named "OBERBEWEHRUNG" on surfaces 1 and 2, using concrete material number 2, and ensuring the reinforcement is on the surface without stirrups, with a concrete cover offset of 3 cm on both the top and bottom, and aligned in the design direction As1, you can use the following code snippet.

python
# ---- ConcreteSurfaceReinforcements Section ----
ConcreteSurfaceReinforcements(
    no=1,
    name='OBERBEWEHRUNG',
    surfaces='1, 2', # Apply to surfaces 1 and 2
    material='2', # Use concrete material number 2
    location_type=SurfaceReinforcementLocationType.LOCATION_TYPE_ON_SURFACE, # Place reinforcement on the surface
    reinforcement_type=SurfaceReinforcementType.REINFORCEMENT_TYPE_REBAR, # Use rebar type reinforcement
    reinforcement_type_parameters=(None, 0.15, False), # Diameter unspecified, 15 cm spacing, no stirrups
    cover_offset=(True, True, 0.03, 0.03), # 3 cm cover offset on top and bottom
    reinforcement_direction=SurfaceReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_IN_DESIGN_REINFORCEMENT_DIRECTION, # Align in design direction As1
    reinforcement_direction_parameters=(SurfaceReinforcementDesignDirection.DESIGN_REINFORCEMENT_DIRECTION_A_S_1),
    comment='Obere Bewehrung Fläche 1 und 2', # Comment for the reinforcement
    params=None, # No additional parameters
    model=model # The model to which this reinforcement belongs
)
# ---- ConcreteSurfaceReinforcements Section ----
```

Der folgende Code muss kopiert und in die Eingabedatei für die Flächenbewehrung eingefügt werden.

Dabei gilt: Für die Werte 0.15 und None, None steht 0.15 als Schätzwert und None, None für die Optimierung. Es ist jedoch nicht zulässig, sowohl den Abstand als auch den Durchmesser mit None zu definieren – mindestens einer der beiden Werte muss vorgegeben werden.

ConcreteSurfaceReinforcements

```
# ---- ConcreteSurfaceReinforcements Section ----
ConcreteSurfaceReinforcements(
    no=1,
    name='OBERBEWEHRUNG',
    surfaces='1, 2', # Apply to surfaces 1 and 2
    material='2', # Use concrete material number 2
    location_type=SurfaceReinforcementLocationType.LOCATION_TYPE_ON_SURFACE, # Place reinforcement on the surface
    reinforcement_type=SurfaceReinforcementType.REINFORCEMENT_TYPE_REBAR, # Use rebar type reinforcement
    reinforcement_type_parameters=(None, 0.15, False), # Diameter unspecified, 15 cm spacing, no stirrups
    cover_offset=(True, True, 0.03, 0.03), # 3 cm cover offset on top and bottom
    reinforcement_direction=SurfaceReinforcementDirectionType.REINFORCEMENT_DIRECTION_TYPE_IN_DESIGN_REINFORCEMENT_DIRECTION, # Align in design direction As1
    reinforcement_direction_parameters=(SurfaceReinforcementDesignDirection.DESIGN_REINFORCEMENT_DIRECTION_A_S_1),
    comment='Obere Bewehrung Fläche 1 und 2', # Comment for the reinforcement
    params=None, # No additional parameters
    model=model # The model to which this reinforcement belongs
)
# ---- ConcreteSurfaceReinforcements Section ----
```

Abb.8.57: Verlauf Flächenbewehrung Abfrage an GPT

8.5.2.10 Beton Konfiguration Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Beton Konfiguration abgefragt, und zwar in folgender Form:

ConcreteUltimateConfiguration: ULS-Konfiguration, alle Stabsets. Name: ULS-Standard“

```

ConcreteUltimateConfiguration: ULS-Konfiguration, alle Stabsets, Name: ULS-Standard
To define the load section for a ULS (Ultimate Limit State) configuration named "ULS-Standard" that applies to all member sets, you can use the following code snippet based on the provided syntax and example:
...python
# ---- ConcreteUltimateConfiguration Section ----
ConcreteUltimateConfiguration(
    no=1,
    name='ULS-Standard', # Name of the ULS configuration
    members='All', # Apply to all members
    member_sets='All', # Apply to all member sets
    surfaces='All', # Apply to all surfaces
    surface_sets='All', # Apply to all surface sets
    nodes='', # No specific nodes defined
    comment='ULS-Konfiguration für alle Stabsets', # Comment for the configuration
    params=None, # No additional parameters
    model=Model, # The model to which this configuration belongs
)
# ---- ConcreteUltimateConfiguration Section ----

```

Der folgende Code muss kopiert und in die Beton Konfiguration eingabedatei eingefügt werden.

```

ConcreteUltimateConfiguratio
# ---- ConcreteUltimateConfiguration Section ----
ConcreteUltimateConfiguration(
    no=1,
    name='ULS-Standard', # Name of the ULS configuration
    members='All', # Apply to all members
    member_sets='All', # Apply to all member sets
    surfaces='All', # Apply to all surfaces
    surface_sets='All', # Apply to all surface sets
    nodes='', # No specific nodes defined
    comment='ULS-Konfiguration für alle Stabsets', # Comment for the configuration
    params=None, # No additional parameters
    model=Model # The model to which this configuration belongs
)
# ---- ConcreteUltimateConfiguration Section ----

```

Abb.8.58: Verlauf Beton Konfiguration Abfrage an GPT

8.5.2.11 Global Parameters Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Global Parameters abgefragt, und zwar in folgender Form:

„GlobalParameter: Erstelle globale Parameter für ein Fundamentmodell. Definiere zwei feste Werte: Breite des Fundaments (0.75 m) und Höhe der Wandplatte (1.00 m). Füge zwei optimierbare Dicken hinzu: Wanddicke (0.20 m, 0.15–0.30 m, 1 Schritte) und Fundamentdicke (0.30 m, 0.20–0.50 m, 1 Schritte). Danach zwei Flächenparameter als Formel: Wandfläche = Wanddicke × Höhe, Fundamentfläche = Fundamentdicke × Breite. Ergänze einen CO₂-Parameter: CO₂ = Fundamentfläche × 250 kg/m². Zusätzlich zwei optimierbare Bewehrungsparameter: Durchmesser (0.012 m, 0.008–0.020 m, 1 Schritte) und Abstand (0.15 m, 0.10–0.25 m, 1 Schritte). Alle Parameter sollen eindeutige Namen, Symbole, passende Einheitengruppen, Definitionstypen und Kommentare enthalten.“

Globalparameter: Erstelle globale Parameter für ein Fundamentmodell. Definiere zwei feste Werte: Breite des Fundaments (0.75 m) und Höhe der Wandplatte (1.00 m). Füge zwei optimierbare Dicken hinzu: Wanddicke (0.20 m, 0.15-0.30 m, 1 Schritte) und Fundamentdicke (0.30 m, 0.20-0.50 m, 1 Schritte). Danach zwei Flächenparameter als Formel: Wandfläche = Wanddicke × Höhe, Fundamentfläche = Fundamentdicke × Breite. Ergänze einen CO₂-Parameter: CO₂ = Fundamentfläche × 250 kg/m³. Zusätzlich zwei optimierbare Bewehrungsparameter: Durchmesser (0.012 m, 0.008-0.020 m, 1 Schritte) und Abstand (0.15 m, 0.10-0.25 m, 1 Schritte). Alle Parameter sollen einheitliche Namen, Symbole, passende Einheitsgruppen, Definitionstypen und Kommentare erhalten.

To define the global parameters for a foundation model with the specified fixed values, optimizable thicknesses, area formulas, and CO₂ parameter, you can use the following code snippet:

```

python
---- GlobalParameter Section ----
# 1. Fixed Geometry Parameters
GlobalParameter.AddParameter(
    no=1,
    name='Breite Fundament',
    symbol='b_f',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_VALUE,
    definition_parameter=[0.75], # 0.75 m
    comment='Breite des Fundaments',
    model=Model
)

GlobalParameter.AddParameter(
    no=2,
    name='Hoehe Wandplatte',
    symbol='h_w',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_VALUE,
    definition_parameter=[1.0], # 1.00 m
    comment='Hoehe der Wandplatte',
    model=Model
)

GlobalParameter.AddParameter(
    no=3,
    name='Wanddicke',
    symbol='d_w',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_OPTIMIZATION,
    definition_parameter=[0.20, 0.15, 0.30, 1], # Start value, min, max, steps
    comment='Optimierbare Dicke der Wandplatte',
    model=Model
)

GlobalParameter.AddParameter(
    no=4,
    name='Fundamentdicke',
    symbol='d_f',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_OPTIMIZATION,
    definition_parameter=[0.30, 0.20, 0.50, 1], # Start value, min, max, steps
    comment='Optimierbare Dicke der Fundamentplatte',
    model=Model
)

# 3. Area Formulas
GlobalParameter.AddParameter(
    no=5,
    name='Wandflaeche',
    symbol='A_w',
    unit_group=GlobalParameterUnitGroup.AREA,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_FORMULA,
    definition_parameter=['Wanddicke*Hoehe Wandplatte'], # Formula for wall area
    comment='Wandflaeche basierend auf Dicke und Hoehe',
    model=Model
)

GlobalParameter.AddParameter(
    no=6,
    name='Fundamentflaeche',
    symbol='A_f',
    unit_group=GlobalParameterUnitGroup.AREA,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_FORMULA,
    definition_parameter=['Fundamentdicke*Breite Fundament'], # Formula for foundation area
    comment='Fundamentflaeche basierend auf Dicke und Breite',
    model=Model
)

# 4. CO2 Parameter
GlobalParameter.AddParameter(
    no=7,
    name='CO2 Fundament',
    symbol='CO2_f',
    unit_group=GlobalParameterUnitGroup.MASS, # kg on t CO2
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_FORMULA,
    definition_parameter=['Fundamentflaeche*250'], # CO2 calculation
    comment='CO2-Emissionen basierend auf Fundamentflaeche',
    model=Model
)

# 5. Optimizable Reinforcement Parameters
GlobalParameter.AddParameter(
    no=8,
    name='Bewehrungsdurchmesser',
    symbol='d_s',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_OPTIMIZATION,
    definition_parameter=[0.012, 0.008, 0.020, 1], # Start value, min, max, steps
    comment='Optimierbarer Durchmesser der Bewehrung',
    model=Model
)

GlobalParameter.AddParameter(
    no=9,
    name='Bewehrungsabstand',
    symbol='a_s',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_OPTIMIZATION,
    definition_parameter=[0.15, 0.10, 0.25, 1], # Start value, min, max, steps
    comment='Optimierbarer Abstand der Bewehrung',
    model=Model
)

# ---- GlobalParameter Section ----
    
```

Der folgende Code muss kopiert und in die Beton Konfiguration Eingabe Datei eingefügt werden.

```

Global Parameters
# ---- GlobalParameter Section ----
# 1. Fixed Geometry Parameters
GlobalParameter.AddParameter(
    no=1,
    name='Breite Fundament',
    symbol='b_f',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_VALUE,
    definition_parameter=[0.75], # 0.75 m
    comment='Breite des Fundaments',
    model=Model
)

GlobalParameter.AddParameter(
    no=2,
    name='Hoehe Wandplatte',
    symbol='h_w',
    unit_group=GlobalParameterUnitGroup.LENGTH,
    definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_VALUE,
    definition_parameter=[1.0], # 1.00 m
    comment='Hoehe der Wandplatte',
    model=Model
)

# 2. Optimizable Thicknesses
GlobalParameter.AddParameter(
    no=3,
    name='Wanddicke',
    
```

Abb.8.59: Verlauf Globalparameters Abfrage an GPT

8.5.2.12 Optimierung Einstellungen Abfrage an GPT

In diesem Abschnitt wird GPT gezielt nach dem gewünschten Optimierung Einstellungen abgefragt, und zwar in folgender Form:

„OptimizationSettings: Erstelle eine Optimierungseinstellung mit dem Namen „CO2-Optimierung“. Ziel ist die Minimierung der CO2-Emissionen. Setze den Mutationsanteil auf 20 Prozent, die Gesamtzahl der Mutationen auf 1.“

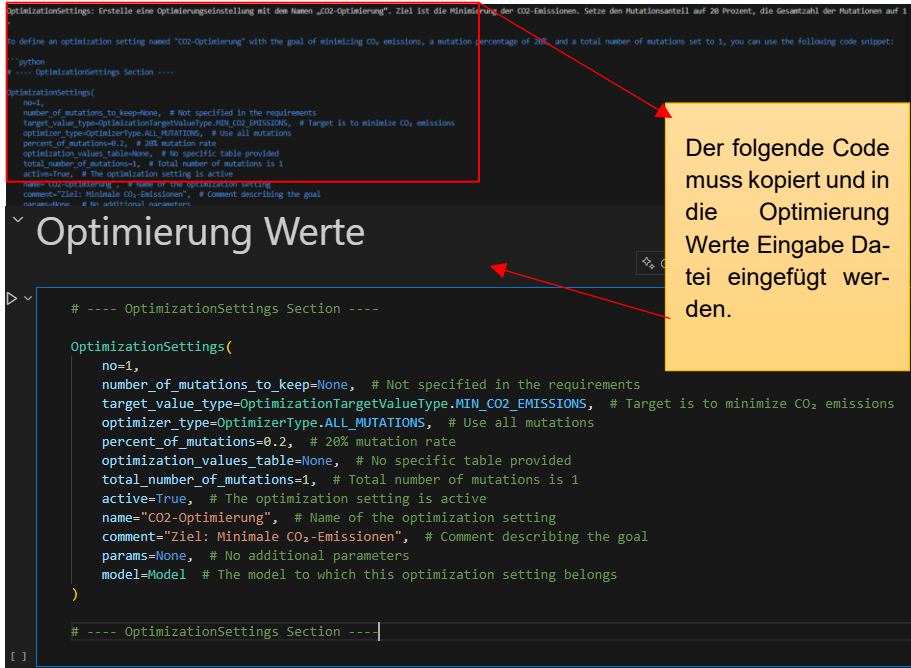


Abb.8.60: Verlauf Optimierungseinstellungen Abfrage an GPT

8.5.2.13 Solver Anrufen

Beim Aufruf des Solvers bleibt der Prozess stets hängen, und zwar beim folgenden Befehl:

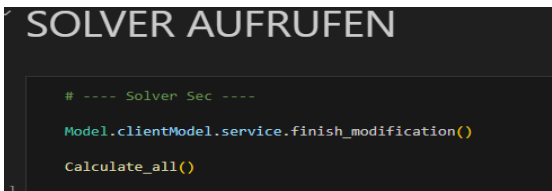


Abb.8.61: Solver Anrufen Befehl

8.5.3 Start der Berechnung

Um die Berechnung zu starten, muss der Webservice von Dlubal aktiviert werden – siehe dazu die Abbildung.

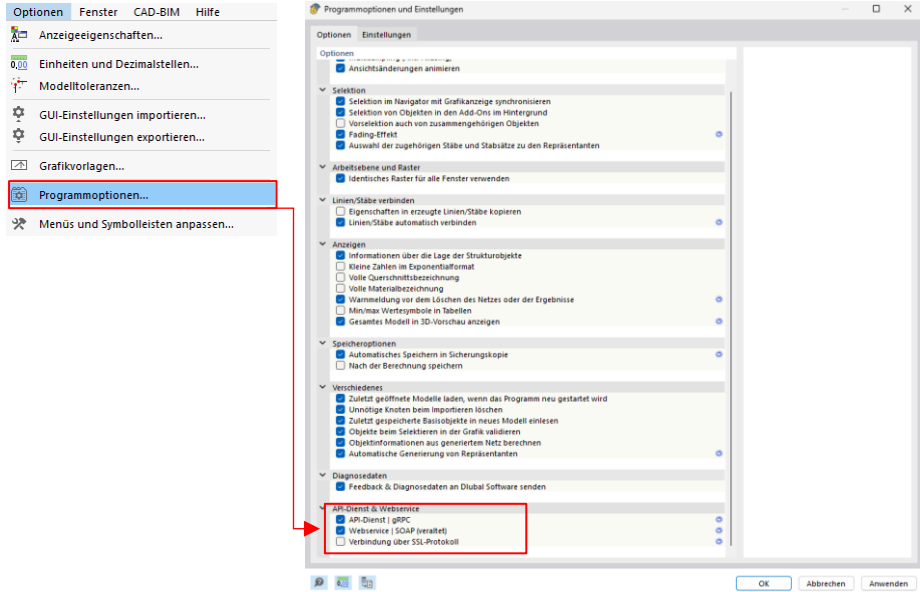


Abb.8.62: Aktivierung Webservices und API

Nach erfolgreicher Aktivierung und Einrichtung ist es nun möglich, die Berechnung direkt in Visual Studio durchzuführen.

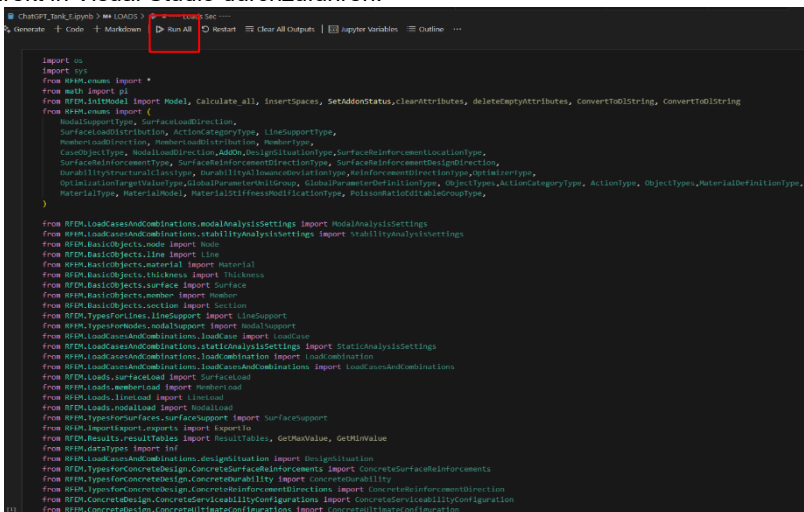


Abb.8.63: Berechnung Starten

8.5.4 Ergebnisse

Als Ergebnisse werden verschiedene Ausgaben generiert, darunter Basisangaben und Betontyp, Ergebnisse der Betonbemessung sowie die CO₂-Optimierung in Abhängigkeit von Plattendicke und erforderlicher Bewehrung (As).

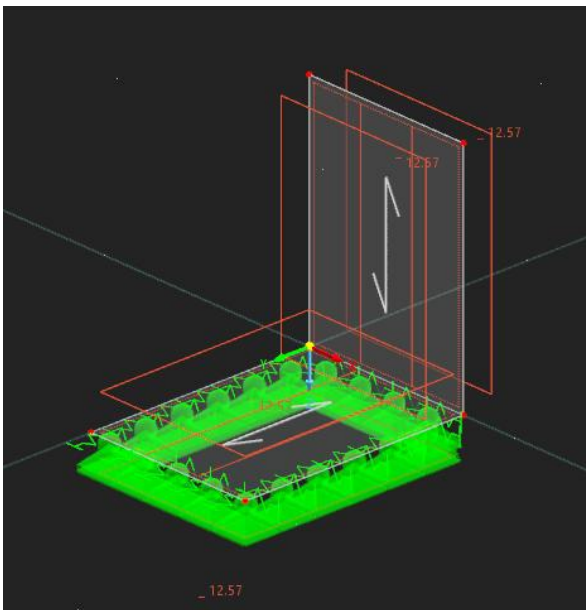
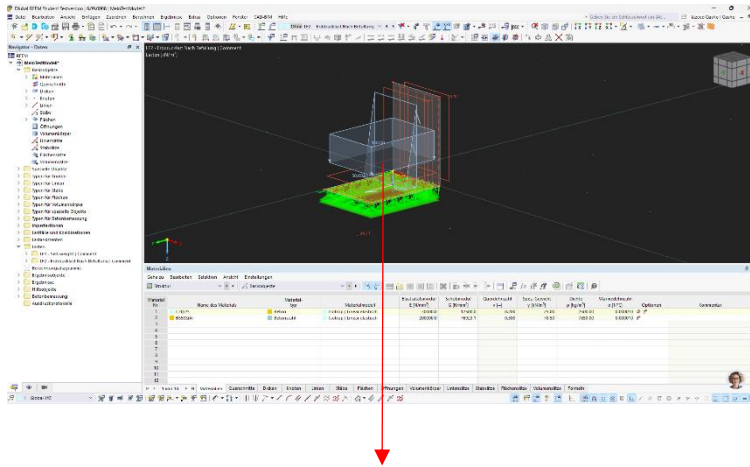


Abb.8.64: Modul nach der Berechnung

8.5.4.1 Basisobjekte

Nachdem alle Befehle zur Definition von Material, Geometrie und Querschnitt, Flächen ausgeführt wurden, werden die Ergebnisse für die im Skript angegebenen unterausgegeben.

➤ **Material:**

Für die Materialdefinition wurden die Materialien Beton C20/25 und B550S(A) abgefragt. Die Ausgabe erfolgte erfolgreich mit den gewünschten Materialien.



Abb.8.65: Ausgabe von Materialien

➤ **Dicken:**

Die Dicken von Fundament und Wand wurden als unbekannte bzw. optimierbare Parameter definiert. Die untenstehende Abbildung zeigt, wie die Benennung erfolgt ist und auf welche Weise die Dicke als optimierbar festgelegt wurde.

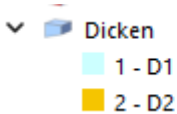


Abb.8.65: Ausgabe von Dicken

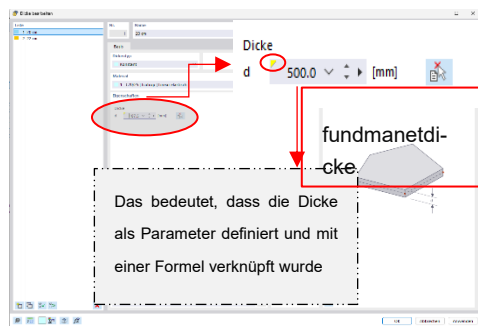
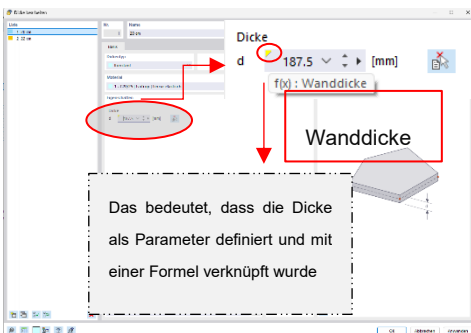


Abb.8.66: Ausgabe von Dicken als Parameters

➤ **Geometry:**

Im Abschnitt Geometrie wurden die Abmessungen des Fundaments eingegeben und erfolgreich in Knoten- und Linienelemente überführt. Die Generierung erfolgte exakt wie vorgesehen, ohne doppelte Knoten oder Linien, was besonders relevant ist, da solche Duplikate zu Fehlern im Berechnungsprozess führen können. Die Abbildung unten zeigt die Knotenanordnung im Modell und veranschaulicht die saubere und fehlerfreie Geometrieerstellung.

- ▼ • Knoten
- 1 - Kartesisch | X: 0.000 m | Y: 0.000 m | Z: 0.000 m
- 2 - Kartesisch | X: 0.750 m | Y: 0.000 m | Z: 0.000 m
- 3 - Kartesisch | X: 0.000 m | Y: 0.000 m | Z: -1.000 m
- 4 - Kartesisch | X: 0.750 m | Y: 0.000 m | Z: -1.000 m
- 5 - Kartesisch | X: 0.000 m | Y: 1.000 m | Z: 0.000 m
- 6 - Kartesisch | X: 0.750 m | Y: 1.000 m | Z: 0.000 m

Insgesamt wurden sechs Knoten definiert. Die obestehende Abbildung zeigt die genaue Lage jedes einzelnen Knotens im Modell

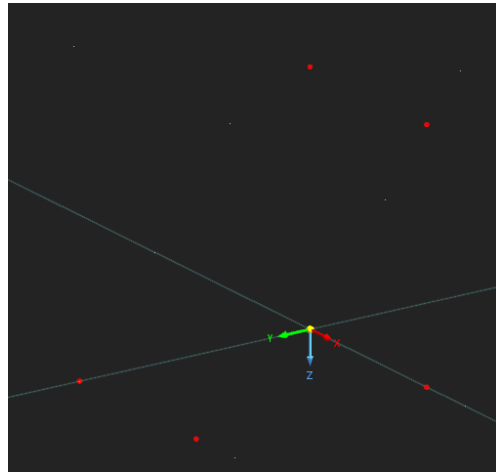


Abb.8.67: Ausgabe von Knoten

Die folgende Abbildung zeigt, wie die Knoten durch Linien miteinander verbunden wurden

- ▼ • Linien
- 1 - 1,2 | Polylinie | L: 0.750 m
- 2 - 1,3 | Polylinie | L: 1.000 m
- 3 - 2,4 | Polylinie | L: 1.000 m
- 4 - 3,4 | Polylinie | L: 0.750 m
- 5 - 1,5 | Polylinie | L: 1.000 m
- 6 - 2,6 | Polylinie | L: 1.000 m
- 7 - 5,6 | Polylinie | L: 0.750 m

Insgesamt wurden sieben Linien definiert. Die obestehende Abbildung zeigt den genauen Verlauf und die Verbindung der Linien im Modell.

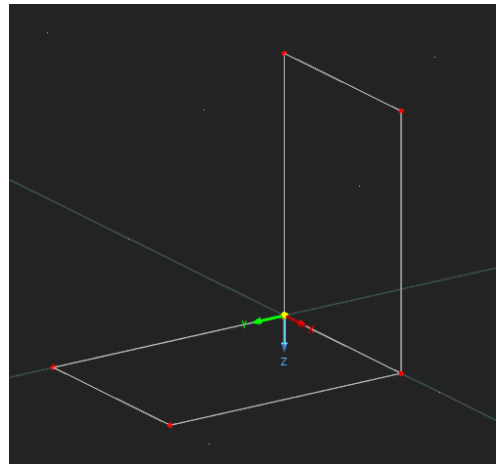


Abb.8.68: Ausgabe von Linien

➤ **Flächen**

Im Modell wurden zwei Flächen definiert. Die Abbildung zeigt diese Flächen mit dem zugewiesenen Material Beton C20/25. Eine Fläche stellt die Fundamentplatte (horizontal), die andere die Wand (vertikal) dar – jeweils mit der entsprechenden Dicke.

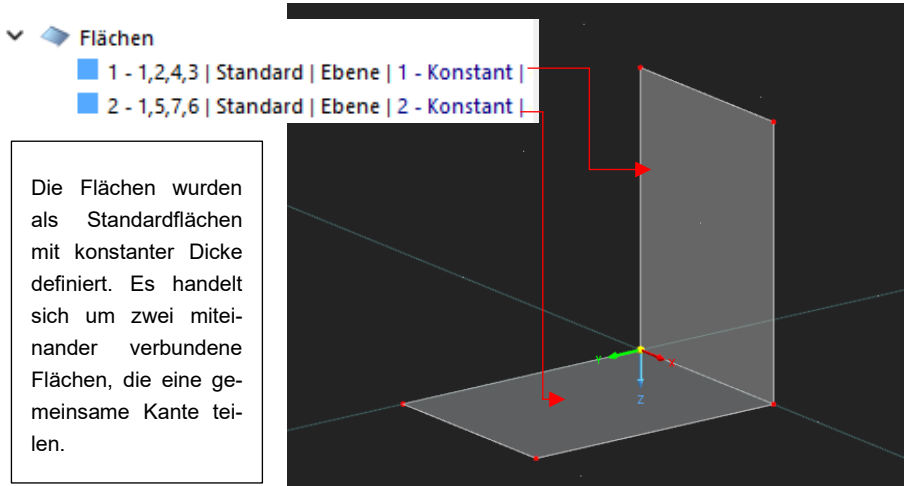


Abb.8.69: Ausgabe von Flächen

8.5.4.2 Lasten

In diesem Abschnitt wurden sowohl das Eigengewicht als auch veränderliche Lasten berücksichtigt und erfolgreich ins Modell eingefügt. Die Abbildung zeigt die genaue Anordnung und Darstellung dieser Lasten.

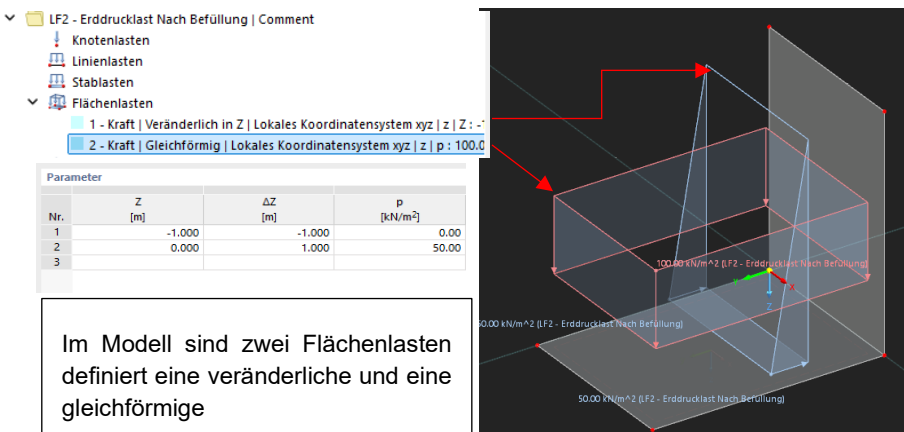


Abb.8.70: Ausgabe von Lasten

8.5.4.3 Lastfälle und Kombinationen

In diesem Abschnitt wurden die Lastfälle und -kombinationen hinzugefügt, basierend auf der GPT-Abfrage. Zu den wichtigsten Punkten zählen die Definition der Bemessungssituationen nach GZT, die Erstellung einer Lastfallkombination sowie die Festlegung der Optimierungseinstellungen. Die untenstehende Abbildung zeigt, wie alle relevanten Parameter im Modell hinterlegt wurden.

In diesem Abschnitt wurden alle gewünschten Lastfallvarianten definiert und entsprechend den Anforderungen umgesetzt.

| Nr. | Name | Standardwert | Einheit | Zustand | Empfindlichkeitfaktor | Aktiv |
|-----|--------------------|--------------|---------|---------|-----------------------|-------------------------------------|
| 1 | Wanddicke | 187.5 | mm | 5 | | <input checked="" type="checkbox"/> |
| 2 | Fundamentdicke | 500.0 | mm | 5 | | <input checked="" type="checkbox"/> |
| 3 | Bewehrungsdurch... | 0.020 | m | 6 | | <input checked="" type="checkbox"/> |
| 4 | Bewehrungsabstand | 0.130 | m | 6 | | <input checked="" type="checkbox"/> |

Abb.8.71: Ausgabe von Lastfälle Kombinationen

8.5.4.4 Lagerung

In diesem Abschnitt wurde das Auflager erfolgreich als Flächenlager definiert. Es handelt sich dabei um eine flächige Unterstüzung, die direkt unter der Fundamentplatte angeordnet ist. Das Lager wurde so eingestellt, dass es vertikale Verschiebungen verhindert, während horizontale Bewegungen – je nach Randbedingung – teilweise zugelassen werden können. Die Definition als Flächenlager sorgt für eine gleichmäßige Lastverteilung und realitätsnahe Abbildung der Auflagerverhältnisse. Die untenstehende Abbildung zeigt die genaue Position sowie die Eigenschaften des definierten Flächenlagers im Modell.

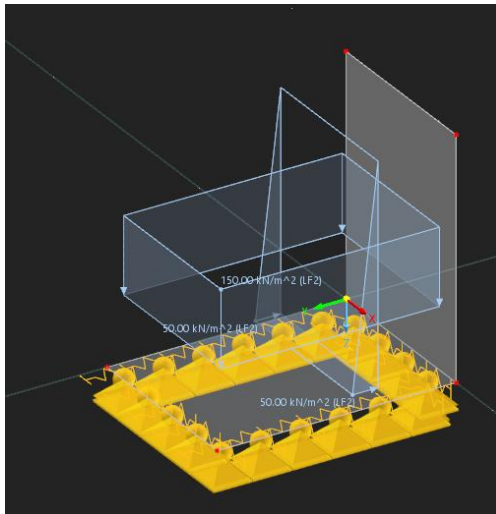
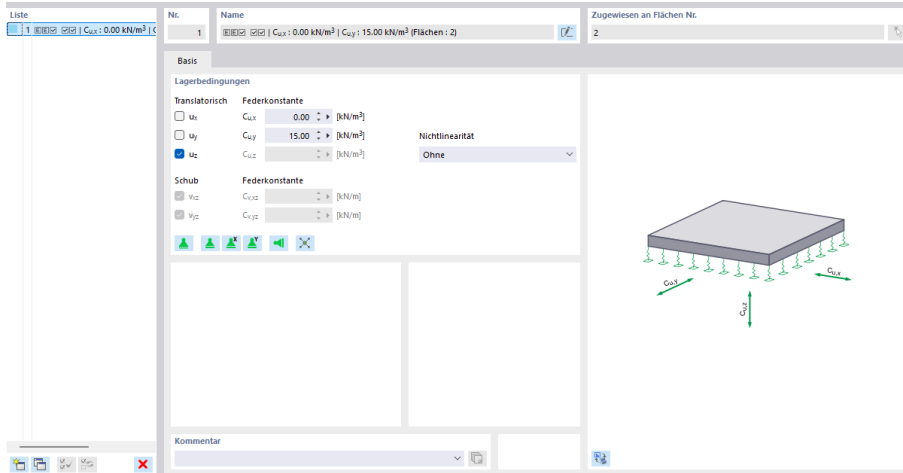


Abb.8.72: Ausgabe von Lagerung

8.5.4.5 Typen Für Betonbemessung

Im Abschnitt zur Betonbemessung wurden alle erforderlichen Parameter erfolgreich hinzugefügt und festgelegt. Einer der zentralen Punkte war die Definition des Bewehrungsdurchmessers (d_s), (A_s) als Optimierungsparameter. Die Abbildung veranschaulicht die Eingabemaske sowie die Festlegung der beiden Optimierungsparameter

The screenshot displays the 'Typen für Betonbemessung' (Types for Concrete Design) configuration window. The left sidebar shows a tree structure with the following items: Knicklängen, Betondauerhaftigkeiten (1 - XC 1), Bewehrungsrichtungen (1 - RD 1), Flächenbewehrungen (1 - OBERBEWEHRUNG), and Durchstanzbewehrungen. The main configuration area is set to 'Ortstyp: Auf Fläche', 'Bewehrung: 2 - B550S(A) | Isotrop | Linear elastisch', and 'Bewehrungstyp: Bewehrungsstab'. The 'Durchmesser' field is set to $d_s = 20.0$ [mm] and the 'Stababstand' field is set to $s = 0.130$ [m]. Below these are fields for 'Zusätzliche Querbewehrung' with $d_{s,90^\circ}$ and s_{90° fields. The 'Basis' section shows 'Richtungstyp: Erste Bewehrungsrichtung in y'. The bottom right features a diagram of a concrete wall cross-section with various exposure classes (XC1, XC2, XC3, XC4, XF1, XF2, XF3, XF4, XF5, XF6, XF7, XF8, XF9, XF10, XF11, XF12, XF13, XF14, XF15, XF16, XF17, XF18, XF19, XF20, XF21, XF22, XF23, XF24, XF25, XF26, XF27, XF28, XF29, XF30, XF31, XF32, XF33, XF34, XF35, XF36, XF37, XF38, XF39, XF40, XF41, XF42, XF43, XF44, XF45, XF46, XF47, XF48, XF49, XF50, XF51, XF52, XF53, XF54, XF55, XF56, XF57, XF58, XF59, XF60, XF61, XF62, XF63, XF64, XF65, XF66, XF67, XF68, XF69, XF70, XF71, XF72, XF73, XF74, XF75, XF76, XF77, XF78, XF79, XF80, XF81, XF82, XF83, XF84, XF85, XF86, XF87, XF88, XF89, XF90, XF91, XF92, XF93, XF94, XF95, XF96, XF97, XF98, XF99, XF100) and environmental conditions (sun, clouds, rain, snow) indicated by icons.

Abb.8.73: Ausgabe von Typen für Betonbemessung

8.5.4.6 Einfluss von Dicke und Bewehrung auf die CO₂-Emissionen

In der Tabelle sind die Ergebnisse von zwölf durchgeführten Optimierungsläufen dargestellt. Für jedes Modell wurden der minimale CO₂-Wert sowie die zugehörigen Konstruktionsparameter dokumentiert. Dazu gehören die Wanddicke, die Fundamentdicke, der Bewehrungsdurchmesser sowie der Bewehrungsabstand.

Die Modellnamen sind fortlaufend nummeriert. Zu jedem Modell ist der berechnete Zielwert der CO₂-Emissionen in Kilogramm CO₂-Äquivalent (kg-CO₂e) aufgeführt. Zusätzlich werden die jeweiligen Dickenangaben für Wand und Fundament in Metern sowie die Bewehrungsparameter (Durchmesser und Abstand) angegeben.

Die dargestellten Daten bilden die Grundlage für eine weiterführende Bewertung hinsichtlich der Nachhaltigkeit und der konstruktiven Effizienz der einzelnen Varianten.

| Ordnung | Model Name | Zielwert
CO ₂ min [kg-CO ₂ e] | Wanddicke [m] | Fundamentdicke [m] | Bewehrungsdurchmesser
[m] | Bewehrungsabstand [m] |
|---------|---------------|--|---------------|--------------------|------------------------------|-----------------------|
| 1 | Modell Nr. 1 | 750.00 | 0.200 | 0.350 | 0.018 | 0.150 |
| 2 | Modell Nr. 2 | 750.00 | 0.300 | 0.200 | 0.016 | 0.200 |
| 3 | Modell Nr. 3 | 750.00 | 0.275 | 0.400 | 0.012 | 0.100 |
| 4 | Modell Nr. 4 | 750.00 | 0.275 | 0.350 | 0.018 | 0.100 |
| 5 | Modell Nr. 5 | 750.00 | 0.300 | 0.350 | 0.008 | 0.175 |
| 6 | Modell Nr. 6 | 750.00 | 0.300 | 0.350 | 0.012 | 0.175 |
| 7 | Modell Nr. 7 | 750.00 | 0.150 | 0.450 | 0.008 | 0.125 |
| 8 | Modell Nr. 8 | 750.00 | 0.275 | 0.350 | 0.010 | 0.100 |
| 9 | Modell Nr. 9 | 750.00 | 0.250 | 0.300 | 0.016 | 0.100 |
| 10 | Modell Nr. 10 | 750.00 | 0.225 | 0.500 | 0.012 | 0.100 |
| 11 | Modell Nr. 11 | 750.00 | 0.200 | 0.200 | 0.012 | 0.200 |
| 12 | Modell Nr. 12 | 750.00 | 0.250 | 0.500 | 0.020 | 0.125 |
| 13 | Modell Nr. 13 | 750.00 | 0.250 | 0.350 | 0.008 | 0.150 |
| 14 | Modell Nr. 14 | 750.00 | 0.300 | 0.500 | 0.016 | 0.200 |
| 15 | Modell Nr. 15 | 750.00 | 0.275 | 0.400 | 0.012 | 0.250 |
| 16 | Modell Nr. 16 | 750.00 | 0.275 | 0.350 | 0.018 | 0.250 |
| 17 | Modell Nr. 17 | 750.00 | 0.300 | 0.350 | 0.020 | 0.175 |
| 18 | Modell Nr. 18 | 750.00 | 0.150 | 0.450 | 0.020 | 0.125 |
| 19 | Modell Nr. 19 | 750.00 | 0.275 | 0.350 | 0.010 | 0.250 |
| 20 | Modell Nr. 20 | 750.00 | 0.250 | 0.300 | 0.016 | 0.250 |
| 21 | Modell Nr. 21 | 750.00 | 0.225 | 0.500 | 0.012 | 0.250 |
| 22 | Modell Nr. 22 | 750.00 | 0.200 | 0.500 | 0.012 | 0.200 |
| 23 | Modell Nr. 23 | 750.00 | 0.250 | 0.350 | 0.020 | 0.150 |
| 24 | Modell Nr. 24 | 750.00 | 0.300 | 0.450 | 0.016 | 0.200 |

Abb.8.74: Ergebnisse von Optimierung

8.5.4.7 Betonbemessung – Ausnutzungen an Dicken (Nachweise)

Die Abbildung 8.75 zeigt die Ergebnisse der Betonbemessung bezogen auf eine vertikale Fläche mit definierter Dicke. Und Die Abbildung 8.76 zeigt die Ergebnisse der Betonbemessung bezogen auf eine vertikale Fläche mit definierter Dicke Es handelt sich um detaillierte Nachweisdaten für eine Fläche mit der Bezeichnung D1, wobei verschiedene Nachweisarten gemäß Eurocode-Normen (ULS, und Konstruktionsregeln) berücksichtigt wurden.

In der Tabelle sind die zugehörigen Netzknotenkoordinaten, die Beanspruchungssituation, die Belastungsnummer sowie das jeweilige Nachweiskriterium η aufgeführt. Die Nachweiskriterien liegen überwiegend deutlich unter dem Grenzwert von 1.0, was auf eine ausreichende Tragfähigkeit und Gebrauchstauglichkeit hinweist.

Zudem wird für jede Position auf der Fläche der zugehörige Nachweistyp benannt – z. B. Längsbewehrungswiderstand, Druckstrebenfähigkeit, Mindestbewehrung oder

maximale Betongüte. Die zugehörige Beschreibung verweist präzise auf die entsprechenden Regelstellen in den Normen (z. B. EN 1992-1-1), sodass die Herkunft der Nachweisformel nachvollziehbar ist.

| Dicke Nr. | Fläche Nr. | Netznoten / Element Nr. | Netznoten-Koordinaten [m] | Bemessungssituation | Belastung Nr. | Nachweiskriterium η [-] | Nachweisart | Beschreibung |
|-----------|------------|-------------------------|---------------------------|---------------------|---------------|------------------------------|--|--------------|
| 1 | D1 | | X Y Z | | | | | |
| | 1 | 9 | 0,375 0,000 0,000 | B51 LK1 | 0,353 | UL0101_01 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Oberseite (-) gem. 6.1 | |
| | | 3 | 0,000 0,000 -1,000 | B51 LK1 | 0,021 | UL0101_03 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Unterseite (-) gem. 6.1 | |
| | | 1 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | UL0101_04 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 2. Richtung an der Unterseite (-) gem. 6.1 | |
| | | 13 | 0,000 0,000 -0,500 | B51 LK1 | 0,020 | UL0102_01 | Grenzustand der Tragfähigkeit Druckbetontragfähigkeit an der Oberseite (-) nach 6.1 | |
| | | 9 | 0,375 0,000 0,000 | B51 LK1 | 0,592 | UL0203_02 | Grenzustand der Tragfähigkeit Schubbeanspruchbarkeit - Bewehrungssubtragfähigkeit nach 6.2 | |
| | | 1 | 0,000 0,000 0,000 | B51 LK1 | 0,333 | DC0401_00 | Dauerhaftigkeit und Betondeckung Betondeckung nach 4.4.1 | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,360 | DW0300_01 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Oberseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0300_02 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Oberseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0300_03 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Unterseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0301_00 | Konstruktionsregeln Mindestschubbewehrungsgrad nach 9.3.2(2), 9.2.2(5) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0302_00 | Konstruktionsregeln Größter Längsabstand zweier aufeinanderfolgender Querkraftbewehrungselemente nach 9.3.2(4) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0303_00 | Konstruktionsregeln Größter Längsabstand zweier aufeinanderfolgender Querkraftbewehrungselemente nach 9.3.2(5) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0304_00 | Konstruktionsregeln Maximale Dicke der Platte mit Schubbewehrung gem. 9.3.2(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,131 | DW0307_01 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Oberseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0307_02 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Oberseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,131 | DW0307_03 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Unterseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0307_04 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Unterseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0308_02 | Konstruktionsregeln Konstruktive Querbewehrung für Platten in Unterseite gem. 9.3.1.1(2) | |
| | | | 0,000 0,000 0,000 | B51 LK1 | 0,222 | MA0100_00 | Baustoffe Maximalwert der Betonfestigkeitsklasse (C_{90}) nach 3.1.2(2) | |

Abb.8.75: Betonbemessung bezogen auf eine vertikale Dicke D1

| Dicke Nr. | Fläche Nr. | Netznoten / Element Nr. | Netznoten-Koordinaten [m] | Bemessungssituation | Belastung Nr. | Nachweiskriterium η [-] | Nachweisart | Beschreibung |
|-----------|------------|-------------------------|---------------------------|---------------------|---------------|------------------------------|--|--------------|
| 2 | | | X Y Z | | | | | |
| | | 1 | 0,375 0,000 0,000 | B51 LK1 | 0,535 | UL0101_01 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Oberseite (-) gem. 6.1 | |
| | | 2 | 0,750 0,500 0,000 | B51 LK1 | 0,121 | UL0101_03 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Unterseite (-) gem. 6.1 | |
| | | 3 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | UL0101_04 | Grenzustand der Tragfähigkeit Längsbewehrungswiderstand in 2. Richtung an der Unterseite (-) gem. 6.1 | |
| | | 4 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | UL0102_01 | Grenzustand der Tragfähigkeit Druckbetontragfähigkeit an der Oberseite (-) nach 6.1 | |
| | | 5 | 0,750 0,500 0,000 | B51 LK1 | 0,094 | UL0102_02 | Grenzustand der Tragfähigkeit Druckbetontragfähigkeit an der Unterseite (-) nach 6.1 | |
| | | 6 | 0,375 0,000 0,000 | B51 LK1 | 1,216 | UL0203_02 | Grenzustand der Tragfähigkeit Schubbeanspruchbarkeit - Bewehrungssubtragfähigkeit nach 6.2 | |
| | | 7 | 0,000 0,000 0,000 | B51 LK1 | 0,333 | DC0401_00 | Dauerhaftigkeit und Betondeckung Betondeckung nach 4.4.1 | |
| | | 8 | 0,000 0,000 0,000 | B51 LK1 | 0,360 | DW0300_01 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Oberseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | 9 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0300_02 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Oberseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | 10 | 0,750 0,500 0,000 | B51 LK1 | 0,360 | DW0300_03 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Unterseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | 11 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0300_04 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Unterseite (-) gem. 9.2.1.1(1) von 9.3.1.1(1) | |
| | | 12 | 0,375 0,000 0,000 | B51 LK1 | 1,000 | DW0304_00 | Konstruktionsregeln Maximale Dicke der Platte mit Schubbewehrung gem. 9.3.2(1) | |
| | | 13 | 0,000 0,000 0,000 | B51 LK1 | 0,131 | DW0307_01 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Oberseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | 14 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0307_02 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Oberseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | 15 | 0,000 0,000 0,000 | B51 LK1 | 0,131 | DW0307_03 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Unterseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | 16 | 0,000 0,000 0,000 | B51 LK1 | 0,000 | DW0307_04 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Unterseite (-) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| | | 17 | 0,000 0,000 0,000 | B51 LK1 | 0,222 | MA0100_00 | Baustoffe Maximalwert der Betonfestigkeitsklasse (C_{90}) nach 3.1.2(2) | |

Abb.8.76: Betonbemessung bezogen auf eine Horizontale Dicke D2

8.5.4.8 Betonbemessung – Ausnutzungen an Flächen (Nachweise)

Die Tabellen in Abbildung 8.77 und 8.78 zeigen die Bemessungsergebnisse für zwei Flächenbereiche mit konstanter Dicke von 200 mm aus C20/25-Beton.

- Abbildung 8.77 betrifft Fläche Nr. 1 (Ebene 1, Flächen 1, 2, 3, 4).
- Abbildung 8.78 betrifft Fläche Nr. 2 (Ebene 2, Flächen 1, 5, 6, 7).

Dargestellt sind die Nachweiskriterien η für Tragfähigkeit (ULS), Dauerhaftigkeit (SLS), Konstruktionsregeln sowie Materialnachweise – jeweils mit zugehöriger Nachweisart, Netzpunktkoordinaten und normgerechter Beschreibung.

Exemplarische Darstellung und methodisches Vorgehen

| Fläche Nr. | Netzknoten / Element Nr. | Netzknoten-Koordinaten [m] | | | Bemess.- Situation | Belastung Nr. | Nachweis- Kriterien η [-] | Nachweis- Art | Beschreibung |
|------------|--------------------------|--|-------|--------|--------------------|---------------|--------------------------------|---|---|
| | | X | Y | Z | | | | | |
| 1 | Standard | I - Konstant d : 200,0 mm I - C20/25 | | | | | | | |
| | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 0,535 | UL0101_01 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Oberseite (-z) gem. 6.1 |
| | 3 | 0,000 | 0,000 | -1,000 | BS1 | LK1 | 0,031 | UL0101_03 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Unterseite (-z) gem. 6.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | UL0101_04 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 2. Richtung an der Oberseite (-z) gem. 6.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,071 | UL0102_01 | Grenz Zustand der Tragfähigkeit Druckbeibrtragfähigkeit an der Oberseite (-z) nach 6.1 |
| | 13 | 0,000 | 0,000 | -0,500 | BS1 | LK1 | 0,030 | UL0102_02 | Grenz Zustand der Tragfähigkeit Druckbeibrtragfähigkeit an der Unterseite (-z) nach 6.1 |
| | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 0,592 | UL0203_02 | Grenz Zustand der Tragfähigkeit Schubbeanspruchbarkeit - Bewehrungsschubtragfähigkeit nach 6.2 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,333 | DC0401_00 | Dauerhaftigkeit und Betondeckung Betondeckung nach 4.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,360 | DWO300_01 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Oberseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO300_02 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Oberseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO300_03 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Unterseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO301_00 | Konstruktionsregeln Mindestschubbewehrungsgrad nach 9.3.2(2), 9.3.2(3) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO302_00 | Konstruktionsregeln Größter Längsabstand zweier aufeinanderfolgender Querkribbewehrungselemente nach 9.3.2(4) |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO303_00 | Konstruktionsregeln Größter Längsabstand zweier aufeinanderfolgender Querkribbewehrungselemente nach 9.3.2(5) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO304_00 | Konstruktionsregeln Maximale Dicke der Platte mit Schubbewehrung gem. 9.3.2(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,131 | DWO307_01 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Oberseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO307_02 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Oberseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,131 | DWO307_03 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Unterseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO307_04 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Unterseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO308_02 | Konstruktionsregeln Konstruktive Querbewehrung für Platten in Unterzug gem. 9.3.1.1(2) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,222 | MA0100_00 | Baustoffe Maximalwert der Betonfestigkeitsklasse (C_{max}) nach 3.1.2(2) | |

Abb.8.77: Betonbemessung bezogen auf Ausnutzungen an Fläche 1

| Fläche Nr. | Netzknoten / Element Nr. | Netzknoten-Koordinaten [m] | | | Bemess.- Situation | Belastung Nr. | Nachweis- Kriterien η [-] | Nachweis- Art | Beschreibung |
|------------|--------------------------|--|-------|-------|--------------------|---------------|--------------------------------|---|---|
| | | X | Y | Z | | | | | |
| 2 | Standard | I - Konstant d : 200,0 mm I - C20/25 | | | | | | | |
| | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 0,535 | UL0101_01 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Oberseite (-z) gem. 6.1 |
| | 10 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,121 | UL0101_03 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 1. Richtung an der Unterseite (-z) gem. 6.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | UL0101_04 | Grenz Zustand der Tragfähigkeit Längsbewehrungswiderstand in 2. Richtung an der Oberseite (-z) gem. 6.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,007 | UL0102_01 | Grenz Zustand der Tragfähigkeit Druckbeibrtragfähigkeit an der Oberseite (-z) nach 6.1 |
| | 10 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,004 | UL0102_02 | Grenz Zustand der Tragfähigkeit Druckbeibrtragfähigkeit an der Unterseite (-z) nach 6.1 |
| | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 0,698 | UL0203_02 | Grenz Zustand der Tragfähigkeit Schubbeanspruchbarkeit - Bewehrungsschubtragfähigkeit nach 6.2 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,333 | DC0401_00 | Dauerhaftigkeit und Betondeckung Betondeckung nach 4.1 |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,360 | DWO300_01 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Oberseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO300_02 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Oberseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 10 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,360 | DWO300_03 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 1. Richtung an der Unterseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO300_04 | Konstruktionsregeln Mindestbewehrung der Zugbereiche in 2. Richtung an der Unterseite (-z) gem. 9.2.1.1(1) von 9.3.1.1(1) |
| | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 1,000 | DWO304_00 | Konstruktionsregeln Maximale Dicke der Platte mit Schubbewehrung gem. 9.3.2(1) |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,131 | DWO307_01 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Oberseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO307_02 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Oberseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,131 | DWO307_03 | Konstruktionsregeln Maximale Bewehrung in 1. Richtung an der Unterseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,000 | DWO307_04 | Konstruktionsregeln Maximale Bewehrung in 2. Richtung an der Unterseite (-z) gem. 9.2.1.1(3) aus 9.3.1.1(1) | |
| 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 0,222 | MA0100_00 | Baustoffe Maximalwert der Betonfestigkeitsklasse (C_{max}) nach 3.1.2(2) | |

Abb.8.78: Betonbemessung bezogen auf Ausnutzungen an Fläche 2

8.5.4.9 Betonbemessung – Erforderlich Bewehrung für Fläche

Die Abbildung zeigt eine tabellarische Darstellung der erforderlichen Flächenbewehrung für zwei unterschiedliche Flächenbereiche. In der Tabelle sind jeweils die zugehörigen Netzknoten bzw. Elemente mit ihren Koordinaten in X-, Y- und Z-Richtung aufgeführt, ebenso wie die Bemessungssituation BS1 und die zugehörige Lastfallkombination LK1. Die Tabelle ist in zwei Abschnitte unterteilt: Der obere Abschnitt bezieht sich auf Fläche 1, der untere auf Fläche 2. Für beide Flächen werden die erforderlichen Bewehrungswerte angegeben, unterteilt in Längsbewehrung in erster und zweiter Richtung, jeweils getrennt nach Oberseite (negative Z-Richtung) und Unterseite (positive Z-Richtung). Zusätzlich wird die erforderliche Bügelbewehrung als flächenbezogener Wert angegeben. Die Zeilen mit der Bezeichnung „Extrema“ kennzeichnen dabei bestimmte lokale Nachweispunkte, während die Zeile „Gesamt“ jeweils die zusammenfassenden Bewehrungswerte für die gesamte betrachtete Fläche angibt.

| Fläche Nr. | Netzknoten / Element Nr. | Netzknoten-Koordinaten [m] | | | Bemess.- Situation | Belastung Nr. | Längsbewehrung [cm ² /m] | | | | Bügel [cm ² /m ²] | |
|------------|--------------------------|----------------------------|-------|--------|--------------------|---------------|-------------------------------------|-------------------------|--------------------------|--------------------------|--|------|
| | | X | Y | Z | | | $a_{s,erf,1,-z}$ (oben) | $a_{s,erf,2,-z}$ (oben) | $a_{s,erf,1,+z}$ (unten) | $a_{s,erf,2,+z}$ (unten) | | |
| 1 | Extrema | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,00 | 0,00 | 0,00 |
| | 1 | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,00 | 0,00 | 0,00 |
| | 3 | 0,000 | 0,000 | -1,000 | BS1 | LK1 | 0,07 | 0,03 | 0,38 | 1,88 | 0,00 | |
| | 1 | 0,000 | 0,000 | -1,000 | BS1 | LK1 | 0,07 | 0,03 | 0,38 | 1,88 | 0,00 | |
| | Gesamt | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,00 | 0,00 | 0,00 |
| 2 | Extrema | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |
| | 1 | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |
| | 10 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,00 | 0,00 | 1,88 | 0,38 | 0,00 | |
| | 1 | 10 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,00 | 0,00 | 1,88 | 0,38 | 0,00 |
| | Gesamt | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |

Abb.8.79: Betonbemessung bezogen auf Erforderliche Bewehrung für Fläche

8.5.4.10 Betonbemessung – Erforderlich Bewehrung für Dicken

Die Abbildung zeigt die tabellarische Darstellung der erforderlichen Längs- und Bügelbewehrung aufgeteilt nach zwei unterschiedlichen Dickenbereichen. Angegeben sind jeweils die Flächennummer, die zugehörigen Netzknoten beziehungsweise Elemente sowie deren Koordinaten in X-, Y- und Z-Richtung. Zudem werden die Bemessungssituation BS1 und die zugehörige Belastung LK1 genannt. Für jede betrachtete Dicke werden die Werte der erforderlichen Längsbewehrung in der ersten und zweiten Richtung separat für die Oberseite und Unterseite der Fläche angegeben. Zusätzlich wird die erforderliche Bügelbewehrung als flächenbezogener Wert dargestellt. Der obere Tabellenblock bezieht sich auf Dicke Nr. 1 und enthält die Angaben für Fläche 1, während der untere Block die Daten für Dicke Nr. 2 mit Fläche 2 enthält. Die Zeilen mit der Bezeichnung „Extrema“ zeigen lokale Werte an bestimmten Netzknoten, während die Zeilen mit der Bezeichnung „Gesamt“ die zusammengefassten Werte für den jeweiligen Dickenbereich wiedergeben.

| Dicke Nr. | Fläche Nr. | Netzknoten / Element Nr. | Netzknoten-Koordinaten [m] | | | Bemess.- Situation | Belastung Nr. | Längsbewehrung [cm ² /m] | | | | Bügel [cm ² /m ²] |
|-----------|------------|--------------------------|----------------------------|-------|--------|--------------------|---------------|-------------------------------------|--------------------|---------------------|---------------------|--|
| | | | X | Y | Z | | | As, erf.1-z (oben) | As, erf.2-z (oben) | As, erf.1-z (unten) | As, erf.2-z (unten) | Bew, erf. |
| Extrema | 1 | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,00 | 0,00 | 0,00 |
| | | 1 | 0,000 | 0,000 | -1,000 | BS1 | LK1 | 0,07 | 0,03 | 0,38 | 1,88 | 0,00 |
| | | 1 | 0,000 | 0,000 | -1,000 | BS1 | LK1 | 0,07 | 0,03 | 0,38 | 1,88 | 0,00 |
| | | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,00 | 0,00 | 0,00 |
| | | 1 | 0,000 | 0,000 | 0,000 | BS1 | LK1 | 1,88 | 0,38 | 0,38 | 1,88 | 0,00 |
| Gesamt | | | | | | | 1,88 | 0,38 | 0,38 | 1,88 | 0,00 | |
| Extrema | 2 | 9 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |
| | | 2 | 0,375 | 0,000 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |
| | | 2 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,00 | 0,00 | 1,88 | 0,38 | 0,00 |
| | | 2 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 0,00 | 0,00 | 1,88 | 0,38 | 0,00 |
| | | 2 | 0,750 | 0,500 | 0,000 | BS1 | LK1 | 2,80 | 0,56 | 0,00 | 0,00 | 6,50 |
| Gesamt | | | | | | | 2,80 | 0,56 | 1,88 | 0,38 | 6,50 | |

Abb.8.80: Betonbemessung bezogen auf Erforderliche Bewehrung für Dicken

8.5.4.11 Betonbemessung – Erforderliche Längsbewehrung Darstellung

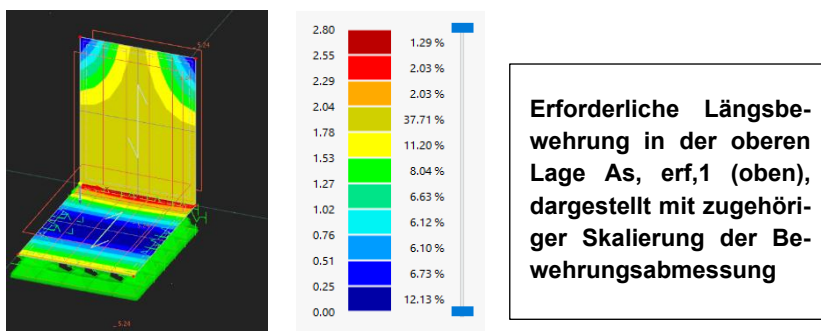


Abb.8.81: Betonbemessung bezogen auf Längsbewehrung As, erf,1 (oben)

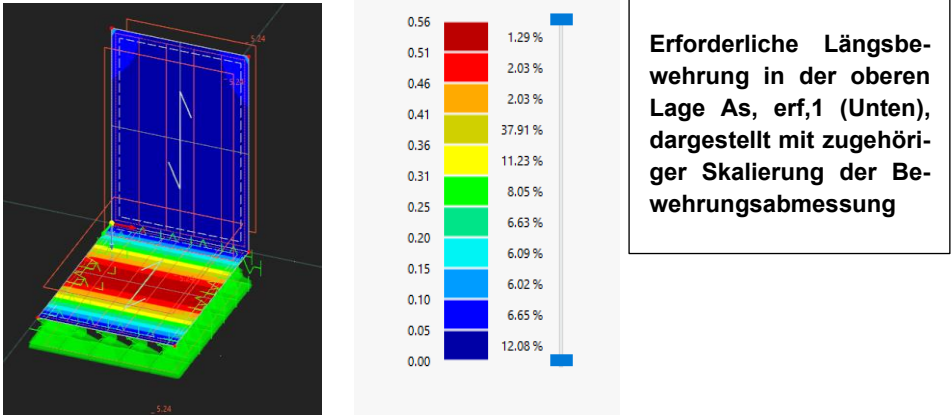


Abb.8.82: Betonbemessung bezogen auf Längsbewehrung As, erf,1 (Unten)

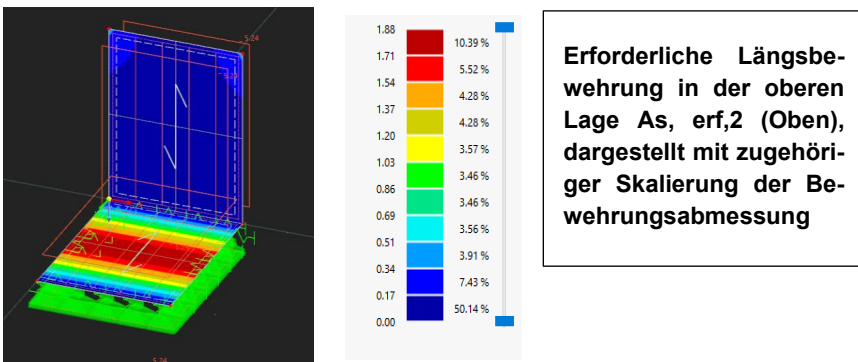


Abb.8.83: Betonbemessung bezogen auf Längsbewehrung As, erf,2 (Oben)

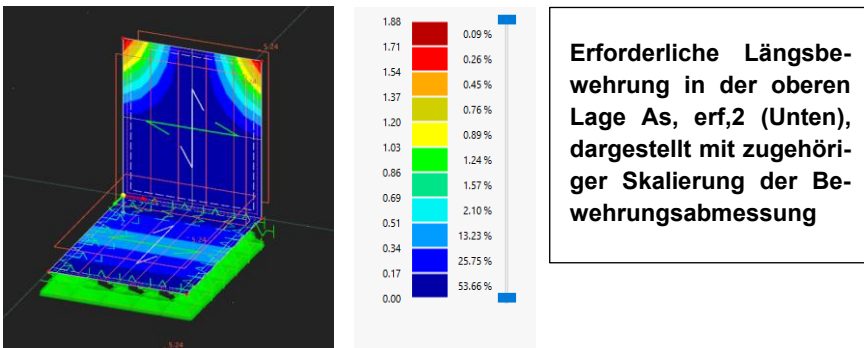


Abb.8.84: Betonbemessung bezogen auf Längsbewehrung As, erf,2 (Unten)

8.5.4.12 Betonbemessung – Erforderliche Bewehrung Werte an Flächen

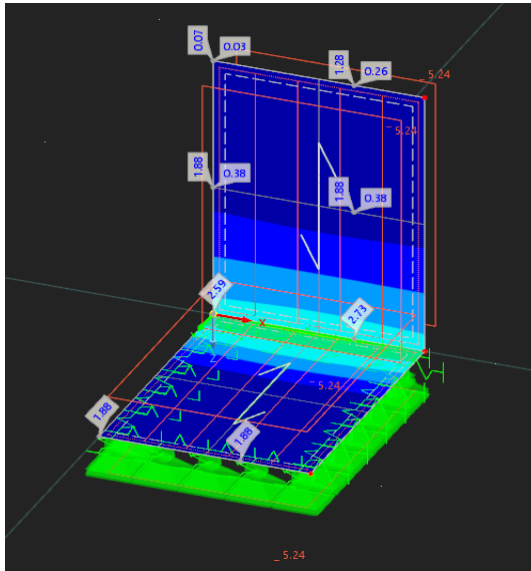


Abb.8.85: Betonbemessung bezogen auf Erforderlich Bewehrung Werte an Flächen (Obere)

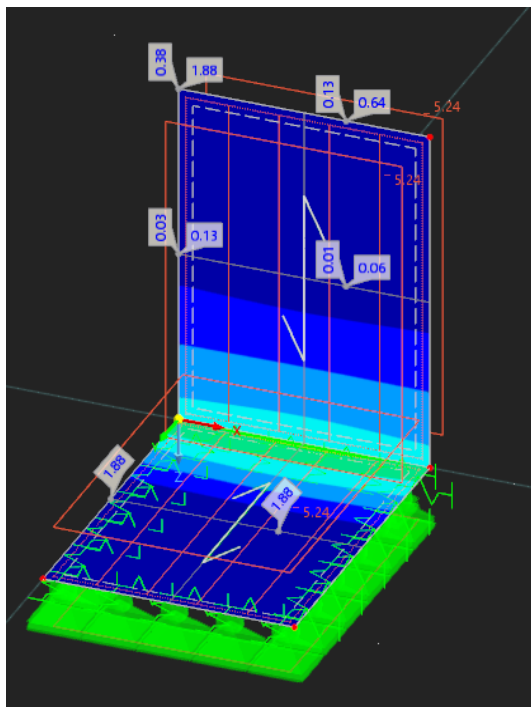


Abb.8.86: Betonbemessung bezogen auf Erforderlich Bewehrung Werte an Flächen (Untere)

8.5.5 Bewertung Der Ergebnisse

8.5.5.1 Bewertung der Einfluss von Dicke und Bewehrung auf die CO₂-Emissionen

- Die Tabelle in Abb.8.74 zeigt die Ergebnisse einer CO₂-Optimierung für ein L-förmiges Fundamentmodell. Ziel ist es, die CO₂-Emissionen durch geeignete Wahl der geometrischen und konstruktiven Parameter zu minimieren. Alle Modelle erreichen denselben CO₂-Zielwert von 750.00 kg-CO₂e, weil Der dargestellte Zusammenhang basiert nicht auf der Verwendung der SetFormula-Funktion innerhalb der Enums im Ordner GlobalParameters. Dieser Befehl ermöglicht es, Parameter wie Dicke oder Bewehrung als wirklich variable Größen zu definieren. Dadurch wird sichergestellt, dass Änderungen dieser Parameter direkten Einfluss auf die Berechnung der CO₂-Emissionen haben. Allerdings wurde die entsprechende Funktionalität in der RFEM-Bibliothek zum Zeitpunkt der Analyse noch nicht vollständig implementiert. Infolgedessen wurden die CO₂-Ergebnisse als konstante Zielwerte betrachtet, ohne dass eine tatsächliche Rückkopplung zu den Eingabeparametern wie Dicke oder Bewehrung erfolgte.
- Die Kombinationen der Eingabeparameter variieren stark – was zeigt, dass es mehrere gleichwertige Lösungen gibt.
- Bewehrungsdurchmesser variieren zwischen 0.008 m (8 mm) und 0.020 m (20 mm).
- Bewehrungsabstand reicht von 0.050 m (sehr eng) bis 0.300 m (weit).
- Manche Modelle (z. B. Modell 5, 6, 7) haben extrem kleine Durchmesser (8 mm) bei relativ großem Bewehrungsabstand (bis zu 0.300 m), was auf minimale Bewehrung zur Reduzierung der CO₂-Menge hindeutet – allerdings evtl. mit Nachteilen für die Tragfähigkeit oder Rissbreitenbegrenzung.

8.5.5.2 Bewertung Betonbemessung – Ausnutzungen an Dicken.

➤ D1, Abb.8.75:

Die Tabelle zeigt die Ergebnisse der Betonbemessung für eine vertikale Fläche mit der Bezeichnung D1 und einer definierten Dicke Nr. 1. Die Nachweise wurden unter der Bemessungssituation B51 und der Lastfallkombination LK1 geführt. Alle Werte sind in Bezug auf das Nachweiskriterium η angegeben, das die Ausnutzung eines Bauteils im Verhältnis zur zulässigen Grenze ($\eta \leq 1.0$) beschreibt.

- Die Nachweise zur Längsbewehrung in der 1. Richtung (oben und unten) liegen bei $\eta = 0.353$ bzw. $\eta = 0.031$, was auf eine sehr gute Sicherheit hinweist.
- Die Druckstrebenfähigkeit der Fläche liegt bei $\eta = 0.015$ bzw. $\eta = 0.040$ – ebenfalls deutlich unter dem Grenzwert.

- Der kritischste Punkt ist beim Schubnachweis mit $\eta = 0.592$ erreicht. Dieser Wert ist zwar zulässig, liegt aber deutlich über den übrigen ULS-Werten und sollte bei weiteren Optimierungen besonders berücksichtigt werden.
- Die Nachweise zur Geometrie und Bewehrungskonfiguration zeigen η -Werte von 0.000 bis 0.015.
- Es gibt keinen Verstoß gegen bautechnische Konstruktionsregeln.
- Der Nachweis zur maximalen Betondruckspannung (C_{max}) ergibt $\eta = 0.222$, was auf eine ausreichende Materialreserve hinweist. Damit ist auch dieser Nachweis sicher erfüllt.

➤ **D2, Abb.8.76:**

Die Tabelle zeigt die Ergebnisse der Betonbemessung für eine horizontale Fläche mit Dicke Nr. 2 unter Lastfallkombination LK1 in der Bemessungssituation B51. Bewertet werden wie zuvor Tragfähigkeit (ULS), Konstruktionsregeln und Materialnachweise über das Nachweiskriterium η .

- Längsbewehrung in der 1. Richtung an der Oberseite zeigt einen Wert von $\eta = 0.535$ → sicher, aber leicht erhöht im Vergleich zu anderen Positionen.
- Schubnachweis (UL 0203.02) fällt negativ auf $\eta = 1.016$ → Grenzwert überschritten. Dies bedeutet, dass die Schubtragfähigkeit der Platte nicht ausreichend ist. Hier besteht akuter Handlungsbedarf – entweder durch:
 1. Erhöhung der Plattendicke,
 2. Gezielte Schubbewehrung oder
 3. Lastumlagerung.
- Alle übrigen Nachweise (z. B. Druckstreben) liegen mit $\eta < 0.01$ weit unterhalb der zulässigen Grenze.
- Betondeckung (DC 0401.00) weist $\eta = 0.333$ auf → innerhalb der Norm, aber höher als bei Dicke Nr. 1.
- Keine weiteren Auffälligkeiten – alle Mindestbewehrungsnachweise in der Zugzone (DM 0300.xx) sind erfüllt mit Werten von $\eta = 0.360$, was eine ausreichende Rissbreitenbegrenzung sichert.
- Nachweise zur Plattendicke, Bewehrungsabständen und Maximalbewehrung zeigen:
 1. $\eta = 1.000$ (z. B. bei maximaler Dicke gemäß Norm)
 2. $\eta = 0.131$ bei Bewehrungsausnutzung in 1. Richtung
 3. $\eta = 0.000$ bei 2. Richtung → keine Bewehrung oder keine Beanspruchung

- Diese Ergebnisse zeigen, dass die Konstruktion formal zulässig ist, jedoch die Dickenbegrenzung vollständig ausgereizt wurde ($\eta = 1.000$).
- Der Wert $\eta = 0.222$ für die maximale Betondruckspannung liegt im sicheren Bereich.

8.5.5.3 Bewertung Betonbemessung – Ausnutzungen an Flächen.

➤ Fläche 1, Abb.8.77:

Im Folgenden werden die Ergebnisse der Nachweisführung für zwei unterschiedliche Flächenbereiche bewertet. Die Berechnung erfolgte jeweils unter der Bemessungssituation BS1 und der Lastfallkombination LK1. Während beide Flächen auf dieselbe Material- und Belastungsgrundlage zurückgreifen, zeigen sich bei der Ausnutzung deutliche Unterschiede – insbesondere im Bereich der Schubtragfähigkeit. Die Nachweise umfassen sowohl den Grenzzustand der Tragfähigkeit (ULS), die Einhaltung von Konstruktionsregeln als auch den Materialnachweis.

- Alle Nachweise liegen unterhalb des Grenzwerts $\eta \leq 1.0$. Der höchste Wert beträgt $\eta = 0.592$ für den Schubnachweis und liegt somit im sicheren, jedoch bereits beanspruchten Bereich. Die Nachweise zur Längsbewehrung und Druckstreben sind sehr gering ausgelastet (zwischen 0.015 und 0.353) und zeigen ausreichende Tragreserven.
- Alle Anforderungen bezüglich Geometrie, Bewehrungsanordnung und Plattendicke sind erfüllt. Der Materialnachweis zur maximalen Betondruckspannung bleibt mit $\eta = 0.222$ ebenfalls im zulässigen Rahmen.

➤ Fläche 2, Abb.8.78:

- Der überwiegende Teil der ULS-Nachweise liegt im unkritischen Bereich (zwischen 0.004 und 0.535). Kritisch ist jedoch der Schubnachweis mit $\eta = 1.016$, womit der Grenzwert überschritten wird. An dieser Stelle ist die Tragfähigkeit nicht gewährleistet.
- Die maximale zulässige Plattendicke wird mit $\eta = 1.000$ erreicht. Damit ist keine geometrische Reserve mehr vorhanden. Die Längsbewehrung liegt bei $\eta = 0.131$ und ist technisch unkritisch.
- Auch hier bleibt der Nachweis zur Betondruckbeanspruchung mit $\eta = 0.222$ im sicheren Bereich.

8.5.5.4 Bewertung Betonbemessung – Erforderliche Bewehrung an Flächen.

Die Tabelle im Abb.8.79 stellt die erforderlichen Bewehrungsmengen (Längs- und Bügelbewehrung) für zwei unterschiedliche Flächenbereiche unter der Bemessungssituation BS1 und der Lastfallkombination LK1 dar. Bewertet werden sowohl die obere und untere Längsbewehrung in zwei Richtungen als auch die flächenbezogene Schubbewehrung (Bügelbewehrung).

➤ Fläche 1

- In Fläche 1 beträgt die erforderliche Längsbewehrung in der 1. Richtung (oben) konstant $1.88 \text{ cm}^2/\text{m}$, ebenso in der 2. Richtung (oben) mit $0.38 \text{ cm}^2/\text{m}$.
- An der Unterseite ist in beiden Richtungen kaum oder keine Bewehrung erforderlich – Werte liegen zwischen 0.00 und $0.38 \text{ cm}^2/\text{m}$.
- Die Bügelbewehrung beträgt $0.00 \text{ cm}^2/\text{m}^2$, was darauf hindeutet, dass keine zusätzliche Querkraftbewehrung erforderlich ist.
- Die Bewehrung ist symmetrisch und moderat ausgelegt, ohne Schubbewehrung. Dies spricht für geringe Beanspruchung durch Querkraft und eine gleichmäßige Momentenverteilung. Die untere Bewehrung ist lokal erforderlich, aber insgesamt gering ausgeprägt.
- Die Bewehrung ist symmetrisch und moderat ausgelegt, ohne Schubbewehrung. Dies spricht für geringe Beanspruchung durch Querkraft und eine gleichmäßige Momentenverteilung. Die untere Bewehrung ist lokal erforderlich, aber insgesamt gering ausgeprägt.

➤ Fläche 2

- Die Längsbewehrung in der 1. Richtung (oben) beträgt hier $2.80 \text{ cm}^2/\text{m}$, in der 2. Richtung (oben) $0.56 \text{ cm}^2/\text{m}$.
- Zusätzlich ist auch an der Unterseite Längsbewehrung notwendig: $1.88 \text{ cm}^2/\text{m}$ in der 1. Richtung und $0.38 \text{ cm}^2/\text{m}$ in der 2. Richtung.
- Die Bügelbewehrung beträgt konstant $6.50 \text{ cm}^2/\text{m}^2$ – dies weist auf eine deutlich höhere Querkraftbeanspruchung hin.
- Fläche 2 ist deutlich stärker beansprucht als Fläche 1. Die benötigte Bewehrung ist sowohl oben als auch unten relevant, was auf eine Wechselwirkung von Momenten und Normalkräften schließen lässt. Die konstante Bügelbewehrung deutet darauf hin, dass die Schubtragfähigkeit ohne zusätzliche Bewehrung nicht ausreicht.

8.5.5.5 Bewertung Betonbemessung – Erforderliche Bewehrung an Dicken.

Die Tabelle im Abb.8.80 zeigt die berechnete erforderliche Längs- und Schubbewehrung getrennt nach den Dicken Nr. 1 und Nr. 2, jeweils unter der Bemessungssituation B51 und der Lastfallkombination LK1. Die Angaben umfassen die Längsbewehrung in zwei Richtungen – getrennt nach Oberseite (-z) und Unterseite (+z) – sowie die benötigte Bügelbewehrung (Schubbewehrung) in cm^2/m^2 .

➤ **Dicke 1**

- Längsbewehrung oben:
In 1. Richtung beträgt die erforderliche Bewehrung $1.88 \text{ cm}^2/\text{m}$, in 2. Richtung $0.38 \text{ cm}^2/\text{m}$.
- Längsbewehrung unten:
In der 1. Richtung ist vereinzelt eine geringe Bewehrung notwendig ($0.07 \text{ cm}^2/\text{m}$), in der 2. Richtung bis zu $1.88 \text{ cm}^2/\text{m}$, jedoch lokal begrenzt.
- Bügelbewehrung (Schub):
 $0.00 \text{ cm}^2/\text{m}^2 \rightarrow$ keine zusätzliche Schubbewehrung erforderlich.
- Die Bewehrung ist weitgehend auf die Oberseite konzentriert, mit geringen lokalen Anforderungen an der Unterseite. Die Schubbeanspruchung ist gering, daher wird keine Bügelbewehrung benötigt. Diese Dicke ist ökonomisch günstig und tragwerkstechnisch effizient.

➤ **Dicke 2**

- Längsbewehrung oben:
 $2.80 \text{ cm}^2/\text{m}$ in der 1. Richtung, $0.56 \text{ cm}^2/\text{m}$ in der 2. Richtung – deutlich höher als bei Dicke 1.
- Längsbewehrung unten:
Es ist ebenfalls eine relevante Bewehrung erforderlich: $1.88 \text{ cm}^2/\text{m}$ in der 1. Richtung, $0.38 \text{ cm}^2/\text{m}$ in der 2. Richtung.
- Bügelbewehrung (Schub):
konstant $6.50 \text{ cm}^2/\text{m}^2$ über alle Punkte hinweg \rightarrow zeigt hohe Querkraftbeanspruchung.
- Die Fläche mit Dicke Nr. 2 ist deutlich stärker beansprucht. Es wird eine symmetrische Bewehrung oben und unten benötigt, und die erforderliche Bügelbewehrung ist hoch. Dies spricht für eine kombinierte Beanspruchung aus Momenten und Querkraft. Die Dicke reicht aus, aber nur mit entsprechend hoher Bewehrungsmenge.

9 Typische Fehlermeldungen und deren Behebung

9.1 Fehlermeldungen

Bei der computergestützten Modellierung und Berechnung von Tragwerken – insbesondere im Zusammenhang mit automatisierten Systemen wie dem KI-gestützten RFEM-Tool – treten häufig bestimmte Fehlermeldungen auf. Diese resultieren oft aus unvollständigen Definitionen, fehlenden Elementverknüpfungen oder falsch initialisierten Eingabewerten. Im Folgenden werden typische Fehlermeldungen aufgeführt, die im Laufe des Projekts beobachtet wurden, sowie die entsprechenden Ursachen und Lösungen.

9.1.1 No surfaces generated – line loop is not closed

- **Problem:**
Diese Fehlermeldung tritt auf, wenn eine Fläche aus Linien erstellt werden soll, die jedoch keine geschlossene Kontur bilden. RFEM erwartet eine konsistente Linie-für-Linie-Definition im Uhrzeigersinn oder gegen den Uhrzeigersinn.
- **Ursache:**
Häufig fehlt eine Linie im Linienzug, oder die Reihenfolge der Linien stimmt nicht überein.
- **Lösung:**
Prüfung der Knotennummern und Sicherstellung der durchgängigen Verbindung der Linien. Automatische Erkennung ungeschlossener Linienzüge und Ergänzung der fehlenden Linie durch das KI-Modell.

9.1.2 Member not assigned to any line

- **Problem:**
Ein Stab wurde definiert, ohne einer gültigen Linie zugewiesen zu sein.
- **Ursache:**
Fehlende oder fehlerhafte Linienzuordnung, häufig durch ungenaue Interpretation des Eingabetexts.
- **Lösung:**
Automatische Zuweisung des Stabs zur passenden Linie bei linienbasierten Bauteilen. Prüfung erfolgt durch das KI-System auf Basis der Liniendefinition.

9.1.3 Invalid material or thickness ID

- **Problem:**
Verweis auf eine Material- oder Dicken-ID, die im Modell nicht vorhanden ist.
- **Ursache:**
Definition des Materials oder der Dicke fehlt oder wurde fehlerhaft generiert.
- **Lösung:**
Vor Beginn der Modellierung erfolgt Prüfung der IDs. Fehlende Parameter werden durch Standardwerte (z. B. C20/25, 0.20 m) ersetzt. Ausgabe einer Warnung erfolgt bei automatischer Ergänzung.

9.1.4 Load direction not recognized

- **Problem:**
Ungültige Angabe der Lastrichtung bei Flächen-, Linien- oder Punktlasten.
- **Ursache:**
Richtungsangabe entspricht nicht den erlaubten RFEM-Konventionen.
- **Lösung:**
Prüfung der Richtungsangabe gegen gültige Enum-Werte. Bei fehlender oder unklarer Richtung wird SurfaceLoadDirection.LOAD_DIRECTION_LOCAL_Z als Standardwert gesetzt.

9.1.5 API key not found or invalid

- **Problem:**
Fehlermeldung tritt auf, wenn der Zugriff auf die RFEM-Webservice-API fehlschlägt, weil kein gültiger API-Schlüssel bereitgestellt wurde.
- **Ursache:**
 1. Kein API-Schlüssel im Skript angegeben
 2. Ungültiger oder abgelaufener Schlüssel
 3. Falsche Konfiguration der Serververbindung (z. B. falscher Port oder URL)
 4. Zugriff auf die API wurde vom Server blockiert
- **Lösung:**
 1. Sicherstellung, dass der gültige API-Schlüssel korrekt im System hinterlegt ist (z. B. in .env-Datei oder als Umgebungsvariable)
 2. Überprüfung der Verbindungseinstellungen (Host, Port, Protokoll)
 3. Bei Bedarf neuen API-Schlüssel im RFEM-Interface generieren
 4. Sicherstellen, dass der API-Client über gültige Zugriffsrechte verfügt
 5. Lokale Firewall-Einstellungen prüfen, um Verbindungsblockaden zu vermeiden

9.1.6 rfem_bot.py nicht gefunden oder falscher Pfad

- **Problem:**
Skript rfem_bot.py wird nicht ausgeführt oder nicht gefunden.
- **Ursache:**
 1. PowerShell befindet sich im falschen Verzeichnis
 2. Skript wurde verschoben oder falsch gespeichert

- **Lösung:**

1. Richtigen Ordner über PowerShell öffnen:

```
Cd "C:\Users\qasho\Desktop\M.Sc\Untersuchungvon KI\RFEM_Python_Client-main\RFEM_Python_Client-main\Examples\GPT_Tank"
```

2. Überprüfung mit:

```
ls
```

Ausgabe zeigt vorhandene Dateien im aktuellen Verzeichnis an.

3. Ausführung des Skripts:

```
python .\rfem_bot.py
```

8.6.7 RFEM API nicht aktiv

- **Problem:**

RFEM-Webservice antwortet nicht oder Verbindung schlägt fehl.

- **Ursache:**

1. RFEM 6 API nicht aktiviert
2. API-Port nicht korrekt (Standard: 8081)
3. Firewall blockiert Verbindung

- ✓ **Lösung:**

1. RFEM öffnen → Optionen → Programmierschnittstellen aktivieren
2. Sicherstellen, dass „WebService & API“ aktiv ist
3. Neustart von RFEM nach Änderung
4. Überprüfung, ob der Port aktiv ist:

```
netstat -ano | findstr :8081
```

Ausgabe muss „LISTENING“ enthalten

Falls kein Ergebnis: anderen Port (z. B. 8082) in RFEM einstellen und erneut starten

8.6.8 Skript nicht dauerhaft verfügbar

- **Problem:**
Skript funktioniert nach Neustart nicht oder Pfad ist ungültig.

- **Ursache:**

Skript liegt in temporärem Verzeichnis oder Desktop-Ordner mit wechselndem Pfad

- **Lösung:**

1. Skript in stabilen Pfad verschieben (z. B. C:\Users\qasho\)
2. Start über absolute Pfadangabe:

```
python C:\Users\qasho\rfem_bot.py
```

8.6.9 Number of mutations to keep not matching or invalid

- **Problem:**
RFEM-Optimierungsmodul meldet Fehler bei der Übergabe des Parameters `number_of_mutations_to_keep`.

- **Ursache:**

Abhängig von der verwendeten RFEM-Version wird eine unterschiedliche Behandlung des Parameters `number_of_mutations_to_keep` erwartet:

1. In älteren RFEM-Versionen ist ein numerischer Wert erforderlich, z. B. 5, um die Anzahl der zu speichernden Mutationen festzulegen.
2. In neueren RFEM-Versionen (z. B. ≥ 6.09) muss der Wert `None` sein, wenn keine explizite Vorgabe gemacht wird – **die Engine erkennt und verwaltet die Mutationen intern.**

- **Lösung:**

Bei Verwendung von `OptimizationSettings(...)` sicherstellen, dass `number_of_mutations_to_keep` abhängig von der RFEM-Version korrekt übergeben wird

8.6.10 Invalid or missing design situation in LoadCombination

Problem:

RFEM meldet Fehler bei der Berechnung oder Erstellung der Lastkombination, weil die zugewiesene **DesignSituation** ungültig, inaktiv oder nicht korrekt referenziert ist.

Ursache:

1. Die angegebene design_situation-ID (z. B. 1) existiert im Modell nicht oder wurde vorher nicht korrekt erstellt.
2. Die **DesignSituation** ist im Modell vorhanden, aber **nicht aktiv** (Parameter active=False).
3. Der zugewiesene Typ (DESIGN_SITUATION_TYPE_STR_PERMANENT_AND_TRANSIENT_6_10) passt nicht zur verwendeten Bemessungsregel oder Analyseart.
4. Fehlerhafte Zuordnung oder Reihenfolge bei gleichzeitiger Verwendung mehrerer Situationen.

• **Lösung:**

1. Sicherstellen, dass DesignSituation(...) vor der LoadCombination(...) erstellt wurde.
2. Aktivierungsstatus prüfen: active=True setzen.
3. Korrekte Typzuweisung nach Eurocode oder nationalem Anhang verwenden

8.6.11 WebFault: Globaler Parameter – Wert außerhalb des erlaubten Bereichs

• **Problem:**

Beim Ausführen von Model.clientModel.service.finish_modification() oder während der Berechnung wird der Vorgang abgebrochen mit einem Serverfehler. Fehlermeldung verweist auf einen **Globalen Parameter**, dessen Wert **außerhalb des zulässigen Bereichs** liegt.

• **Ursache:**

1. Der **Wert des GlobalParameters** (z. B. Wanddicke, Plattenhöhe etc.) wurde außerhalb der in RFEM festgelegten Grenzen eingegeben.
2. Beispiel: Eingabe "Wert = 90" bei erlaubtem Bereich $100 \leq \text{Wert} \leq 300$.
3. Der Fehler tritt erst beim „finish_modification“ auf, da RFEM zu diesem Zeitpunkt alle Eingaben validiert.

✓ **Lösung:**

1. Eingabebereich für den betroffenen Parameter im Voraus prüfen.
2. Grenzwerte durch Abfrage oder Dokumentation des RFEM-Modells ermitteln.
3. Validierung direkt im Skript einbauen – z. B.:

```
if not (100 <= parameter_value <= 300):
    raise ValueError("✗ Ungültiger Wert: Erlaubt sind nur Werte zwischen 100 und 300.")
```

Alternativ: dynamische Einschränkung in der GUI setzen oder API-seitig mit `.set_bounds(...)`, (falls verfügbar).

Beispiel für robusten Parameter:

```
Model.clientModel.service.set_global_parameter(
    GlobalParameter(
        no=1,
        name="d",
        definition_type=GlobalParameterDefinitionType.DEFINITION_TYPE_VALUE,
        value=150, # Gültiger Bereich
        unit="mm"
    )
)
```

8.6.12 SyntaxError: invalid syntax – ungültiger Import

- **Problem:**
Beim Ausführen des Skripts wird ein Syntaxfehler (SyntaxError) ausgelöst, weil eine Import-Anweisung unvollständig ist.

➤ **Fehlermeldung (Beispiel):**

```
from RFEM.BasicObjects.material import
                                     ^
SyntaxError: invalid syntax
```

- **Ursache:**
 1. Der Modulname wurde bei der Import-Anweisung nicht angegeben.
 2. import-Anweisung ist syntaktisch unvollständig.

3. Fehler tritt häufig auf, wenn Codezeilen beim Kopieren abgeschnitten oder nicht korrekt abgeschlossen wurden.

✓ **Lösung:**

1. Importzeile vervollständigen.

Beispiel (korrekt):

```
from RFEM.BasicObjects.material import Material
```

2. Importstruktur regelmäßig prüfen und doppelte oder fehlerhafte Zeilen entfernen.
3. Einheitliche Importgruppen nach Themen gliedern: Grundobjekte, Lager, Lasten, Bemessung, Optimierung usw.

9.2 Hinweis zur Nutzung von KI

Der Einsatz Künstlicher Intelligenz bei der statischen Berechnung bietet erhebliche Potenziale zur Automatisierung und Beschleunigung von Modellierungsprozessen. Um jedoch eine zuverlässige und fehlerfreie Anwendung sicherzustellen, sind bestimmte Richtlinien und Vorgehensweisen zu beachten. Die folgenden Hinweise dienen der gezielten Fehlervermeidung bei der Nutzung eines KI-gestützten Systems in Verbindung mit RFEM.

Empfehlungen zur fehlerfreien Nutzung eines KI-gestützten Systems in Verbindung mit RFEM:

1. Klare und strukturierte Spracheingabe verwenden

Eingaben sollten eindeutig und vollständig sein. Begriffe wie „Platte“, „Stab“, „Lager“ oder „Last“ sind konkret zu benennen. Beispiel:

„Erstelle eine Stahlbetonplatte (3 m × 4 m), 20 cm dick, auf Linienlagern, mit einer Flächenlast von 5 kN/m² in Z-Richtung.“

2. Modellparameter mit zulässigen Wertebereichen definieren

Vor Übergabe an GPT sollten Grenzwerte bekannt und eingehalten werden (z. B. Dicke zwischen 100 mm und 300 mm). Werte außerhalb dieser Bereiche führen zu Abbrüchen bei der Modellgenerierung.

3. Nur gültige RFEM-Enums und Begriffe verwenden

Typen wie `LoadDistribution.LOAD_DISTRIBUTION_UNIFORM` oder `Material-Type.TYPE_CONCRETE` müssen exakt benannt werden. Abweichungen in Groß-/Kleinschreibung oder Struktur führen zu ungültigem Code.

4. RFEM-Version berücksichtigen

Bei Parametern wie `number_of_mutations_to_keep` oder `analysis_type` ist die verwendete Programmversion entscheidend. Unterschiedliche Versionen erwarten unterschiedliche Eingaben oder Datentypen (z. B. None statt Zahl).

5. Globale Parameter vorab prüfen

Vor Verwendung globaler Parameter (z. B. Dicken, Bewehrung, CO₂-Faktoren) ist sicherzustellen, dass alle IDs, Wertebereiche und Einheiten korrekt definiert sind. Empfehlung: Standardwerte setzen und auf Gültigkeit prüfen.

6. Erstellungsschritte logisch aufbauen

Die Reihenfolge von Objekten ist entscheidend: Erst Knoten, dann Linien, dann Flächen oder Stäbe. Flächen dürfen nur mit vollständig geschlossenen Linien erzeugt werden. Die KI sollte diese Struktur im Code einhalten.

7. Prüfung durch Kontrollskripte vor „finish_modification()“

Vor Abschluss des Modells empfiehlt sich ein automatisierter Check auf:

- nicht definierte Materialien, Dicken oder Lager
- ungeschlossene Flächen
- ungültige Lastverteilungen
- leere Parameter oder fehlerhafte Zuordnungen

8. Verwendung von GPT-internen Templates und Beispielen

Zur Sicherstellung korrekter Syntax und Strukturen sollten alle Eingaben durch vorab geprüfte Templates laufen. Fehlerhafte oder unklare Nutzerangaben können dadurch automatisch angepasst oder ergänzt werden

9. Ergebnisse nach der Berechnung validieren

Auch bei erfolgreicher Berechnung ist eine Plausibilitätsprüfung der Ergebnisse erforderlich – insbesondere bei Extremwerten (z. B. ungewöhnlich hohe Spannungen, negative Dickenwerte, un stabile Eigenformen).

10 Vorteile und Nachteile der KI-gestützten statischen Berechnung

Der Einsatz von Künstlicher Intelligenz in der Tragwerksplanung eröffnet neue Möglichkeiten zur Automatisierung und Effizienzsteigerung. Gleichzeitig bringt diese Methode auch bestimmte Herausforderungen mit sich. Die folgende Übersicht stellt die wesentlichen Vorteile und Nachteile zusammenfassend gegenüber.

10.1 Vorteile:

- **Automatisierte Modellgenerierung**
Wiederkehrende Prozesse wie das Erstellen von Platten, Trägern, Materialien oder Lasten können automatisiert durch Spracheingabe erfolgen. Dies reduziert manuelle Eingaben und spart Zeit.
- **Fehlerprävention durch strukturierte Templates**
Die Verwendung von vorgefertigten Codebausteinen (Templates) minimiert typische Eingabefehler bei der Definition von Objekten, Lasten und Parametern.
- **Schnelle Variantenbildung**
Verschiedene Entwurfsvarianten (z. B. mit unterschiedlichen Dicken, Materialien oder Bewehrungen) können durch kleine Änderungen im Prompt effizient erstellt und berechnet werden.
- **Integration mit Optimierung und Nachhaltigkeit**
Zielgrößen wie CO₂-Emissionen oder Kosten lassen sich automatisiert analysieren und bewerten, wodurch nachhaltige Tragwerkslösungen gefördert werden.
- **Effiziente Kommunikation mit der API**
GPT kann vollständigen Python-Code für RFEM generieren, inklusive API-Logik – ohne manuelles Codieren.
- **Skalierbarkeit**
Geeignet für kleine Einzelbauteile bis hin zu komplexen Strukturmodellen mit mehreren Komponenten.

10.2 Nachteile:

- **Abhängigkeit von korrekten Prompts**
Unklare oder unvollständige Eingaben führen zu fehlerhaftem Code. Fachwissen ist erforderlich, um sinnvolle Spracheingaben zu formulieren.
- **Keine automatische fachliche Prüfung**
Die KI prüft nicht automatisch auf normgerechte Nachweise oder ingenieurtechnische Plausibilität. Eine fachliche Kontrolle durch den Nutzer ist zwingend erforderlich.
- **Versionsabhängigkeiten der API**
Änderungen in der RFEM-Webservice-API können zu Inkompatibilitäten führen. Bestimmte Parameter (z. B. None vs. Integer) sind versionspezifisch.
- **Keine Garantie für Stabilität oder Berechenbarkeit**
Trotz korrekter Syntax können Modelle instabil oder unvollständig sein, wenn die Logik (z. B. Lagerung, Randbedingungen) nicht sinnvoll umgesetzt wurde.
- **Eingeschränkte Erkennung komplexer Systeme**
KI ist aktuell besser für einfache bis mittelschwere Strukturen geeignet. Komplexe Systeme mit nichtlinearem Verhalten, Vorspannung oder Zwischenschritten sind nur begrenzt umsetzbar.

11 Zusammenfassung und Ausblick

11.1 Zusammenfassung:

Die vorliegende Arbeit untersuchte die Möglichkeiten der KI-gestützten Modellierung und Optimierung statischer Systeme am Beispiel eines L-förmigen Fundaments in RFEM. Ziel war die Entwicklung eines Systems, das durch natürliche Spracheingabe ein vollständiges Tragwerksmodell erzeugt, dieses berechnet und anschließend die Ergebnisse automatisiert auswertet. Dazu wurde ein auf GPT basierender KI-Ansatz mit der RFEM-Webservice-API kombiniert.

Durch die intelligente Verarbeitung von Benutzereingaben konnte die KI in der Lage sein, eigenständig Modellierungsentscheidungen zu treffen – z. B. das Erzeugen von Knoten, Linien, Flächen, Materialien, Dicken, Lasten, Lagerungen und Kombinationen. Zusätzlich wurden globale Parameter zur Steuerung von Dicken und Bewehrungswerten definiert, die eine gezielte Optimierung hinsichtlich der CO₂-Emissionen ermöglichten. Die Struktur des Modells sowie die Berechnungslogik wurden automatisch generiert, ausgeführt und in tabellarischer Form dokumentiert.

Besonders hervorgehoben werden kann die Flexibilität des Systems in der Variantenbildung. Mehrere Fundamentszenarien konnten mit minimalem Aufwand durch Anpassung der Spracheingabe generiert und berechnet werden. Die Auswertungen zu Materialverbrauch, Bewehrung und CO₂-Ausstoß wurden einheitlich strukturiert ausgegeben, wodurch eine fundierte technische Bewertung möglich war. Dabei zeigte sich, dass durch gezielte Parametervariation eine Reduktion der CO₂-Bilanz bei gleichzeitiger Wahrung der Tragfähigkeit erreichbar ist.

Die Arbeit zeigt, dass der Einsatz von KI im Bereich der Tragwerksplanung nicht nur möglich, sondern sinnvoll ist – insbesondere für die frühe Entwurfsphase, die Variantenanalyse und die Bewertung von Nachhaltigkeitsaspekten. Die Verbindung von Text-Eingabe, Python-Code-Generierung, API-Kommunikation und Ergebnisverarbeitung wurde in einem geschlossenen Workflow erfolgreich umgesetzt.

11.2 Ausblick:

Auf Basis der vorliegenden Ergebnisse ergeben sich konkrete Weiterentwicklungsmöglichkeiten innerhalb des untersuchten Themenfelds. Eine naheliegende Erweiterung besteht in der Einbindung zusätzlicher Optimierungsgrößen, wie z. B. Tragfähigkeitsreserven, Verformungsgrenzen oder minimaler Bewehrungsanteile – stets in Kombination mit der bestehenden CO₂-Analyse.

Auch eine tiefere Integration in den Bemessungsprozess ist denkbar, z. B. durch eine automatisierte Übergabe der berechneten Schnittgrößen an ein statisches Nachweisskript. Ebenso wäre die Erweiterung des GPT-Modells um typische Fehlermuster denkbar, damit bereits während der Eingabe automatisch auf unplausible Angaben hingewiesen werden kann.

Ein weiterer sinnvoller Schritt wäre die automatische Kombination von mehreren Fundamenttypen innerhalb eines Modells – etwa bei unterschiedlichen Lastfällen oder Bodenverhältnissen. Dies würde die Anwendbarkeit des Systems auf reale Projekte erweitern.

Insgesamt bietet die Methode großes Potenzial für zukünftige Planungsaufgaben, bei denen Variantenanalyse, Nachhaltigkeit und Automatisierung im Vordergrund stehen. Die in dieser Arbeit entwickelte Lösung liefert hierfür eine solide und direkt einsetzbare Grundlage.

11.3 Persönliche Meinung

Meiner Meinung nach ist dieses Thema äußerst spannend und bietet großes Potenzial für die Zukunft der Tragwerksplanung. Besonders beeindruckt mich, dass Künstliche Intelligenz (KI) zunehmend komplexe Aufgaben wie statische Berechnungen nahezu vollständig übernehmen kann – von der Eingabe bis zur Ergebnisanalyse. Das verändert unsere Arbeitsweise grundlegend und eröffnet neue Möglichkeiten, Prozesse effizienter, präziser und sicherer zu gestalten.

KI spielt heute eine wichtige Rolle in vielen Lebensbereichen und hat längst auch die Bau- und Ingenieurbranche erreicht. Während früher alle Berechnungen manuell oder mit spezieller Software durchgeführt werden mussten, können heute viele Arbeitsschritte durch intelligente Systeme automatisiert werden. Dies spart nicht nur Zeit und Kosten, sondern verbessert auch die Qualität und ermöglicht eine bessere Kontrolle komplexer Projekte.

Am Beispiel von Dlubal RFEM wird deutlich, welches große Potenzial diese Technologien besitzen, auch wenn aktuell noch nicht alle Funktionen über die API vollständig verfügbar sind. Einige wichtige Bibliotheken und spezifische Berechnungsroutinen fehlen noch, doch die kontinuierliche Weiterentwicklung zeigt klar, dass diese Lücken in absehbarer Zeit geschlossen werden sollen.

Ich bin überzeugt, dass KI in der Tragwerksplanung künftig eine noch größere und unverzichtbare Rolle spielen wird. Komplexe Modelle, Simulationen und Auswertungen könnten dann auf Knopfdruck zuverlässig und nachvollziehbar erstellt werden. Die vollständige Integration in einen digitalen und automatisierten Bauprozess wird die Effizienz, Qualität und Nachhaltigkeit unserer Bauwerke nachhaltig verbessern.

Literaturverzeichnis

Quelle: Babel, T. (2024): *Künstliche Intelligenz – Lösungsansätze*, S. 17–18, 123.

Quelle: Babel, T. (2024): *Künstliche Intelligenz – Lösungsansätze*, S. 41–44

Quelle: Koch, C. (2021): *Künstliche Intelligenz im Bauwesen*, Springer Vieweg, S. 38–42

Quelle: Babel, T. (2024): *Künstliche Intelligenz – Lösungsansätze*, S. 45–49

Quelle: Babel, T. (2024): *Künstliche Intelligenz – Lösungsansätze*, S. 123

Quelle: Koch, C. (2021): *KI im Bauwesen*, S. 103–108

Quelle: RFEM-Dlupal Bibliothek

