Fachbereich III – Bauingenieur- und Geoinformationswesen
Konstruktiver Hoch- und Ingenieurbau (M.Eng.)
Ingenieurholzbau

Erweiterung der Ingenieuranwendungen von Excel mit Hilfe der Programmiersprache Python

Abschlussarbeit

Zur Erlangung des Grades eines Master of Engineering

Eingereicht von: Cal Mense
Matrikel-Nr.: 914553
Am: 26.09.2022
Betreuer: Prof. Dr. Jens Kickler
Gutachter: Prof. Dr. Volker Dick

Eidesstattliche Erklärung

Ich, Cal Mense, Matr.-Nr. 914553
Versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen habe, als solche kenntlich gemacht habe und mich auch keiner als der angegebenen Quellen oder sonstiger Hilfsmittel bedient habe.
Ich versichere an Eides statt, dass ich die vorgenannten Angaben nach bestem Wissen und Gewissen gemacht habe und dass die Angaben der Wahrheit entsprechen und ich nichts verschwiegen habe.

Berlin, den 26.09.2022

Ich danke meinen Eltern, Takako und Wolfgang für Ihre aufrichtige und allgegenwärtige Unterstützung während meines Studiums.

Kurzfassung

Im Ingenieurbereich und speziell im Bauwesen lassen sich viele Berechnungen mit Hilfe von Exceldatenblättern automatisieren. Excel bietet hierzu eine Vielzahl von Möglichkeiten insbesondere durch die Nutzung von Verknüpfungen und einfachen Programmierungen in der Befehlszeile. Dieses Vorgehen kommt aber bei etwas komplexeren Berechnungen oder Parameterstudien an Grenzen. Hierzu gibt es in Excel die Möglichkeit eigene Programmierungen mit VBA zu erstellen. Eine weitere Möglichkeit ergibt sich, wenn die Programmierungen außerhalb der Excelumgebung erfolgen. Mit der Unterstützung von Python können Werte aus Exceldatenblättern herausgelesen, verarbeitet und die Ergebnisse wieder in ein Datenblatt zurückgeschrieben werden. Diese Methode verbindet geschickt den Freiraum einer Programmiersprache mit der übersichtlichen Darstellung von Daten in Excel.

Abstract

This study aims to evaluate and present the possibilities of integrating Python programming with several application examples in timber engineering.

In engineering and particularly in structural engineering, Excel spreadsheets can automatise many forms of calculations. Excel offers various possibilities for creating intuitive and effective spreadsheet programs by linking the cells and using simple formulas.
However, the functionality of Excel quickly reaches its limit with complex tasks, such as the utilisation of parameter studies.
To increase the productivity of Excel, the built-in programming language "Visual Basic (VBA)" could be used as an alternative. Another approach is to connect Excel with another programing language, such as Python. This enables the adjustments of the data within the Excel program and allows the integration of a modern and easy-to-learn programming language with the structured and familiar interface of Excel.

Vorwort

Die vorliegende Abschlussarbeit behandelt zwei Themenbereiche des Holzingenieurbaus, die ich als junger Ingenieur in der Berufspraxis behandelt habe.

Zum einen bestand meine Aufgabe im Rahmen einer Planung eines Bürogebäudes darin, etwa eintausend Stützen zu bemessen. Nach einem Erkunden für die gängige Vorgehensweise bei dieser Aufgabe, war ich erstaunt, dass dies üblicherweise mit einer aufwendigen Typisierung bewerkstelligt wird. Die Typisierung aller Stützen des Gebäudes ergaben ca. 60 verschiedene Stützentypen, die je nach Lage, Abmessungen, Länge und Einwirkungen gruppiert wurden. Berechnet wurden sie mit Einzelnachweisen mithilfe des Programms Frilo (Friedrich + Lochner). Zusammen mit der Dokumentation für die abzugebende prüfbare Statik beendete ich diese Aufgabe in annähernd drei Monaten.

Im weiteren Verlauf des Projektes war es anschließend meine Aufgabe, mich den Bauteilanschlüssen zu widmen. Das Ziel dabei war es, im Sinne einer Rückbaubarkeit möglichst einfache Stahlverbindungen mit Vollgewindeschrauben zu konzipieren. Die Bemessung der Holzschrauben erfolgte anhand firmenspezifischer Produktblätter, die teilweise aus über hundert Seiten bestanden.

Wieder sah ich mich also in einer Situation, in der ich dieselben Arbeitsschritte teils hundertfach wiederholen musste. Sowohl bei der Stützenbemessung als auch bei der Schraubenbemessung sah ich daher ein großes Potenzial für die Automatisierung des Arbeitsablaufes. Dies würde nicht nur in einem geringeren Arbeitsaufwand resultieren, sondern würde ebenfalls eine viel genauere Bemessung, eine deutlich höhere Zeiteffizienz und dadurch hohe Kosteneinsparungen ergeben.

Aus diesen praktischen Erfahrungen heraus ergab sich mir die Motivation, die Anwendbarkeit von Python anhand dieser Anwendungsbeispiele zu evaluieren und aufzuzeigen, wie die Produktivität der Statik mit der Unterstützung Pythons gesteigert werden kann. Die Abschlussarbeit begann ich mit sehr wenigen Vorkenntnissen im Programmieren, sodass ich mir sämtliche Kenntnisse während der Bearbeitung angeeignet habe. Besonders wichtig war mir eine möglichst zugängliche und interaktive Aufbereitung der Materie, um der weitverbreiteten Befangenheit gegenüber dem Programmieren zu entgegnen.

Inhaltsverzeichnis

Eidesstattliche Erklärung **2**

Danksagung **3**

Kurzfassung **4**

Vorwort **5**

1. Einführung **9**
1.1 Motivation der Arbeit 9
1.2 Herangehensweise 10
1.2.1 Formate 11
1.2.2 Ordnerstruktur 12
1.3 Überblick der Anwendungsbeispiele 13
1.3.1 Integration von Python in Excel 13
1.3.2 Entwicklung einer Webapplikation mit Streamlit 14
1.3.3 Erstellung von interaktiven Datenblättern mit Jupyter Notebook 15

2. Programmierung in Python **17**
2.1 Merkmale 17
2.2 Vorbereitungsmaßnahmen 18
2.2.1 Entwicklungsumgebung: Anaconda 18

3. Anwendungsbeispiel 1: Stützenbemessung **25**
3.0 Einführung 25
3.1 Grundlagen zur Stützenbemessung im Holzbau 30
3.1.1 Holzbau 30
3.1.2 Stützenbemessung 31
3.2 Stabilitätsnachweis nach dem Ersatzstabverfahren 34
3.2.1 Allgemeines 34
3.2.2 Eingangswerte 35
3.2.3 Berechnung 38

3.2.4 Nachweise ... 44
3.3 Stabilitätsnachweis nach der Theorie II. Ordnung ... 48
3.3.1 Vorgehensweise nach EC5 ... 49
3.3.2 Vorgehensweise in Python ... 51
3.3.3 Grafische Auswertung der Ergebnisse ... 56
3.3.4 Nachweise ... 59
3.4 Erstellung von Funktionen ... 60
3.5 Verknüpfung mit Excel ... 60
3.5.1 Bibliothek: xlwings ... 60
3.6 Parameterstudie ... 69
3.6.1 Bemessung nach Theorie I. und II. Ordnung ... 69
3.6.2 Variable: Schlankheit ... 69
3.6.3 Theorie II. Ordnung: Gegenüberstellung mit Näherungslösung ... 71
3.6.4 Gegenüberstellung der Bemessungsverfahren ... 73
3.7 Optimierungsalgorithmus ... 73
3.7.1 Schematischer Ablauf ... 74
3.7.2 Detaillierter Ablauf ... 75
3.8 Dokumentation ... 78
3.8.1 Grafische Dokumentation ... 78
3.8.2 Schriftliche Dokumentation ... 80

4. Anwendungsbeispiel 2: Schraubenbemessung ... 81
4.0 Einführung ... 81
4.1 Grundlagen zur Bemessung von Holzschrauben ... 84
4.1.1 Einleitung ... 84
4.1.2 Axialtragfähigkeit ... 85
4.1.3 Ausziehwiderstand ... 88
4.1.4 Kopfdurchziehwiderstand ... 89
4.1.5 Zugfestigkeit ... 89
4.1.6 Resultierende Axialtragfähigkeit ... 89
4.1.7 Tragfähigkeit auf Abscheren ... 90
4.1.8 Einschnittige Holz-Holzverbindung ... 92
4.1.9 Einschnittige Holz-Stahlverbindung (dickes Blech) ... 94
4.1.10 Einschnittige Holz-Stahlverbindung (dünnes Blech) ... 95

4.2 Erstellung einer Funktion 96
4.3 Entwicklung einer Web-App mit Streamlit 98
4.3.1 Einführung 98
4.3.2 Erste Schritte 98
4.3.3 Elemente in Streamlit 99
4.3.4 Bibliothek: Plotly 101
4.3.5 Erstellung eines Einzelnachweises 108
4.4 Vollständiger Code 109
4.4.1 Schematischer Ablauf 109

5. Fazit 111

Abbildungsverzeichnis 113

Tabellenverzeichnis 115

Codeverzeichnis 116

Literaturverzeichnis 118

Anhang ..... Error! Bookmark not defined.

1. Einführung

1.1 Motivation der Arbeit

Das innovative Konzept der Einfachheit und Übersichtlichkeit macht Python seit Jahren zu eine der beliebtesten Einsteigerprogrammiersprachen und wird zunehmend an Hochschulen und Universitäten gelehrt. Python ist kostenlos nutzbar und bietet eine umfangreiche Standardbibliothek, die mit einer Vielzahl an Zusatzmodulen erweitert werden kann. Somit präsentiert Python für fast jedes erdenkliche wissenschaftliche Problem eine zugeschnittene Sammlung von Funktionen an und macht das Programmieren mit Python vor allem in den Bereichen der Datenverarbeitung, der Web-Entwicklung und der Ingenieurswissenschaften zu einer spannenden Innovation. Mit wenig Aufwand können effiziente grafische Benutzeroberflächen erstellt oder Ergebnisse umfangreicher Berechnungen übersichtlich präsentiert werden (vgl. Steyer, 2017: 3).

Im Bereich des kommerziellen Bauwesens wird Python in Deutschland trotz alledem erst sehr wenig angewandt. Besonders effizient lässt sich das Programmieren bei sich wiederholenden Prozessen einsetzen. Das Automatisieren einfacher Berechnungen oder kognitiver Routinearbeiten ersparen langfristig gesehen eine Menge Zeit und minimieren dabei die Fehleranfälligkeit.

Die Bearbeitung von immer größeren Datenmengen im Zeitalter der Digitalisierung stellt eine Herausforderung dar, die auch im Bauwesen zu bewerkstelligen gilt. Die rasante technologische Entwicklung in allen beruflichen Bereichen wird dazu führen, dass es ebenso für die Ingenieure wichtig werden wird, Kenntnisse im Programmieren zu erlernen. Seit der Einführung von computergestützten statischen Berechnungsverfahren entwickelt sich der Anspruch im Bauwesen dahingehend, dass Bauwerke zunehmend komplexer und anspruchsvoller werden. Die hohen Anforderungen werden aus den Folgen des Klimawandels, wie die daraus resultierende Ressourcenknappheit befeuert, sodass ökonomisches und effizientes Bauen unter Einbeziehung ökologischer Aspekte wichtiger ist denn je.

Im Zuge dessen erfährt auch der Holzbau aufgrund seiner Nachhaltigkeit wieder zunehmend an Popularität und an Vertrauen in der Gesellschaft. Als einzig

erneuerbarer Bauwerkstoff wird mit dem Einsatz von Holz eine nachhaltige Entscheidung getroffen, die viele Vorteile mit sich bringt. Neben einer sehr guten ökologischen Bilanz besitzt Konstruktionsvollholz zudem hervorragende mechanische Eigenschaften und schafft ein gesundes Raumklima. Eines der entscheidenden Eigenschaften des Materials ist die Bindung des klimaschädlichen Kohlenstoffs und dessen Umwandlung in Sauerstoff. Dadurch leistet Holz einen wichtigen Beitrag zum Klimawandel (vgl. Neuhaus 2017: 1).

Aus diesem Grund werden im Rahmen dieser Arbeit das Potenzial von Python mit Hilfe verschiedener Anwendungsmöglichkeiten praxisnah erläutert. Das Ziel besteht darin, Stärken und Schwächen von Python anhand dieser Anwendungsbeispiele aufzuzeigen und dabei eine Schritt-für-Schritt Anleitung zu erstellen. Die Beispiele sind so gewählt, dass sie möglichst simpel in die alltägliche Arbeitsweise eines Statikers integriert und zu eigenen Anwendungen und Skripten weiterentwickelt werden können.

1.2 Herangehensweise

Der Anspruch dieser Arbeit ist darin definiert, dass es grundsätzlich als Einsteigerkurs für die Programmiersprache Python konzipiert ist. Anhand von Anwendungsbeispielen werden die Grundlagen des Programmierens aufgezeigt und verschiedene Algorithmen sukzessive aufgebaut, sodass die Arbeit zusammenhängend gelesen werden sollte.

Für das Programmieren von eigenen Anwendungen mit Python sind zu Beginn nur eine Handvoll an Grundlagen notwendig, die immer wieder zum Einsatz kommen. Es gibt ein unzähliges Angebot an verschiedenen kostenlosen Seminaren und Tutorien im Internet, die auf Deutsch und Englisch angeboten werden, die die theoretischen Grundlagen ausführlich und umfangreich erklären. Daher wird im Rahmen dieser Arbeit ein praktischer Ansatz gewählt, wobei das Programmieren mit bekannten ingenieurtechnischen Problemstellungen aufgeschlüsselt wird. Anhand dessen wird der Leser dazu angehalten, trotz weniger Vorkenntnisse die angeführten Beispiele für eigene Zwecke anzuwenden. Erst durch die praktische Anwendung sind schnelle und nachhaltige Erfolge garantiert.

1.2.1 Formate

Die Arbeit wird als gebundenes Exemplar und als PDF-Datei abgegeben. Zusätzlich wird die gesamte Arbeit im Sinne der Quelloffenheit inhaltsgleich in Form von Web-basierten und interaktiven Arbeitsblättern auf der Cloud-basierten Entwicklungsplattform **GitHub** online bereitgestellt. Diese werden mit der Programmierumgebung **Jupyter Notebook** verfasst und können als alleinstehendes Dokument gelesen werden.

Das Erstellen und Freigeben von wissenschaftlichen Dokumenten mit Jupyter Notebook eignet sich besonders, um Code mit Texten und Visualisierungen zu erklären. Es wird ermöglicht, dass die demonstrierten Algorithmen aktiv von Nutzern ausprobiert werden können und führt dazu, dass die Kenntnisse auf einer interaktiven Art und Weise vermittelt und erlernt werden.

Link zu GitHub:

https://github.com/calmense

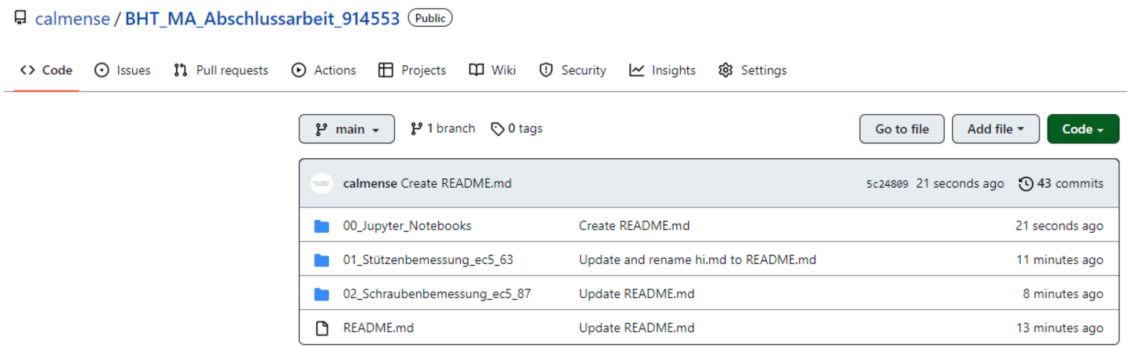


Abbildung 1: Ausschnitt aus GitHub

Die Abbildung zeigt die drei Repositorien der Plattform GitHub. Die Arbeitsblätter im Ordner *00_Jupyter_Notebooks* haben das Format „.ipynb“ und können dort geöffnet und gelesen werden. Um die Programmierungen anwenden zu können, müssen die Ordner in der Gesamtheit als Zip-Datei heruntergeladen werden. Die Jupyter Notebook-Dateien können schließlich in der Entwicklerumgebung **Anaconda Distribution** für eigene Zwecke ausprobiert und weiterentwickelt werden. Die Installation von Anaconda wird in Kapitel 2.2 behandelt. Die Ordnerstruktur darf

aufgrund der Datei-Abhängigkeiten *nicht* verändert werden. Im weiteren Verlauf wird auf die zugehörigen Arbeitsblätter verwiesen.

1.2.2 Ordnerstruktur

Folgende Dateien sind den Repositorien zu entnehmen:

00_Jupyter_Notebooks:

- Thematisch gegliederte Spiegelung der Masterarbeit als *Jupyter Notebooks*
- Liste der Bibliotheken: requirements.txt

01_Stützenbemessung_ec5_63:

- Excel-Stützenbemessungsprogramm: Stützenbemessung_ec5_63.xlsm
- Ordner: **00_Programmhandbuch**:
    - Stützenbemessung_ec5_63_Handbuch.ipynb
- Ordner: **01_Funktionen:**
    - Stützenbemessung_ec5_63.py (Quellcode)
- Ordner: **02_PDF_Ausgaben:**
    - Stützenbemessung_ThII_S-201.pdf (Beispielausdruck)
- Ordner: **03_Statiknachweise:**
    - ec5_63_Stützenbemessung_Ersatzstabverfahren.ipynb
    - ec5_63_Stützenbemessung_Theorie II. Ordnung.ipynb

02_Schraubenbemessung_ec5_87:

- Link zum Schraubenbemessungsprogramm
- schraubenbemessung_ec5_87.py (Quellcode)
- schraubenbemessung_ec5_87_functions.py (Quellcode)
- Liste der Bibliotheken: requirements.txt
- Grafik-Dateien
- Ordner: **00_Programmhandbuch**:
    - Schraubenbemessung_ec5_87_Handbuch.ipynb
- Ordner: **01_Statiknachweise:**
    - ec5_87_Schraubenbemessung_Einzelnachweis.ipynb
    - ec5_87_Schraubenbemessung_Bemessungstabellen.ipynb

1.3 Überblick der Anwendungsbeispiele

Die Anwendungsbeispiele lassen sich grob in folgende 3 Themen aufteilen:

1. **Integration von Python in Excel**
   Erstellung eines Stützenbemessungsprogramms nach dem Eurocode 5
2. **Entwicklung einer Webapplikation mit Python und Streamlit**
   Programm für die Berechnung von Tragfähigkeiten von Vollgewindeschrauben
3. **Erstellung von interaktiven Datenblättern mit Jupyter Notebook**
   Statikberechnungsvorlage für Bemessungen nach dem Eurocode 5

Abbildung 2: Verwendete Programme für Anwendungsbeispiele

1.3.1 Integration von Python in Excel

Als Ingenieur ist es stets von großer Bedeutung, die Berechnungen inhaltlich zu verstehen und Ergebnisse anhand des eigenen Fachwissens zu hinterfragen. Trotz der Anwendung diverser Programme und Softwares sollte das Verständnis und das Vertrauen der Berechnungen niemals geringer ausfallen.

Das Arbeiten mit dem Tabellenkalkulationsprogramm Microsoft Excel bietet aus diesen Gründen enorme Vorteile als Ingenieur, um leistungsstark und effizient Dateien zu erstellen und zu bearbeiten. Es können intuitive Berechnungstabellen erzeugt werden, die flexibel angepasst und von anderen Nutzern leicht nachvollzogen werden können. Vor allem in der Buchhaltung, der Finanzwelt und im Ingenieurswesen ist Excel nicht mehr wegzudenken und genießt ein hohes Maß an Vertrauen bei den Nutzern.

Bei komplexeren und größeren Datenmengen kann es jedoch dazu führen, dass sich die Ausführung von Excel verlangsamt und das Arbeiten dadurch stark beeinträchtigt wird. Eines der Hauptursachen stellen eine hohe Anzahl an Formeln und

Zellenreferenzen dar, die in großen Tabellen dazu führt, dass die Geschwindigkeit der Datenverarbeitung dadurch abnimmt.

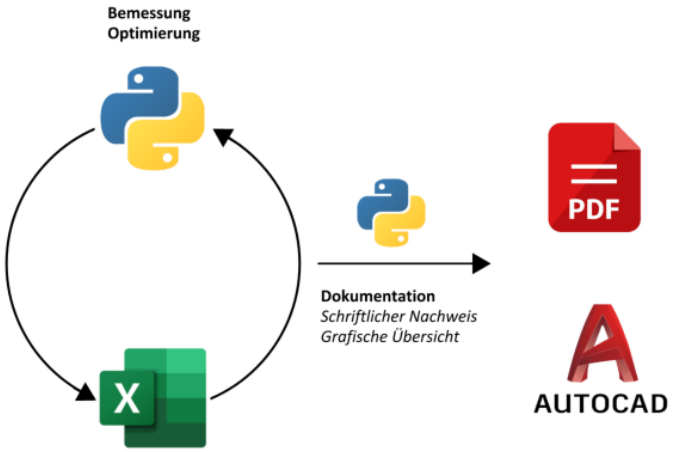


Abbildung 3: Schematischer Ablauf des Stützenbemessungsprogramms mit Python

Aus diesem Grund soll im ersten Anwendungsbeispiel aufgezeigt werden, wie durch das Verknüpfen von Excel mit Python ein Stützenbemessungsprogramm entsteht, welches die aufgeführten Aspekte miteinander vereint. Im Sinne der Nachvollziehbarkeit sollen Zwischenergebnisse angezeigt werden und es soll hinzu ermöglicht werden, dass durch das Nutzen von Eingabeparametern eine Interaktion mit dem Nutzer erzielt wird. Die Berechnung selbst übernimmt das Python-Skript, welches innerhalb der Excel-Oberfläche mit einem Schalter ausgeführt wird. Das Ergebnis wird anschließend an Excel übermittelt, sodass keine komplizierten Formeln benötigt werden.

Das Stützenbemessungsprogramm soll einen gesamten Arbeitsablauf von der Bemessung, Optimierung, bis hin zur Dokumentation abbilden.

1.3.2 Entwicklung einer Webapplikation mit Streamlit

In diesem Anwendungsbeispiel soll ein Programm entwickelt werden, welches die Tragfähigkeiten von Vollgewindeschrauben des Herstellers Würth berechnet. Die Berechnungen der Tragfähigkeiten eines Verbindungsmittels hinsichtlich einer Axial- und Scherbeanspruchung ist in der DIN EN 1995-1-1 Absatz 8 geregelt. Durch die hohe Anzahl der Einflussparametern und herstellerspezifischen Materialkennwerten erweist sich die Bemessung als vergleichsweise komplex und kompliziert. Aus diesem Grund gibt es herstellereigene Programme, die verschiedene Holzverbindungen

nachweisen. Als Alternative werden Bemessungstafeln angeboten, anhand dessen die Axial- und die Schertragfähigkeiten der Schrauben unter Berücksichtigung gewisser Randbedingungen, wie Seitenholzdicke oder Holzgüte abgelesen werden können.

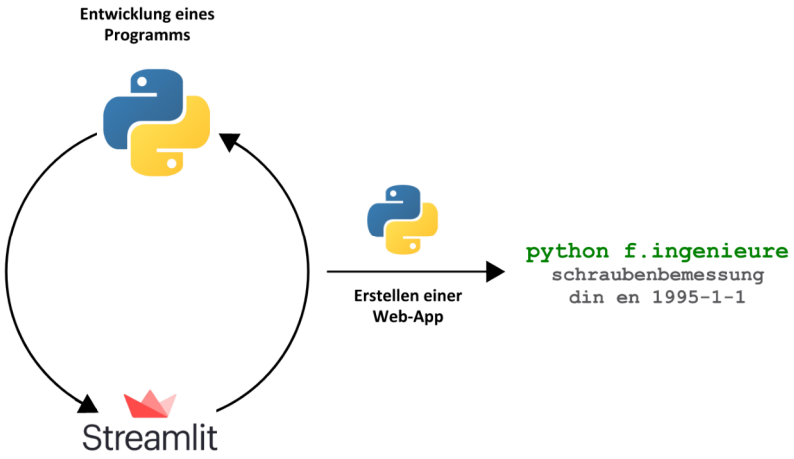


Abbildung 4: Verwendete Programme für Anwendungsbeispiele

Die Nutzung der herstellereigenen Programme und Tabellen ist vor allem bei häufiger Nutzung zeitintensiv und stellt eine Arbeitsweise dar, die verbesserungsfähig ist. Mit der Python-Bibliothek **Streamlit** soll daher eine Webapplikation für die Bemessung von Vollgewindeschrauben entwickelt werden.

Bei Streamlit handelt es sich um eine Bibliothek eines Drittanbieters, der sich mit interaktiven webbasierten Applikationen beschäftigt. Mit wenig Aufwand können benutzerfreundliche Oberflächen erstellt werden, die anschließend als Programm exportiert und öffentlich geteilt werden können. Streamlit soll daher genutzt werden, um ein Bemessungsprogramm zu schreiben, welche die Schwächen der existierenden Tools aufgreift und eine simple Möglichkeit darbietet Vollgewindeschrauben zu bemessen.

1.3.3 Erstellung von interaktiven Datenblättern mit Jupyter Notebook

Für die Aufbereitung des Codes in Jupyter Notebook wird das Modul **handcalcs** verwendet, welches das automatische Rendern bzw. Formatieren von mathematischen Formeln in **Latex** ermöglicht. Jupyter Notebook bietet zudem eine Funktion an, um die Code-Zeilen auszublenden, sodass übersichtliche statische Bemessungsvorlagen erstellt werden können, die eine vielseitig anwendbare und leistungsstarke Alternative zu herkömmlichen Programmen, wie Excel oder Word darstellen.

```
%%render 2
eta_y = (sigma_cd/f_c0d)**2 + k_m1*sigma_my2d_imp/f_myd + k_m2*sigma_mz2d/f_mzd
eta_z = (sigma_cd/f_c0d)**2 + k_m3*sigma_y2d/f_myd + k_m4*sigma_mz2d_imp/f_mzd
```

$$\eta_y = \left(\frac{\sigma_{cd}}{f_{c0d}}\right)^2 + k_{m1} \cdot \frac{\sigma_{my2d_{imp}}}{f_{myd}} + k_{m2} \cdot \frac{\sigma_{mz2d}}{f_{mzd}}$$
$$= \left(\frac{13000.00}{17230.77}\right)^2 + 1 \cdot \frac{5416.50}{17230.77} + 0.70 \cdot \frac{1912.20}{17230.77}$$
$$= 0.96$$

$$\eta_z = \left(\frac{\sigma_{cd}}{f_{c0d}}\right)^2 + k_{m3} \cdot \frac{\sigma_{y2d}}{f_{myd}} + k_{m4} \cdot \frac{\sigma_{mz2d_{imp}}}{f_{mzd}}$$
$$= \left(\frac{13000.00}{17230.77}\right)^2 + 0.70 \cdot \frac{4128.75}{17230.77} + 1 \cdot \frac{2907.00}{17230.77}$$
$$= 0.91$$

Abbildung 5: Ausschnitt aus einem Jupyter Notebook

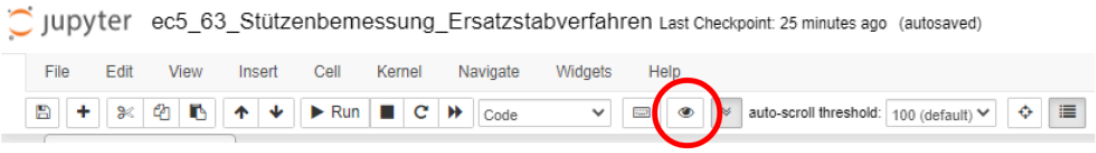


Abbildung 6: Funktion zur Ausblendung der Code-Zeilen

Link zur Online-Dokumentation:

https://github.com/connorferster/handcalcs

2. Programmierung in Python

2.1 Merkmale

Python ist eine moderne Programmiersprache, die 1991 von Guido van Rossum am niederländischen Forschungsinstitut für Mathematik und Informatik entwickelt wurde. Ziel für die Entwicklung der neuen Programmiersprache war ein Nachfolger der Lehrsprache ABC, welche nach dem Konzept der größtmöglichen Einfachheit und Übersichtlichkeit weiterentwickelt werden sollte. Dies umfasste eine Innovation durch mehrere Merkmale hinweg und macht Python heute zu eines der beliebtesten Programmiersprachen.

Besonders für Anfänger ist die simple *Syntax* in Python attraktiv und gestaltet das Programmieren deutlich intuitiver als andere Sprachen. Unter Syntax versteht man im Allgemeinen ein Regelsystem, welches den Aufbau und das Zusammenfügen der Programmierelemente vorgibt und ist das Äquivalent zur Grammatik in der Sprache. Python verzichtet fast gänzlich auf Zeichen oder Symbole wie Klammern und Semikolons und ist stattdessen durch Einrückungen strukturiert. Dies minimiert eines der häufigsten Fehlerquellen. Hinzu besteht die Python-Sprache aus Funktionen, Befehlen und Argumenten aus englischen Wörtern, die neben einer Übersichtlichkeit enorm zum besseren Verständnis beitragen (vgl. Bonacina 2018: 11).

Auch bei der *Typisierung* der Variablentypen ist Python weitaus innovativer. Oft ist es bei anderen Programmiersprachen erforderlich den Datentypen manuell zu definieren, welcher im weiteren Verlauf nicht mehr verändert werden kann. Python hingegen weist bei der Definition einer Variablen automatisch einen Datentypen zu, der dynamisch geändert wird.

Ein weiteres besonderes Merkmal von Python ist die große Auswahl an *Bibliotheken*. Neben der vorinstallierten Standardbibliothek, können je nach erforderlicher Problemstellung und Anwendungsbereich eine Vielzahl an kostenlosen Zusatzmodulen von Drittanbietern importiert werden. Sie befassen sich mit meist wissenschaftlich oder technisch orientierten Problemen und bieten vorgefertigte Funktionen für das beispielsweise Rechnen mit Matrizen oder das Visualisieren von Datensätzen an.

Sämtliche Module der Standardbibliothek oder von Drittanbietern sind online ausführlich dokumentiert. Die sogenannte *API*-Dokumentation (Application Programming Interface) ist eine Art Handbuch und erklärt die technische Funktionsweise eines Moduls. Sie beinhaltet einen Leitfaden zu sämtlichen Funktionalitäten und führt eine Vielzahl an Beispielen und Tutorials an. Im weiteren Verlauf der Arbeit wird bei jeder Einführung eines Moduls ein Link zur API-Dokumentation angegeben.

Python ist eine *quelloffene* und kostenfreie Programmiersprache und wird von einer großen Gemeinschaft aus Programmierern und Entwicklern im Internet gewartet, weiterentwickelt und angewendet. Zudem lassen sich unzählige von Nutzern erstellte Video-Seminare und Lerntutorien finden, die beim Anwenden stets eine Hilfe darstellen. Auch in der Literatur wird eine weite Bandbreite an Büchern angeboten, die von Einsteiger- bis hin zu fachspezifischen und fortgeschrittenen Python-Lehrbüchern reichen.

Grundsätzlich gibt es zwei verschiedene Versionen von Python (Python 2 und *Python 3*), die derzeit noch weiterentwickelt werden. Eines der Merkmale von Python ist die Rückwärtskompatibilität der Versionen. Ein geschriebenes Programm in Python funktioniert demnach stets auf einer neueren Version. Für die Erstellung der Arbeit wird Python 3 verwendet.

2.2 Vorbereitungsmaßnahmen

2.2.1 Entwicklungsumgebung: Anaconda

Es gibt verschiedene Wege, um unter Microsoft Windows Python zu installieren. Die manuelle Installation und Verwaltung von Python kann mitunter komplex ausfallen, sodass Python mit einer Distribution installiert werden sollte. Eine Distribution ist eine Softwarepaketverwaltung, die verschiedene Komponenten, wie Python-Interpreter und Installationsmanager für Bibliotheken vereint. Die gesamte Arbeit wird mit der Distribution **Anaconda** verfasst.

Anaconda bietet eine quelloffene und kostenlose Möglichkeit Python und die Programmiersprache *R* zu nutzen und installiert automatisch die wichtigsten Pakete und Entwicklungsumgebungen. Sie bieten mit unterschiedlichen Schwerpunkten

umfangreiche Funktionalitäten an, die das Programmieren vereinfachen und vor allem in den Bereichen der Datenverarbeitung, künstlicher Intelligenz und naturwissenschaftlichem Programmieren zum Einsatz kommen und sich auf die Verarbeitung von großen Datenmengen, Vorhersageanalyse und wissenschaftlichem Rechnen spezialisieren (vgl. Steinkamp 2021: 18).

2.2.1.1 Installation

Der folgende Link führt zum Download-Bereich der Anaconda Distribution.

Download-Bereich:
https://www.anaconda.com/products/distribution

Installationsanweisung:
https://docs.anaconda.com/anaconda/install/

Folgt man dem Link, wird bereits automatisch erkannt, nach welcher CPU-Architektur das Windows System konzipiert ist (32-Bit oder 64-Bit). Die meisten Computer heutzutage verfügen über genug Arbeitsspeicher, sodass eine Installation der 64-Bit Version erfolgen sollte. Nachdem die .msi-Datei heruntergeladen wurde kann die Installation von Anaconda ausgeführt werden.

Nach den Empfehlungen der Installationsanweisung, kann die Installation mit den Optionen der Voreinstellungen durchgeführt werden.

2.2.1.2 Startseite

Nach der erfolgreichen Installation kann die Applikation unter dem Namen „Anaconda Navigator (Anaconda3)“ gestartet werden. Auf der Startseite sind die verschiedenen Entwicklungsumgebungen dargestellt, die teils bereits vorinstalliert wurden.

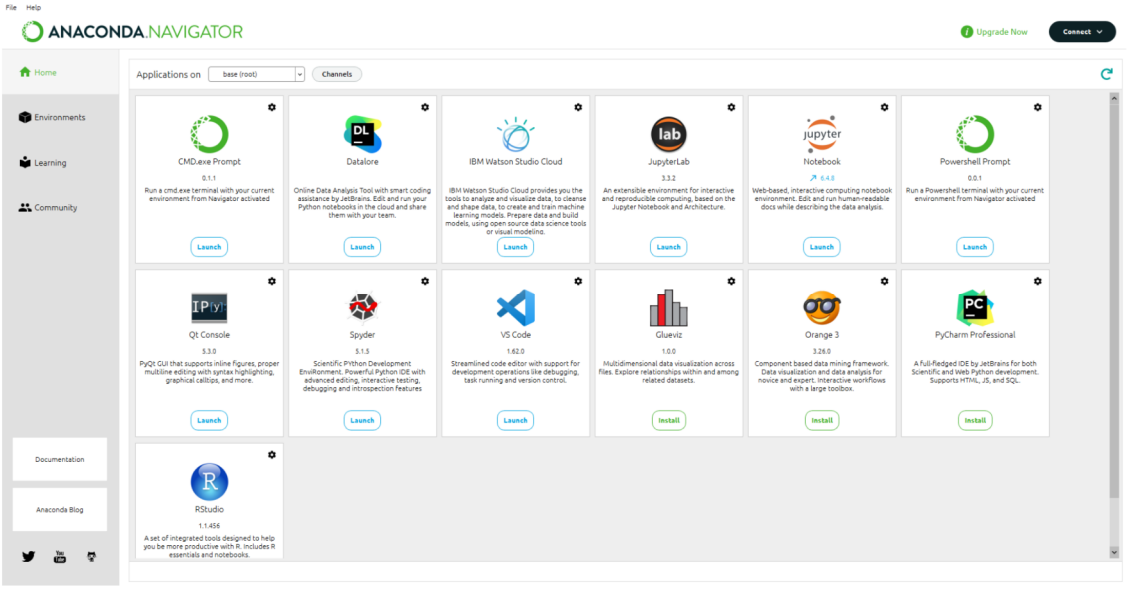


Abbildung 7: Startseite der Entwicklungsumgebung Anaconda

Im Folgenden wird nur auf die Komponenten eingegangen, die für die Bearbeitung und Ausführung dieser Arbeit relevant sind.

2.2.1.3 Komponenten von Anaconda

Python Paketmanager (pip)

Eines der Vorteile für das Nutzen eine Distribution ist der integrierte Paketmanager. Bei Anaconda wird standardmäßig der offizielle Python-Paketmanager „pip“ installiert, der aus dem zentralen Paketpool „Python Package Index (PyPI)“ Bibliotheken von Drittanbietern herunterlädt. Pip steht für „pip installs packages“ und übernimmt die Aufgabe für das Installieren, Aktualisieren und Entfernen von Bibliotheken (vgl. Steinkamp 2021: 19).

Python Anaconda Prompt

Eine weitere Komponente von Anaconda ist die auf der Startseite zu sehende Eingabeaufforderung “CMD.exe Prompt”. Hier lassen sich alle Applikation von der Konsole aus starten. Nach dem Öffnen ist einerseits in Klammern der Name der virtuellen Umgebung zu sehen (1) und andererseits der Pfad des aktuellen Arbeitsverzeichnisses (2). Der Name `base` wird automatisch von Anaconda vergeben. Das Einrichten unterschiedlicher Umgebungen ist zweckmäßig, wenn man verschiedene Versionen von Python oder Bibliotheken verwendet und ist hier daher

nicht von Bedeutung. Das Arbeitsverzeichnis stellt die aktuelle Position in der Verzeichnishierarchie dar.

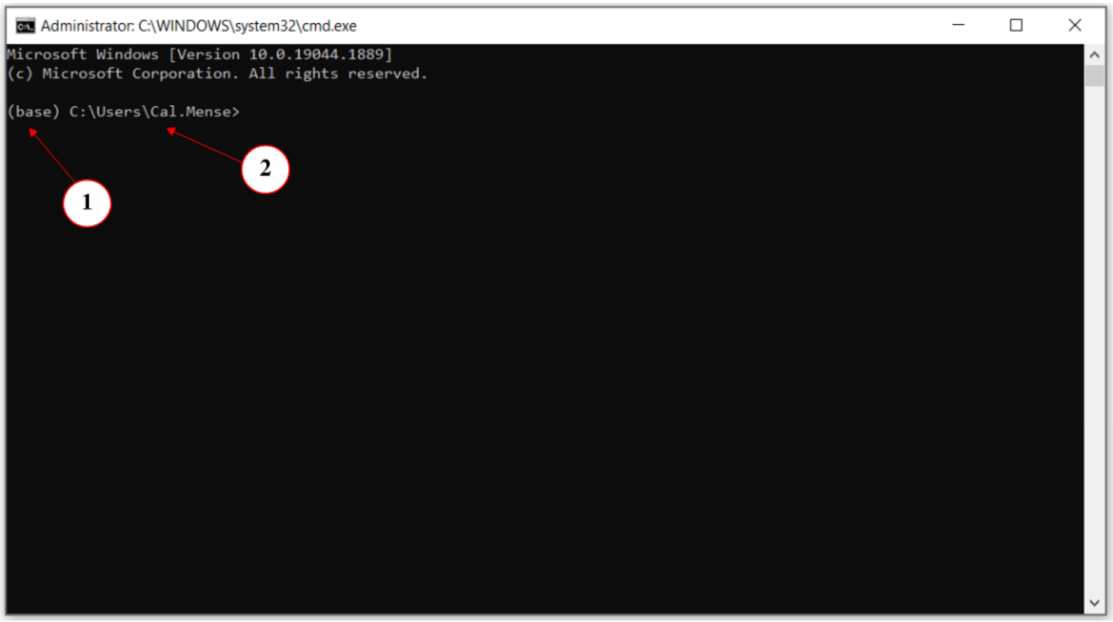


Abbildung 8: Eingabeaufforderung in Windows

Im Rahmen dieser Arbeit ist jedoch nur das Installieren der Bibliotheken wesentlich. Dies wird mit dem Paketmanager *pip* durchgeführt. Um alle benötigten Bibliotheken auf einmal zu installieren, gibt es die Möglichkeit mit folgendem Befehl eine Text-Datei einzulesen, die eine Liste der Bibliotheken mit zugehörigen Versionen enthält.

```
pip install -r requirements.txt
```

Hierzu müssen die folgenden Zeilen mit der Windows Applikation Notepad als Text-Datei *requirements.txt* gespeichert werden.

```
handcalcs==1.6.0

jupyter-nbextensions-configurator==0.5.0

pandoc==2.2

pdflatex==0.1.3

pyautocad==0.2.0

pywin32==302

streamlit==1.12.0
```

```
xlwings==0.26.2
```

Um den Befehl ausführen zu können, muss zunächst das Arbeitsverzeichnis gewechselt werden. Dies erfolgt mit dem Befehl

```
cd C:\Users\Pfad\zum\Ordner
```

Eine Animation für die Installation nach dieser Vorgehensweise ist im Jupyter-Arbeitsblatt *2.0_Programmierung in Python.ipynb* zu finden.

Entwicklungsumgebung: Jupyter Notebook

Jupyter Notebook ist eine Web-basierte Anwendung, die es ermöglicht Programmier-Codes übersichtlich mit erklärenden Textpassagen zu visualisieren. Aufgebaut ist die Oberfläche mit einer Sequenz aus Befehlszeilen, die entweder Text oder Code enthalten können. Die sogenannte *MarkDown* Funktion für Textelemente können vielseitig formatiert werden. Durch das Zusammenspiel von Code- und Textzellen können ganze Skripte so aufbereitet werden, dass sie ideal für das wissenschaftliche Arbeiten verwendet werden können (vgl. The Jupyter Notebook 2022).

Link zur API-Dokumentation:

https://jupyter-notebook.readthedocs.io/en/stable/index.html

Entwicklungsumgebung: Spyder

Spyder verfolgt das Ziel einer plattformübergreifenden Entwicklungsumgebung für das wissenschaftliche Programmieren. Es ist kompatibel mit einer Reihe an Paketen, wie NumPy, SciPy, Matplotlib oder Pandas. Weitere Merkmale, die das Programmieren erleichtern sind Funktionen für schrittweises Debuggen, Untersuchen von Variablen oder Syntax-Hervorhebung mit Code-Vervollständigung (vgl. Spyder 5 Documentation 2022).

Link zur Online-Dokumentation:

https://docs.spyder-ide.org/current/index.html

Bibliothek: xlwings

Nach der erfolgreichen Installation aller Bibliotheken muss als Nächstes die Excel-Erweiterung (Add-In) der Bibliothek **xlwings** eingerichtet werden. Das Modul xlwings ermöglicht die Interaktion Pythons mit Excel und wird in Kapitel 3.5 behandelt. Das Add-In ermöglicht das Abrufen von Python-Code innerhalb der Excel-Datei und wird mit folgendem Befehl im Anaconda Prompt installiert.

Code 1: Installation der xlwings-Erweiterung für Excel (Add-in)

```
xlwings addin install
```

Nachdem Excel nun neu gestartet wird, sollte in der Registerleiste ein Reiter xlwings zu sehen sein.

Anmerkung: Um die Excel-Datei des Stützenbemessungsprogramms nutzen zu können, müssen die folgenden vier Felder richtig ausgefüllt werden.

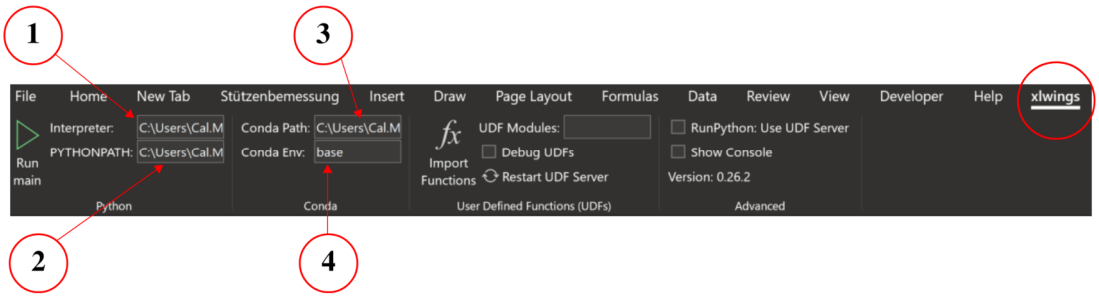


Abbildung 9: xlwings Register in Excel

1. Interpreter: C:\Users\Cal.Mense\Anaconda3\pythonw.exe
2. PYTHONPATH: Pfad zum Python-Code-Ordner (01_Funktionen)
3. Conda Path: C:\Users\Cal.Mense\Anaconda3
4. Conda Env: base

Die folgenden Schritte sollten bereits richtig eingerichtet sein, werden dennoch kurz erläutert.
Das Ausführen des Python-Codes erfolgt über die Microsoft Programmiersprache VBA. Hierzu kann mit der Tastenkombination `alt+F11` das Menüfenster für VBA gestartet werden. Unter `Extras (Tools)` → `Referenzen (References)` muss ein Haken bei xlwings gesetzt sein. Mit dem `RunPython`-Befehl von VBA können einzelne Funktionen von einer Python-Datei (.py) exekutiert werden.

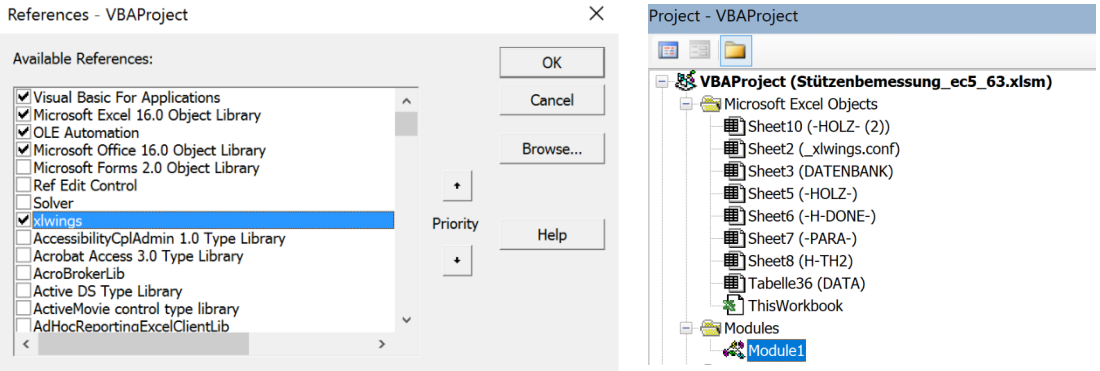


Abbildung 10: VBA Einstellungen für die xlwings Erweiterung

Code 2: Aufruf eines Python-Codes in VBA

```
Sub esv_iteration()
    ActiveSheet.Select
    RunPython "import Stützenbemessung_ec5_63;
               Stützenbemessung_ec5_63.ec5_esv_iteration()"
End Sub
```

Link zur Online-Dokumentation:

https://docs.xlwings.org/en/stable/addin.html

3. Anwendungsbeispiel 1: Stützenbemessung

3.0 Einführung

In den nachfolgenden Kapiteln wird aus der Verknüpfung mit Excel und Python ein leistungsstarkes Stützenbemessungsprogramm erstellt, welches einen typischen Arbeitsablauf vom *Lastabtrag*, der *Dimensionierung* der Stützen und der nachfolgenden *Dokumentation* vereint. Dabei wird besonders großen Wert draufgelegt, dass die übersichtliche Oberfläche von Excel genutzt wird, um den Nutzer durch Eingabeparameter einzubinden. Zusätzlich gewährt die Ausgabe von Zwischenergebnissen die Kontrolle während des gesamten Vorgangs.

Bemessen werden die Stützen nach den beiden Berechnungsverfahren des Eurocodes 5 (Theorie I. und II. Ordnung). Diese werden im Folgenden anhand einer programmierten Beispielrechnung erläutert und anschließend sukzessive zu einem komplexen Programm mit diversen Funktionalitäten aus unterschiedlichen Python-Bibliotheken aufgebaut und detailliert aufgeschlüsselt.

Es folgt eine kurze Einführung der Funktionalitäten des Stützenbemessungsprogramms. Eine Anleitung zur genaueren Anwendung befindet sich im Ordner *01_Stützenbemessung_ec5_63* (*Stützenbemessung_ec5_63_Handbuch.ipynb*).

Lastabtrag

In dem Reiter „Lastabtrag“ steht eine Vorlage zur Verfügung, wo die Ergebnisse eines Lastabtrags geschossweise eingetragen werden. Die Information beinhalten Stützenpositionsnummern, Abmessungen, Einwirkungen, sowie x- und y-Koordinaten der jeweiligen Stütze.

python f.ingenieure
stützenbemessung
din en 1995-1-1

Ergebnisse des Lastabtrags | **Stützenpositionen** | **Decken**

Grundparameter				Geometrie			Einwirkung			Koordinaten			Koordinaten	
Pos.	Geschoss	Typ	Material	L	b	h	Ned	Myd	Mzd	x	y	Winkel	x	y
-	-	-	-	[m]	[m]	[m]	[kN]	[kNm]	[kNm]	[m]	[m]	[gr]	[m]	[m]
S-201	EG	Pendelstütze	GL 24c	6	0,5	0,5	4082	0	0	62	39		71	27
S-202	EG	Pendelstütze	GL 24c	8	0,5	0,5	4816	0	0	47	31		35	8
S-203	EG	Pendelstütze	GL 24c	8	0,5	0,5	2911	0	0	42	29		12	38
S-204	EG	Pendelstütze	GL 24c	8	0,5	0,5	2883	0	0	37	29		12	69
S-205	EG	Pendelstütze	GL 24c	8	0,5	0,5	2930	0	0	33	34		60	69
S-206	EG	Pendelstütze	GL 24c	4	0,5	0,5	3382	86	0	27	47		62	54
S-207	EG	Pendelstütze	GL 24c	4	0,5	0,5	3431	86	0	27	53		28	54
S-208	EG	Pendelstütze	GL 24c	4	0,5	0,5	3656	113	0	29	54		28	43
S-209	EG	Pendelstütze	GL 24c	4	0,5	0,5	3858	113	0	34	54		39	28
S-210	EG	Pendelstütze	GL 24c	4	0,5	0,5	3758	113	0	50	54		66	42
S-211	EG	Pendelstütze	GL 24c	4	0,5	0,5	3900	86	0	56	54		73	32
S-212	EG	Pendelstütze	GL 24c	4	0,5	0,5	2447	86	86	56	62		69	30
S-213	EG	Pendelstütze	GL 24c	8	0,5	0,5	3843	0	0	50	62		71	27
S-214	EG	Pendelstütze	GL 24c	8	0,5	0,5	2881	0	0	45	62			
S-215	EG	Pendelstütze	GL 24c	8	0,5	0,5	3085	0	0	40	62			
S-216	EG	Pendelstütze	GL 24c	8	0,5	0,5	2977	0	0	34	62			

Abbildung 11: Reiter „Lastabtrag“ des Stützenbemessungsprogramms

Stützenbemessung

Die Werte des Lastabtrags können im Bereich der **Eingabeleiste** über einen Schalter importiert werden. Zusätzlich ist es möglich, über die Eingabe von Rundungswerten, die Einwirkungen aufzurunden **(1).** Die Parameter der einzelnen Stützenpositionen können ebenso individuell angepasst werden **(2).**

python f.ingenieure
stützenbemessung
din en 1995-1-1

Stützenparameter

Stützenbemessung nach DIN EN 1995-1-1

Eingabeleiste

Import aus Lastabtrag

Geschoss: 1.OG

Importieren: Importieren

Gruppierung der Lasten

N_{ed}: 200 [kN]

M_{ed}: 50 [kNm]

Aufrunden: Runden

Theorie II. Ordnung

Vorkrümmung e0: 0,0025 [-]

System					Belastung							Th.I	Th.II
Pos.	Geschoss	Länge	Breite	Höhe	N_{ed}	M_{yd}	M_{zd}	Güte	$f_{c,0,k}$	$f_{m,k}$	$E_{0,mean}$	$E_{0,05}$
-	-	[m]	[m]	[m]	[kN]	[kNm]	[kNm]	-	[N/mm²]	[N/mm²]	[N/mm²]	[N/mm²]
S-201	1.OG	3,00	0,50	0,40	2000	50	30	GL 28h	28	28	12.500	10.500
S-202	1.OG	8,30	0,70	0,70	4800	0	0	GL 28c	28	28	12.500	10.400
S-203	1.OG	8,30	0,60	0,60	3000	0	0	GL 24c	24	24	11.000	9.100
S-204	1.OG	8,30	0,60	0,60	3000	0	0	GL 24c	24	24	11.000	9.100
S-205	1.OG	8,30	0,60	0,60	3000	0	0	GL 24c	24	24	11.000	9.100
S-206	1.OG	3,60	0,60	0,60	3000	100	0	GL 28c	28	28	12.500	10.400
S-207	1.OG	3,60	0,60	0,60	3000	100	0	GL 28c	28	28	12.500	10.400
S-208	1.OG	3,60	0,60	0,60	3200	150	0	GL 32h	32	32	13.700	11.100
S-209	1.OG	3,60	0,65	0,65	3400	150	0	GL 28c	28	28	12.500	10.400
S-210	1.OG	3,60	0,65	0,65	3400	150	0	GL 28c	28	28	12.500	10.400
S-211	1.OG	3,60	0,60	0,60	3600	100	0	GL 32h	32	32	13.700	11.100
S-212	1.OG	3,60	0,60	0,60	2400	100	100	GL 28c	28	28	12.500	10.400
S-213	1.OG	7,80	0,60	0,60	4000	0	0	GL 32h	32	32	13.700	11.100
S-214	1.OG	7,80	0,55	0,55	3000	0	0	GL 28c	28	28	12.500	10.400
S-215	1.OG	7,80	0,60	0,60	3200	0	0	GL 24c	24	24	11.000	9.100
S-216	1.OG	7,80	0,55	0,55	3000	0	0	GL 28c	28	28	12.500	10.400

Abbildung 12: Eingabeparameter des Stützenbemessungsprogramms

Die Bemessung nach dem Ersatzstabverfahren und Theorie II. Ordnung werden über die grünen Schalter ausgeführt **(1)**. Das Programm bemisst 60 Stützen in etwa 10 Sekunden. Sämtliche Zwischen- und Endergebnisse können zur Kontrolle

nebeneinander überblickt werden **(2 - 3).** Gemäß einer angegebenen Zielausnutzung in der Eingabeleiste werden die maßgebenden Bemessungsergebnisse farblich markiert **(4)**.

Biegeknicken von Druckstäben
Biegespannung nach **Theorie I. Ordnung**
nach DIN EN 1995-1-1 Abs. 6.3
Ersatzstabverfahren
Rechnen

ESV								
λ	λ_{rel}	k_c	k_{crit}	Versagen	σ_{cd}	σ_{myd}	σ_{mzd}	$\eta_{_Th.1}$
-	-	-	-	-	[kN/m²]	[kN/m²]	[kN/m²]	[%]
26	0,43	0,98	1,00	Kn	10.000	3.750	1.800	88%
41	0,68	0,94	1,00	Kn	9.796	0	0	81%
48	0,78	0,90	1,00	Kn	8.333	0	0	84%
48	0,78	0,90	1,00	Kn	8.333	0	0	84%
48	0,78	0,90	1,00	Kn	8.333	0	0	84%
21	0,34	1,00	1,00	Kn	8.333	2.778	0	86%
21	0,34	1,00	1,00	Kn	8.333	2.778	0	86%
21	0,36	0,99	1,00	Kn	11.667	0	0	80%
34	0,58	0,96	1,00	Kn	12.813	0	0	90%
21	0,36	0,99	1,00	Kn	12.778	0	0	87%
18	0,29	1,00	1,00	Sp	11.429	0	0	78%
18	0,30	1,00	1,00	N/A	11.837	0	0	0%
18	0,29	1,00	1,00	Sp	11.429	0	0	78%
18	0,29	1,00	1,00	Sp	11.020	0	0	73%
21	0,34	1,00	1,00	Kn	11.111	0	0	86%
23	0,37	0,99	1,00	Kn	11.240	0	0	88%
45	0,77	0,91	1,00	Kn	11.111	0	0	83%
21	0,36	0,99	1,00	Kn	9.444	0	2.778	83%
19	0,33	1,00	1,00	Kn	8.994	0	3.277	83%

Biegeknicken von Druckstäben
Biegespannung nach **Theorie II. Ordnung**
nach DIN EN 1995-1-1 Abs. 6.3
Rechnen

Theorie II								
e_{I_y}	M_{I_y}	e_{II_y}	M_{II_y}	e_{I_z}	M_{I_z}	e_{II_z}	M_{II_z}	$\eta_{_Th.2}$
[mm]	[kNm]	[mm]	[kNm]	[mm]	[kNm]	[mm]	[kNm]	[%]
8	65	10	70	0	30	1	31	72%
21	100	25	121	0	0	0	0	74%
21	62	27	81	0	0	0	0	77%
21	62	27	81	0	0	0	0	77%
21	62	27	81	0	0	0	0	77%
0	100	1	104	9	27	9	28	70%
0	100	1	104	9	27	9	28	70%
9	38	9	40	0	0	0	0	70%
20	160	103	840	0	0	0	0	95%
9	41	10	44	0	0	0	0	83%
9	50	9	52	0	0	0	0	85%
9	52	9	54	0	0	0	0	71%
9	50	9	52	0	0	0	0	85%
9	49	9	51	0	0	0	0	80%
9	36	10	38	0	0	0	0	82%
9	31	10	33	0	0	0	0	85%
20	78	25	100	0	0	0	0	75%
9	31	9	32	0	100	1	104	66%
9	34	9	35	0	150	1	155	65%

Abbildung 13: Gegenüberstellung der Bemessungsverfahren

Über die Eingabe von Optimierungsparametern hinsichtlich der Festigkeit und den Querschnittsabmessungen wird eine Optimierungsfunktion angeboten, die innerhalb der Grenzen iterativ die Parameter anpasst, bis die gewünschte Ausnutzung erreicht ist.

Optimierungsparameter

Parameter 1: Festigkeit		
oben	GL 28h	
unten	GL 24h	
Parameter 2: Querschnitt		
oben	0,3	[m]
unten	0,8	[m]
steps	0,05	[m]
Optimierung der Ausnutzung		
oben	0,9	
unten	0,8	
Optimieren	Rechnen	

Abbildung 14: Eingabefelder von Optimierungsparameter

Die nachfolgende Abbildung zeigt einen Ausschnitt der Ergebnisse einer Optimierung. Der Algorithmus passt zunächst die Festigkeit des Holzes **(1)** und anschließend die Stützenabmessungen an **(2).** Dies erfolgt, bis die gewünschte Ausnutzung erreicht wird **(3).** Ist dies nicht der Fall, kann ein zweiter Iterationsdurchgang durchgeführt werden.

Optimierung

Biegespannung nach **Theorie I. und II. Ordnung**

nach DIN EN 1995-1-1 Abs. 6.3

Rechnen

Festigkeit - Iteration i (1)					Querschnitt - Iteration i (2)					Ausnutzung - Iteration i (3)					Optimierung
i=1	i=2	i=3	i=4	i=5	i=1	i=2	i=3	i=4	i=5	i=1	i=2	i=3	i=4	i=5	abgeschlossen
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	62/62
GL 28h					0,50					88%					NW erfüllt
GL 32h	GL 32h	GL 32h	GL 28c		0,50	0,60	0,70	0,70		251%	108%	71%	81%		NW erfüllt
GL 32h	GL 32h	GL 28c	GL 28h	GL 24c	0,50	0,60	0,60	0,60	0,60	104%	63%	72%	72%	84%	NW erfüllt
GL 32h	GL 32h	GL 28c	GL 28h	GL 24c	0,50	0,60	0,60	0,60	0,60	104%	63%	72%	72%	84%	NW erfüllt
GL 32h	GL 32h	GL 28c	GL 28h	GL 24c	0,50	0,60	0,60	0,60	0,60	104%	63%	72%	72%	84%	NW erfüllt
GL 32h	GL 32h	GL 28c			0,50	0,60	0,60			115%	76%	86%			NW erfüllt
GL 32h	GL 32h	GL 28c			0,50	0,60	0,60			115%	76%	86%			NW erfüllt
GL 32h	GL 32h				0,50	0,60				138%	89%				NW erfüllt
GL 32h	GL 32h	GL 32h	GL 32h	GL 28c	0,50	0,60	0,70	0,75	0,75	402%	171%	91%	79%	89%	NW erfüllt
GL 32h	GL 32h				0,50	0,60				143%	80%				NW erfüllt
GL 32h	GL 32h	GL 32h	GL 32h		0,50	0,60	0,70	0,80		766%	293%	154%	90%		Grenze
GL 32h	GL 32h				0,50	0,60				170%	87%				NW erfüllt
GL 32h	GL 32h	GL 32h	GL 28c		0,50	0,60	0,70	0,70		249%	121%	66%	85%		NW erfüllt
GL 32h	GL 32h	GL 32h	GL 28c	GL 32h	0,50	0,60	0,70	0,70	0,70	267%	130%	71%	91%	71%	NW n. erbracht
GL 32h	GL 32h	GL 32h	GL 28c		0,50	0,60	0,70	0,70		249%	121%	66%	85%		NW erfüllt
GL 32h	GL 32h	GL 32h	GL 28c	GL 28h	0,50	0,60	0,70	0,70	0,70	232%	113%	62%	80%	80%	NW n. erbracht
GL 32h	GL 32h	GL 28c			0,50	0,60	0,60			130%	76%	86%			NW erfüllt
GL 32h	GL 32h				0,50	0,60				131%	83%				NW erfüllt
GL 32h	GL 32h	GL 32h			0,50	0,60	0,65			172%	100%	83%			NW erfüllt

(4)

Abbildung 15: Optimierungsalgorithmus des Stützenbemessungsprogramms

Dokumentation

Nachdem die Stützenbemessung erfolgt und abgeschlossen ist, wird für den Zweck der Dokumentation das Exportieren von schriftlichen Einzelnachweisen als PDF und grafischen Übersichtsdarstellungen in AutoCAD ermöglicht.

Die nachfolgenden Abbildungen zeigen ein Beispiel eines AutoCAD-Plots. Die Stützen und Decken werden gemäß den Koordinaten und den Bemessungsergebnissen gezeichnet. Zusätzlich werden Eckdaten, wie Positionsnummer, Geschoss, Festigkeit, Abmessungen und Einwirkung als Text beigefügt. Zur weiteren Differenzierung werden die Stützen nach der Festigkeit und den Abmessungen farblich gruppiert.

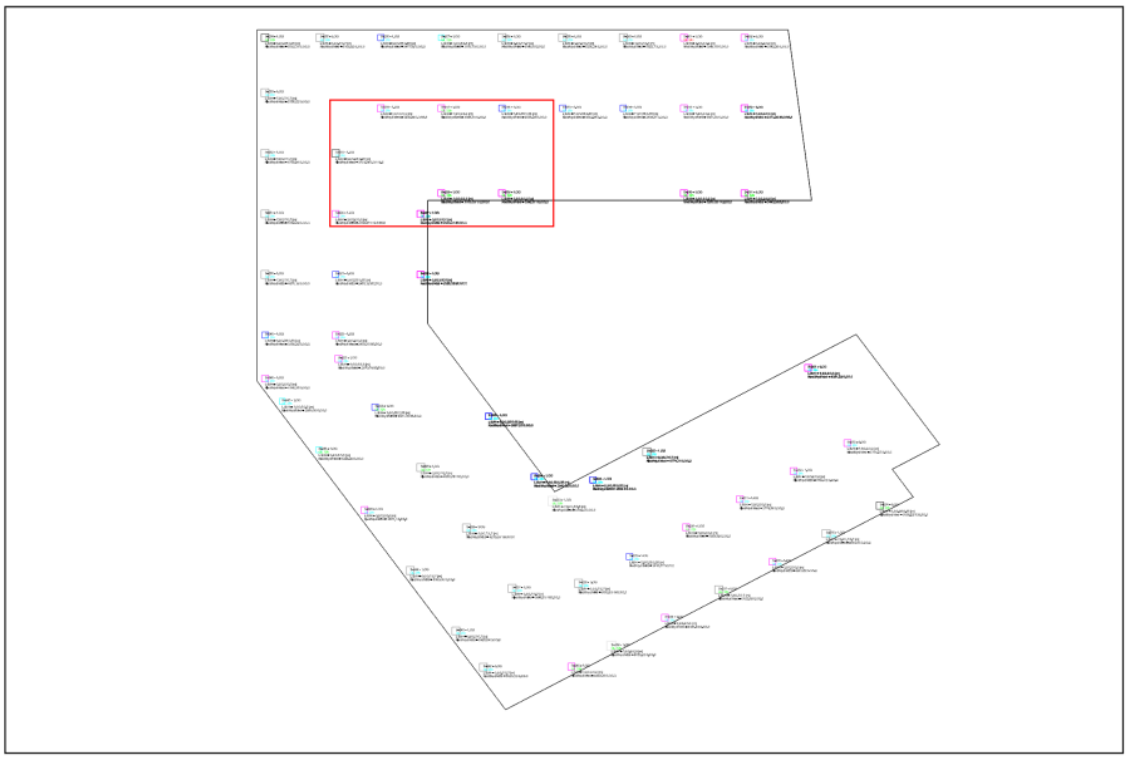


Abbildung 16: PDF-Plot der grafischen Übersicht in AutoCAD

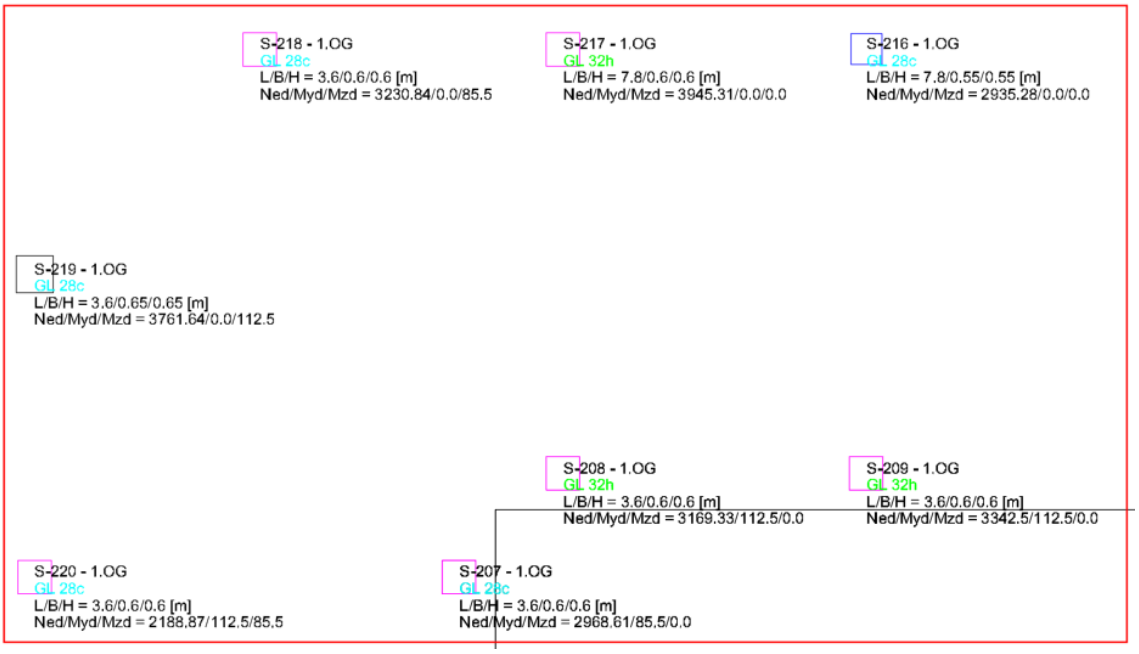


Abbildung 17: Ausschnitt des AutoCAD-Plots

3.1 Grundlagen zur Stützenbemessung im Holzbau

3.1.1 Holzbau

Im Ingenieurholzbau wird aufgrund ihrer unterschiedlichen Zusammensetzung und Eigenschaften im Wesentlichen zwischen Nadel- und Laubhölzern unterschieden.

Nadelhölzer, wie Fichte, Tanne, Lärche und Kiefer weisen eine homogenere und einfachere Zellstruktur auf, bei dem die Fasern parallel entlang des Stammes verlaufen. In Deutschland gehört die Fichte aufgrund des hohen Vorkommens und der schnell nachwachsenden Eigenschaft zu den am häufigsten verwendeten Holzarten. Die durch die simple Struktur des Holzes resultierende geringe Rohdichte von Nadelhölzern führt zu einer sehr guten Bearbeitbarkeit und einer gleichzeitig hohen Belastbarkeit. Laubhölzer hingegen besitzen einen komplexeren Zellaufbau und weisen höhere mechanische Festigkeiten auf.

Anders als bei isotropen Baustoffen wie Beton oder Stahl liegt die Besonderheit des Holzbaus bei der Berücksichtigung der unterschiedlichen Eigenschaften des Holzes in Abhängigkeit der Faserrichtung. Diese inhomogenen Holzeigenschaften spiegeln sich in den mechanischen Festigkeitswerten wider. Die nachfolgende Abbildung lässt erkennen, dass die Holzeigenschaften anhand von drei Hauptachsen definiert werden können:

- x-Achse: quer zur Faserrichtung und parallel zum Jahrring
- y-Achse: quer zur Faserrichtung und rechtwinklig zum Jahrring
- z-Achse: parallel zur Faserrichtung (vgl. Becker/Rautenstrauch 2012: 4)

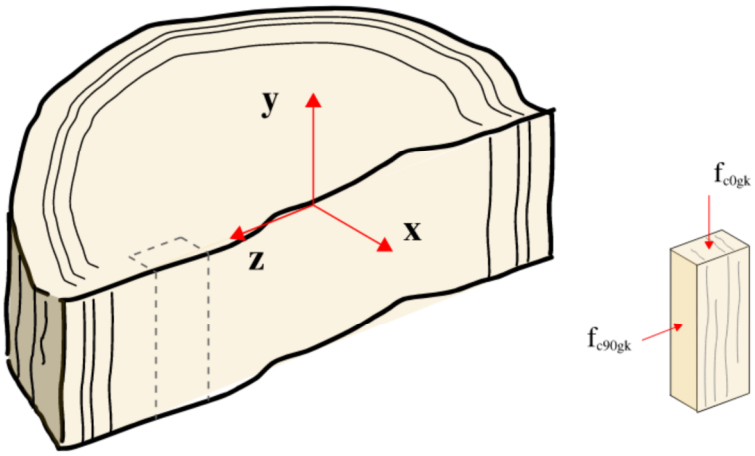


Abbildung 18: Richtungsabhängigkeiten der Holzfestigkeiten

Tabelle 1: Druckfestigkeiten nach der DIN EN 14080

Druckfestigkeiten			
	$f_{c,0,k}$	$f_{c,90,k}$	ρ_k
	[N/mm²]	[N/mm²]	[kg/m³]
GL24h	24	2,5	385
C25/30	25	25	2300

Diese Unterschiede der Holzeigenschaften sollen kurz anhand der Abbildung und der Druckfestigkeiten nach der DIN EN 14080 veranschaulicht werden. Hier ist ersichtlich, dass die Druckbeanspruchbarkeit des Holzes bei einem Brettschichtholz der Güte GL24h mit 24 N/mm^2 in Faserrichtung (y) etwa 10-mal höher ist als die Beanspruchbarkeit quer zur Faserrichtung (z) mit 2.5 N/mm^2. Parallel zur Faserrichtung ist demnach die Druckfestigkeit vergleichbar mit der Eigenschaft eines C25/30 Betons und ist dabei etwa 6-mal leichter.

Die hohen Festigkeitseigenschaften parallel zur Faser kommen vor allem bei Stützen zum Einsatz.

3.1.2 Stützenbemessung

Die Bemessung eines Tragwerks in der Baustatik verfolgt das Ziel, Verformungen möglichst kleinzuhalten. Ist dies der Fall, können die Verformungen vernachlässigt werden, sodass die Schnittgrößenermittlung am unverformten Gleichgewichtssystem erfolgen kann. Diese Betrachtungsweise der geometrischen Linearität nach der

Theorie I. Ordnung trifft auf die meisten Bauteile zu. Ebenso wird von einer physikalischen Linearität ausgegangen, bei der ein linearer Zusammenhang zwischen einwirkender Last und Verformung besteht. Das Materialverhalten wird hierbei als elastisch bezeichnet, da die Verformungen nach einer Belastung reversibel sind und das Bauteil sich in den Ursprung zurück verformt (vgl. Neuhaus 2017: 820). Die Steifigkeit gegenüber dieser Verformung wird mit dem Elastizitätsmodul beschrieben:

$$E = \frac{\sigma}{\epsilon} \quad (N/mm^2)$$

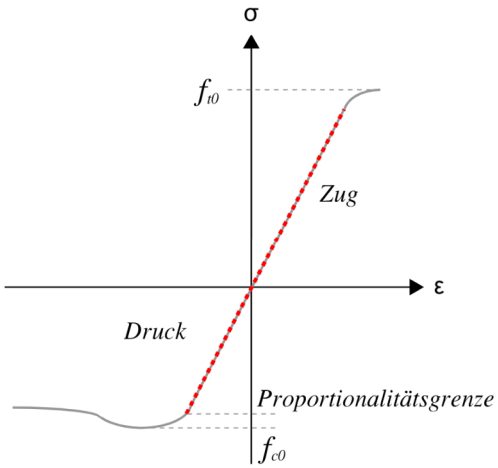


Abbildung 19: Spannungs-Dehnungs-Linie von Holz

Die Gültigkeit der physikalischen Linearität wird in der Spannungs-Dehnungs-Linie in Abbildung 13 deutlich. Dargestellt ist das Ergebnis eines Kurzzeitversuches, bei dem der Zusammenhang zwischen der Dehnung bei aufgebrachter Beanspruchung parallel zur Faserrichtung ermittelt wird. Die rot gestrichelte Linie markiert den linear-elastischen Bereich des Holzes, bei der eine umkehrbare Abhängigkeit zwischen Spannung und Verformung besteht verformt. Bei höheren Beanspruchungen oberhalb der Proportionalitätsgrenze kommt es zu plastischen, nicht reversiblen Verformungen, die stärker als die Spannungen zunehmen (vgl. Neuhaus 2017: 28).

Sind Verformungen durch beispielsweise baupraktische Imperfektionen für die Standsicherheit eines Bauteils maßgebend, muss die Berechnung am verformten System erfolgen. Hierbei ist die Rede von geometrisch nicht-linearem Tragverhalten oder Schnittgrößenermittlung nach Theorie II. Ordnung. Zur Vereinfachung kann das Materialverhalten dennoch anhand der Theorie I. Ordnung beschrieben werden. Plastische und irreversible Verformungen (Theorie II. Ordnung) treten nicht auf.

Tabelle 2: Theorie I. und II. Ordnung bei der Stützenbemessung

Stützenbemessung nach Eurocode 5		
	Theorie I. Ordnung	Theorie II. Ordnung
Schnittgrößen	*Linear* • Ermittlung am unverformten System • Verformungen werden vernachlässigt	*Nichtlinear* • Ermittlung am verformten System • Kleine Verformungen
Materialverhalten	*Linear* • Elastisch • Verformungen reversibel	*Linear* • Elastisch • Verformungen reversibel

Bei Stützen handelt es sich um druckbeanspruchte Stäbe, bei denen aufgrund der schlanken Geometrie ein Stabilitätsversagen noch vor dem Erreichen der Festigkeit maßgebend werden kann. Die dabei auftretenden seitlichen Auslenkungen verursachen weitere Biegemomente, die die Stütze weiter beanspruchen, sodass bei Stützen ein Nachweis der Tragfähigkeit mit Theorie II. Ordnung notwendig ist. In der DIN EN 1995-1-1 im Abschnitt 6.3 werden zwei Nachweisverfahren angeboten, die die Stabilität und die Standsicherheit gegenüber seitlichem Ausweichen regelt.

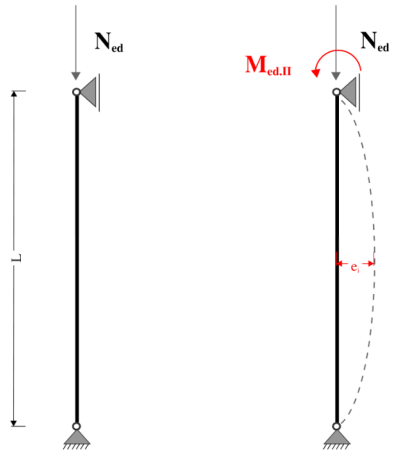


Abbildung 20: Geometrische Linearität und Nicht-Linearität

Das **Ersatzstabverfahren nach der Theorie I. Ordnung** ist ein vereinfachter Nachweis, bei dem die Stabilitätsuntersuchung als Näherung vorgenommen wird. Für das Tragsystem wird anhand einer Ersatzstablänge nach Euler *(s. Kapitel 3.2)* ein Knickbeiwert k_c ermittelt, der die geometrische Nicht-Linearität berücksichtigt, indem die Druckbeanspruchbarkeit abgemindert wird. Ein zusätzliches Ansetzen einer Imperfektion ist beim Ersatzstabverfahren also nicht notwendig. Dieses Verfahren

liefert hinreichend genaue Ergebnisse und hat sich baupraktisch durch den geringen Rechenaufwand bewährt.

Als Alternative ist nach dem Eurocode 5 ein **Spannungsnachweis nach Theorie II. Ordnung** zugelassen. Im Allgemeinen ist dieses Verfahren realitätsnäher und liefert überwiegend wirtschaftlich günstigere Ergebnisse. Besonders bei stabilitätsgefährdeten Stützen durch Imperfektionen oder hohen Momenten Beanspruchungen sollte die Schnittgrößenermittlung am verformten System durchgeführt werden. Die durch die Exzentrizitäten ausgelöste Biegebeanspruchung, führt zu einem überproportionalen Ansteigen des Momentes, welches nur durch eine iterative Berechnung und einem erhöhten Rechenaufwand ermittelt werden kann (vgl. Neuhaus 2017: 819).

Imperfektionen sind ungewollte und unvermeidbare Verformungen und werden in geometrische und strukturelle Imperfektionen unterschieden. Im Folgenden werden nur unverschiebliche Stützen betrachtet, die beispielsweise durch horizontale Deckenscheiben gehalten sind. Hier ist eine Vorkrümmung in der Stabachse anzusetzen. Weitere geometrische Imperfektionen, wie ungewollte Schrägstellungen sind für Rahmentragwerke relevant und werden hier nicht weiter betrachtet.

3.2 Stabilitätsnachweis nach dem Ersatzstabverfahren

3.2.1 Allgemeines

Das Ersatzstabverfahren ist ein vereinfachtes Nachweisverfahren, welches die Schnittgrößen nach der Theorie I. Ordnung, also am unverformten System, ermittelt und dabei die geometrische Nicht-Linearität für die Versagensfälle Knicken und Kippen anhand von Beiwerten berücksichtigt. Üblicherweise kommt dieses Verfahren bei der Betrachtung von Einzelstäben zum Einsatz. Im Folgenden wird der Nachweis anhand eines Rechenbeispiels weiter erläutert und dabei in Python programmiert.

Grundsätzlich wird das Skript so aufgebaut, dass sämtliche Eingabewerte am Anfang definiert werden, sodass bei der weiteren Nutzung der Nachweis schneller angepasst werden kann.

3.2.2 Eingangswerte

3.2.2.1 Bibliotheken

Generell sollen sämtliche Module am Anfang des Codes importiert und dabei wie folgt priorisiert werden:

1) Module aus Standardbibliothek
2) sonstige Module
3) eigene Module

Die einzelnen Gruppen werden alphabetisch geordnet.

Code 3: Importieren von Bibliotheken

```
from math import sqrt, pi, cos, sin     # Mathematische Funktionen
import handcalcs.render     # formattierte Gleichungen
```

Kommentare werden nach vier Leereichen mit einem Rautezeichen und einem weiteren Leerzeichen geschrieben.

3.2.2.2 System

Betrachtet wird eine Pendelstütze, die durch eine zentrische Druckkraft und doppelter Biegung beansprucht wird.

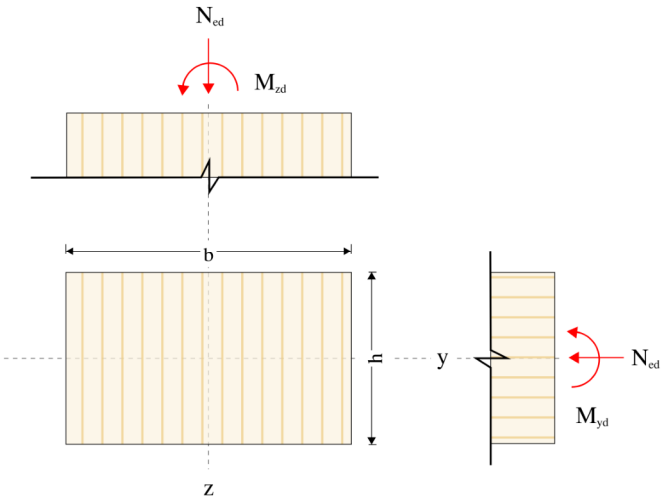


Abbildung 21: Stützenquerschnitt des Rechenbeispiels

3.2.2.3 Geometrie

Variablen

Eines der grundlegenden Konzepte von Programmiersprachen und Computeranwendungen ist die Fähigkeit Informationen zu speichern. Damit beispielsweise eine Anwendung mathematische Berechnungen ausführen kann, ist es notwendig, dass es bestimmte Werte als Variablen speichert, um mit diesem Wert weitere Berechnungen durchzuführen. Dies tut es, indem die elektronischen Informationen (auch Bits) auf dem Arbeitsspeicher abgelegt werden. Sobald diese Informationen benötigt werden, können sie mit dem vorgegebenen Namen abgerufen werden.

Code 4: Geometrie

```
L = 3     # m - Länge
b = 0.5     # m - Breite
h = 0.4     # m - Höhe
Lagerung = 'Pendelstütze'
```

Eine Variable besteht aus dem Namen der Variable und der zugehörigen Information. Da es bei komplexeren Berechnungen schnell zu einer Vielzahl an Variablen kommt, ist darauf zu achten, dass die Wahl des Namens verständlich bleibt und die Eindeutigkeit einer Variablen bewahrt wird.

Datentypen

Die Information, die der Variable zugeordnet wird, gehört ein Datentyp an, der in Python automatisch erkannt wird. Dies vereinfacht das Programmieren, da auf die manuelle Festlegung des Datentyps verzichtet werden kann. Der Datentyp kann also innerhalb eines Programms geändert werden, ohne einen neuen Namen festlegen zu müssen. Die Basis-Datentypen, auch native Datentypen, sind:

- ganze Zahlen: *integers*

  `1, 2, 3`

- Fließkommazahlen: *float*

  `1.1, 2.1, 3.1`

- Zeichenketten: *string*

  `"Zeichenkette"`

- Wahrheitswerte: *boolean*

  `True` oder `False`

3.2.2.4 Bemessungswerte der Einwirkung

Variablen sollten grundsätzlich in Kleinbuchstaben benannt und mit einem Unterstrich getrennt werden, wenn sie aus mehreren Elementen bestehen. Wenn sie jedoch wie nachfolgend aus der Terminologie der Statik anders bekannt sind, sollte dies übernommen werden.

Code 5: Einwirkung

```
N_ed = 2000     # kN - Normalkraft
M_yd = 50     # kNm - Moment um y-Achse
M_zd = 30     # kNm - Moment um z-Achse
```

Auch Kommentare sollen nach einer einheitlichen Regelung verfasst werden. Sie helfen den Code zu umschreiben und sollten daher nicht im Widerspruch zum Skript stehen. Es sollten prägnante und möglichst kurze Formulierungen verwendet werden. Kommentare werden in Python grundsätzlich mit einem Rautezeichen initiiert und sollen ohne Leerzeichen beginnen.

3.2.2.5 Charakteristische Festigkeitswerte

Es werden die charakteristischen Festigkeitswerte der Holzgüte GL28h herangezogen.

Code 6: Charakteristische Festigkeitswerte

```
güte = 'GL28h'
f_c0k = 28*1000     # $kN/m^2$ - Druck
f_myk = 28*1000     # $kN/m^2$ - Biegung
f_mzk = 28*1000     # $kN/m^2$ - Biegung
```

3.2.2.6 Charakteristische Steifigkeitswerte

Neben der Druckfestigkeit parallel zur Faserrichtung und den Biegefestigkeiten wird nach DIN EN 1995-1-1 6.3.1 (2) der 5%-Fraktilwert des Elastizitätsmoduls angesetzt. Der E-Modul $E_{0,05}$ stellt einen Mittelwert dar, bei dem 5% der Prüfergebnisse kleiner ausfielen.

Code 7: Charakteristische Steifigkeitswerte

```
E_05 = 12500*1000     # $kN/m^2$ - E-Modul
G_05 = 10500*1000     # $kN/m^2$ - Schubmodul
```

3.2.2.7 Widerstandsbeiwerte

Die Widerstandsbeiwerte ergeben sich nach der DIN EN 1995-1-1 2.3.1 aus der Klasse der Lasteinwirkungsdauer und der Nutzungsklasse.

Code 8: Widerstandsbeiwerte

```
k_mod = 0.8     # Modifikationsbeiwert
gamma = 1.3     # Teilsicherheitsbeiwert
xi = k_mod/gamma     # Abminderungsbeiwert
```

3.2.3 Berechnung

Nachdem alle Eingabewerte definiert worden sind, beginnt nun die Berechnung. Im Skript wird ab dieser Stelle alles automatisch berechnet.

3.2.3.1 Bemessungswerte der Festigkeit

Mit den zuvor definierten Widerstandsbeiwerten werden die Festigkeitsbeiwerte abgemindert.

Code 9: Bemessungswerte der Festigkeit

```
f_c0d = f_c0k*xi     # $N/mm^2$ - Bemessungswert der Druckfestigkeit
f_myd =f_myk*xi     # $N/mm^2$ - Bemessungswert der Biegefestigkeit
f_mzd =f_mzk*xi     # $N/mm^2$ - Bemessungswert der Biegefestigkeit
```

$$f_{c0d} = f_{c0k} \cdot \xi = 28000 \cdot 0.615 \qquad = 17230.769 \quad (N/mm^2 \text{ - Bemessungswert der Druckfestigkeit})$$

$$f_{myd} = f_{myk} \cdot \xi = 28000 \cdot 0.615 \qquad = 17230.769 \quad (N/mm^2 \text{ - Bemessungswert der Biegefestigkeit})$$

$$f_{mzd} = f_{mzk} \cdot \xi = 28000 \cdot 0.615 \qquad = 17230.769 \quad (N/mm^2 \text{ - Bemessungswert der Biegefestigkeit})$$

3.2.3.2 Querschnittsparameter

Für beide Achsrichtungen werden Querschnittsparameter bestimmt, die für die weitere Berechnung erforderlich sind.

Code 10: Querschnittsparameter

```
A = b*h     # $m^2$

I_y = (b*h**3)/12     # $m^4$ - FTM
w_y = (b*h**2)/6     # $m^3$ - Widerstandsmoment
i_y = h/sqrt(12)     # $m$ - polares Trägheitsmoment

I_z = (h*b**3)/12     # $m^4$ - FTM
w_z = (h*b**2)/6     # $m^3$ - Widerstandsmoment
i_z = b/sqrt(12)     # $m$ - polares Trägheitsmoment
```

$I_y = 0.00267$ (m^4 - FTM) $w_y = 0.01333$ (m^3 - Widerstandsmoment) $i_y = 0.11547$ (m - polares TM)

$I_z = 0.00417$ (m^4 - FTM) $w_z = 0.01667$ (m^3 - Widerstandsmoment) $i_z = 0.14434$ (m - polares TM)

3.2.3.3 Spannungen

Anhand der Querschnittsparameter können die Spannungen berechnet werden.

Code 11: Spannungen

```
sigma_cd = N_ed/A     # $kN/m^2$ - Druckspannung
sigma_myd = M_yd/w     # $kN/m^2$ - Biegespannung um y-Achse
sigma_mzd = M_zd/w     # $kN/m^2$ - Biegespannung um z-Achse
```

$$\sigma_{cd} = \frac{N_{ed}}{A} = \frac{2000}{0.20000} = 10000.00000 \quad (kN/m^2 \text{ - Druckspannung})$$

$$\sigma_{myd} = \frac{M_{yd}}{w_y} = \frac{50}{0.01333} = 3750.00000 \quad (kN/m^2 \text{ - Biegespannung um y-Achse})$$

$$\sigma_{mzd} = \frac{M_{zd}}{w_z} = \frac{30}{0.01667} = 1800.00000 \quad (kN/m^2 \text{ - Biegespannung um z-Achse})$$

3.2.3.4 Knicken

Das Biegeknicken einer Stütze entspricht eines der möglichen Versagensfälle, welches bei Stabilitätsuntersuchungen auftreten kann. Es wird in erster Linie durch Druckbeanspruchungen verursacht, welche im Tragwerk zu seitlichen Auslenkungen führen. Die daraus resultierenden Biegemomente verstärken das plötzliche Versagen der Stütze (vgl. Neuhaus 2017: 594).

Knicklänge

Der Mathematiker Leonhard Euler beschrieb 1744 die Biegeverformbarkeit von Druckstäben mithilfe einer *effektiven Länge*. Dafür definierte er vier verschiedene Lagerungsarten, aus denen sich mit einem *Knicklängenbeiwert* β die Knicklänge ergibt. Anhand dieser Ersatzstablänge l_{ef} wird die der Lagerung entsprechenden Verformungsfigur repräsentiert und die Knickgefährdung berücksichtigt (s. Abbildung 16).

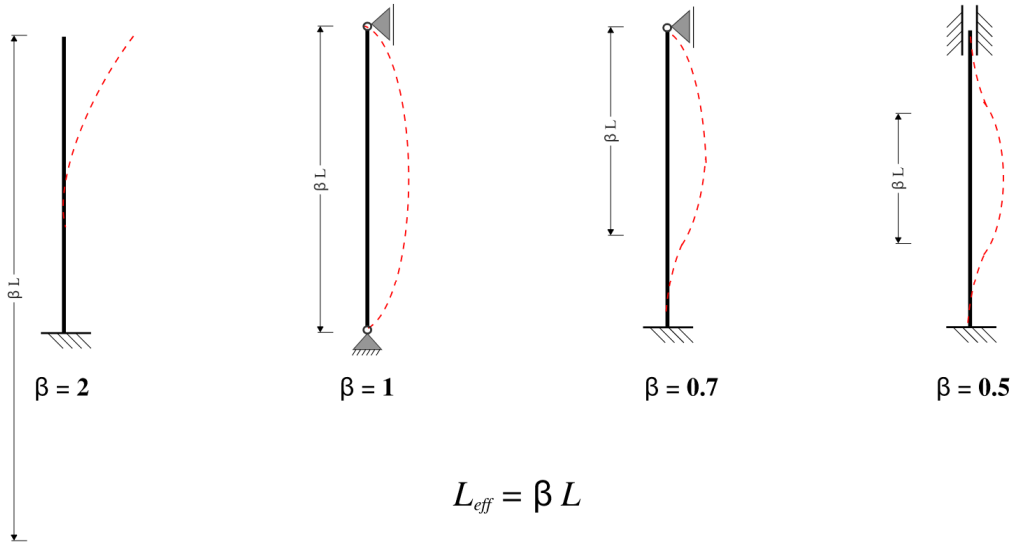


Abbildung 22: Euler-Stäbe

Im Allgemeinen werden bei Stützen, die an den Enden horizontal unverschieblich gehalten sind, auf der sicheren Seite liegend eine gelenkige Lagerung angenommen. Dies entspricht einer Pendelstütze nach dem Eulerfall 2 und ergibt einen β-Wert von 1. Im Code wird dies mit einer if-Abfrage ermittelt.

Code 12: Knicklänge

```
if Lagerung == 'Pendelstütze':
    beta = 1
elif Lagerung == 'Kragstütze':
    beta = 2
elif Lagerung == 'Eingespannt (u/o)':
    beta = 0.7
else:
    beta = 0.5
```

$$l_{ef} = \beta \cdot L = 1 \cdot 3 \qquad = 3 \ \text{(m - Knicklänge)}$$

3.2.3.5 Entscheidungsanweisung

Mittels *if-Abfragen* wird es dem Programm ermöglicht, Entscheidungen zu treffen, die an Bedingungen geknüpft sind. Python bietet mehrere Varianten an. Die klassische und ausführliche Form einer if-Abfrage wird wie folgt formuliert:

```
# Variante 1
if <test>:
    wert_richtig
```

```
elif <test>:
    wert_richtig
```

Dieselbe Anweisung kann ebenso kompakt in einer Zeile erfolgen.

```
if <test>: wert_richtig                    # Variante 2
wert_richtig if <test> else wert_falsch    # Variante 3
wert_richtig, wert_falsch][<test>]         # Variante 4
```

Knicklast

Anhand der effektiven Länge definiert Euler eine *Knicklast* F_{ki}, bei der eine Stütze anfängt, seitlich auszuweichen. Diese lässt sich für einen beidseitig gelenkig gelagerten, ideal geraden Stab mit mittiger Drucklast wie folgt bestimmen:

$$F_{ki} = \frac{(\pi)^2 \cdot E \cdot I}{\left(l_{ef}\right)^2}$$

Schlankheit

Die *Schlankheit* hängt von den Bauteilabmessungen und den Lagerungsbedingungen ab und dient als Indikator für die Einschätzung nach dem Versagensmechanismus. Sie ist definiert als das Verhältnis aus der effektiven Länge des Stabes und dem Trägheitsradius des Querschnitts.

$$\lambda = L_{eff}/i$$

Der *Flächenträgheitsradius i* ist die Wurzel aus dem Quotienten von Flächenträgheitsmoment I und der Querschnittsfläche.

$$i = \sqrt{I/A}$$

Mit Hilfe des bezogenen Schlankheitsgrades wird der Knickbeiwert k_c ermittelt.
Der Knickbeiwert berücksichtigt die verminderte Drucktragfähigkeit einer Stütze bei Knickgefährdung und vergrößert sich entsprechend mit Zunahme der Schlankheit λ. Somit ist ein Spannungsnachweis möglich, bei der die Druckspannung nach Theorie I. Ordnung einer abgeminderten Druckfestigkeit f_{c0d} gegenübergestellt wird (vgl. Neuhaus 2017: 602).

Code 13: Schlankheit

```
# um y-Achse

lamb_y = (beta*L)/i_y     # Schlankheitsgrad
lamb_rel_y = lamb_y/pi*sqrt(f_c0k/E_05)     # bez. Schlankheit
k_y_y = 0.5*(1+0.1*(lamb_rel_y-0.3)+lamb_rel_y**2)     # Beiwert
k_c_y = 1/(k_y_y+sqrt(k_y_y**2-lamb_rel_y**2))     # Knickbeiwert

# um z-Achse

lamb_z = (beta*L)/i_z     # Schlankheitsgrad
lamb_rel_z = lamb_z/pi*sqrt(f_c0k/E_05)     # bez. Schlankheit
k_y_z = 0.5*(1+0.1*(lamb_rel_z-0.3)+lamb_rel_z**2)     # Beiwert
k_c_z = 1/(k_y_z+sqrt(k_y_z**2-lamb_rel_z**2))     # Knickbeiwert
```

$$\lambda_y = \frac{\beta \cdot L}{i_y} = \frac{1 \cdot 3}{0.115} \qquad = 25.981 \quad \text{(Schlankheitsgrad)}$$

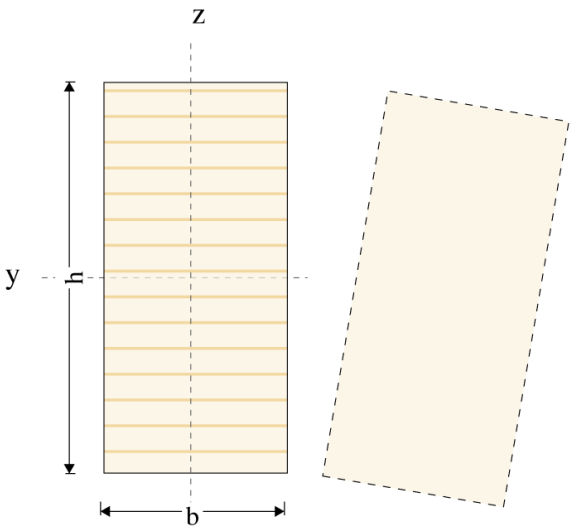
$$\lambda_{rel_y} = \frac{\lambda_y}{\pi} \cdot \sqrt{\frac{f_{c0k}}{E_{05}}} = \frac{25.981}{3.142} \cdot \sqrt{\frac{28000}{10500000}} \qquad = 0.427 \quad \text{(bezogene Schlankheit)}$$

$$\begin{aligned} k_{y_y} &= 0.5 \cdot \left(1 + 0.1 \cdot \left(\lambda_{rel_y} - 0.3\right) + \left(\lambda_{rel_y}\right)^2\right) \\ &= 0.5 \cdot \left(1 + 0.1 \cdot (0.427 - 0.3) + (0.427)^2\right) \\ &= 0.598 \quad \text{(Beiwert)} \end{aligned}$$

$$k_{c_y} = \frac{1}{k_{y_y} + \sqrt{\left(k_{y_y}\right)^2 - \left(\lambda_{rel_y}\right)^2}} = \frac{1}{0.598 + \sqrt{(0.598)^2 - (0.427)^2}} \qquad = 0.985 \quad \text{(Knickbeiwert)}$$

3.2.3.6 Kippen

Ein weiterer Versagensmechanismus, der im Rahmen einer Stabilitätsprüfung untersucht werden muss, ist das Biegedrillknicken (auch Kippen). Das Kippen stellt ebenfalls eine Instabilität infolge einer Druckbeanspruchung dar, bei der ein Stab gleichzeitig seitlich ausweicht und sich verdreht.



$$h > b \rightarrow W_y = \frac{b\,h^2}{6} > W_z = \frac{h\,b^2}{6}$$

Abbildung 23: Versagensmechanismus Kippen

$$\lambda_{relm} = \sqrt{\frac{l_{ef}}{\pi \cdot (b)^2}} \cdot \sqrt{\frac{f_{mk}}{\sqrt{E_{05} \cdot G_{05}}}}$$

$$k_{\text{crit}} = \begin{cases} 1 & \text{für} \quad \lambda_{\text{rel,m}} \leq 0{,}75 \\ 1{,}56 - 0{,}75\lambda_{\text{rel,m}} & \text{für} \quad 0{,}75 < \lambda_{\text{rel,m}} \leq 1{,}4 \\ \dfrac{1}{\lambda_{\text{rel,m}}^2} & \text{für} \quad 1{,}4 < \lambda_{\text{rel,m}} \end{cases}$$

Der Kippbeiwert k_{crit} wird mit dem bezogenen Kippschlankheitsgrad $\lambda_{rel,m}$ um die starke Achse berechnet und mindert die entsprechende Biegespannung ab. Die starke Achse wird dabei als diejenige Achse definiert, bei der das Widerstandsmoment W_y größer ist (vgl. Neuhaus 2017: 630).

$$\lambda_{relm} = \sqrt{\frac{l_{ef}}{\pi \cdot (b)^2}} \cdot \sqrt{\frac{f_{mk}}{\sqrt{E_{05} \cdot G_{05}}}}$$
$$= \sqrt{\frac{3}{3.142 \cdot (0.400)^2}} \cdot \sqrt{\frac{28000}{\sqrt{10500000 \cdot 540000}}}$$
$$= 0.265 \quad \text{(bez. Schlankheitsgrad)}$$

$$\text{Since, } \lambda_{relm} \leq 0.75 \rightarrow (0.265 \leq 0.75): \quad \text{(Kippbeiwert)}$$
$$k_{crit} = 1$$

3.2.4 Nachweise

Die Nachweise werden in Abhängigkeit der Versagensmechanismen (Knicken und Kippen) und der vorhandenen Schlankheit gewählt. Dadurch ergeben sich folgende Möglichkeiten.

Tabelle 3: Mögliche Nachweise bei der Stützenbemessung

EC5 Abs.	Knicken	Kippen
6.2.4	nein	nein
6.3.2	ja	nein
6.3.3	ja/nein	ja

Die Nachweise werden in beide Richtungen geführt, wobei die höhere Ausnutzung maßgebend wird. Mit dem Beiwert k_m muss hinzu die Inhomogenitäten des Holzes im Querschnitt berücksichtigt werden (k_m = 0.7 für Brettschichtholz). Beim Nachweis der zweiachsigen Biegung wird dieser nacheinander mit der einwirkenden Biegespannung multipliziert. Bei einachsiger Momentenbeanspruchung entfällt der Beiwert.

3.2.4.1 Listen

Die Herausforderung beim Umsetzen des Codes ist nun dabei, eine allgemeingültige Programmierung zu entwickeln, die sämtliche Kombinationsmöglichkeiten richtig abbildet. Ein wichtiges Werkzeug, welches permanent zum Einsatz kommt ist die Nutzung von **Listen**. Es ist eines der unterschiedlichen Möglichkeiten in Python mehrere Werte abzuspeichern. Bei der Bearbeitung mit Datenstrukturen geht es im Wesentlichen um das Zugreifen, Einfügen und Löschen von Elementen und ermöglicht eine bestimmte Struktur der Informationen beizubehalten.

Die Definition von Listen erfolgt analog zu der Definition einer Variablen. Die Einträge werden mit einem Komma getrennt und mit einer eckigen Klammer umfasst.

```
liste_1 = [5, 6, 7, 8]

liste_2 = [[0, 1, 2], 3, 4](verschachtelte Liste)
```

Zugriff auf Listenelemente

Auf Listenelemente wird mit einer eckigen Klammer zugegriffen. Bei einer verschachtelten Liste werden zwei Klammern benötigt.

```
element_1 = liste_1[1]

element_2 = liste_2[0][1]
```

Einfügen eines Elementes

Mit der `append()`-Funktion werden neue Elemente einer Funktion angehängt.

```
liste_1.append(9)
```

Einfügen mehrerer Elemente

Möchte man mehrere Elemente anhängen, muss man mit der `extend()`-Funktion die Elemente in Form einer Liste anhängen.

```
list_2_new = [5,6]

liste_2.extend(list_2_new)
```

3.2.4.2 Umsetzung in Python

1. Der Nachweis wird mit Hilfe einer Schleife mit if-Abfragen gewählt. Die Schleife funktioniert, indem die Variable M nacheinander durch die Liste der Momenten `L_Md` iteriert und die Werte `M_yd` und `M_zd` annimmt. Der Befehl enumerate() wird zusammen mit der Variable `n` verwendet und stellt einen Zähler dar, der in diesem Fall die Werte `0` und `1` zugewiesen bekommt. Diese wird innerhalb der Schleife verwendet, um auf das `n`-te Element der Listen zuzugreifen.

```
L_Md = [M_yd, M_zd]

## Schleife für Nachweise um y- und z-Achse

for n,M in enumerate(L_Md):
```

2. In den if-Abfragen wird gemäß Tabelle 3 bestimmt, welche Versagensmechanismen maßgebend sind. Es wird mit dem `and` Operator geprüft, ob das Knicken und/oder das Kippen relevant ist.

```
if k_crit == 1 and lamb_rel < 0.3:

        eta = (sigma_cd/f_c0d)**2 + L_km[n][0]*sigma_myd/f_
myd +

                L_km[n][1]*sigma_mzd/f_mzd
```

3. Die Positionen bestimmter Beiwerte (z.B. k_m-Beiwert) variieren je nach Nachweis um die y- oder z-Achse. Daher werden sie vor der Schleife in Listen gespeichert und mittels Index-Zugang richtig zugeordnet.

```
L_km = [[1,0.7],[0.7,1]]
```

Beispiel: In der Schleife nimmt die Variabel `n` nacheinander für die y- und z-Achse den Wert 0 und 1 an. Im ersten Durchgang der Schleife wird mit L_km[n] die erste Liste innerhalb der verschachtelten Liste angesprochen und mit L_km[n`[0]` = 1 und L_km[n][1] = 0.7 die Werte richtig positioniert.

```
L_km = [[1,0.7],[0.7,1]]
n = 0  (um y-Achse)
L_km[n][0] = 1     L_km[n][1] = 0.7
```

$$\left(\frac{\sigma_{cd}}{f_{c0d}}\right)^2 + k_m \frac{\sigma_{myd}}{f_{myd}} + k_m \frac{\sigma_{mzd}}{f_{mzd}}$$

```
L_km[n][0] = 0.7  L_km[n][1] = 1
n = 1  (um z-Achse)
```

Diese Vorgehensweise wird ebenfalls für die Potenz im Kippnachweis und dem Knickbeiwert k_c umgesetzt.

```
L_kc = [k_c_y, k_c_z]      # Knickbeiwerte

L_pot = [[1,2],[2,1]]      # Exponenten für Kippnachweise
```

4. Für den Fall einer einachsigen Momentenbeanspruchung muss vorher noch sichergestellt werden, dass der k_m-Wert gleich 1 gesetzt wird.

Code 14: Ersatzstabverfahren – Nachweis

```
# Nachweise
## Listen
L_Md = [M_yd, M_zd]     # Momente
lamb_rel = min(lamb_rel_y, lamb_rel_z)     # bez. Schlankheiten
L_km = [[],[]]     # km-Beiwerte
L_kc = [k_c_y, k_c_z]     # Knickbeiwerte
L_pot = [[1,2],[2,1]]     # Exponenten für Kippnachweise

## if-Abfrage für km-Beiwert
if 0 in L_Md:
    index = L_Md.index(0)
    L_km[0][0] = L_km[0][1] = L_km[1][0] = L_km[1][1] = 1
else:
    L_km = [[1,0.7],[0.7,1]]

## Schleife für Nachweise um y- und z-Achse
for n,M in enumerate(L_Md):


    ### EC5 Abs. 6.2.4: Biegung m/o Druck ohne Knicken ohne Kippen
    if k_crit == 1 and lamb_rel < 0.3:
        eta = (sigma_cd/f_c0d)**2 + L_km[n][0]*sigma_myd/f_myd +
                L_km[n][1]*sigma_mzd/f_mzd

    ### EC5 Abs. 6.3.2: Biegung m/o Druck mit Knicken ohne Kippen
    elif k_crit == 1 and lamb_rel > 0.3:
        eta = (sigma_cd)/(f_c0d*L_kc[n]) + L_km[n][0]*sigma_myd/f_myd +
               L_km[n][1]*sigma_mzd/f_mzd

    ### EC5 Abs. 6.3.3: Biegung m/o Druck mit Knicken und Kippen
    elif k_crit < 1:
        eta = (sigma_cd)/(f_c0d*L_kc[n]) +
               (sigma_myd/(f_myd*L_k_crit[0]))**L_pot[n][0] +
               (sigma_mzd/(f_mzd*L_k_crit[1]))**L_pot[n][1]
```

Maßgebend ist der Knicknachweis nach der DIN EN 1995-1-1 Abs. 6.3.2 (3).

Since, $k_{crit} = 1 \rightarrow (1 = 1)$: (EC5 Abs. 6.3.2: Biegung m/o Druck mit Knicken ohne Kippen)

$$\eta_y = \frac{\sigma_{cd}}{f_{c0d} \cdot k_{c_y}} + 1 \cdot \frac{\sigma_{myd}}{f_{myd}} + 0.7 \cdot \frac{\sigma_{mzd}}{f_{mzd}}$$

$$= \frac{10000.00}{17230.77 \cdot 0.98} + 1 \cdot \frac{3750.00}{17230.77} + 0.7 \cdot \frac{1800.00}{17230.77}$$

$$= 0.88$$

$$\text{Since, } k_{crit} = 1 \rightarrow (1 = 1): \quad \text{(EC5 Abs. 6.3.2: Biegung m/o Druck mit Knicken ohne Kippen)}$$

$$\begin{aligned}\eta_z &= \frac{\sigma_{cd}}{f_{c0d} \cdot k_{c_z}} + 0.7 \cdot \frac{\sigma_{myd}}{f_{myd}} + 1 \cdot \frac{\sigma_{mzd}}{f_{mzd}} \\ &= \frac{10000.00}{17230.77 \cdot 1.00} + 0.7 \cdot \frac{3750.00}{17230.77} + 1 \cdot \frac{1800.00}{17230.77} \\ &= 0.84\end{aligned}$$

Dateien

Die Statikvorlage für eine Stützenbemessung nach dem Ersatzstabverfahren als Jupyter Notebook befindet sich im Ordner *01_Stützenbemessung_ec5_63* unter dem Namen *ec5_63_Stützenbemessung_Ersatzstabverfahren.ipynb.* Ein Ausdruck des Bemessungsbeispiels ist dem Anhang A5, der Python Code des Kapitels 3.2 der Datei *3.2_Anwendungsbeispiel_1_Stabilitätsnachweis_Ersatzstabverfahren.ipynb* zu entnehmen.

3.3 Stabilitätsnachweis nach der Theorie II. Ordnung

Vor allem bei Stützen können baupraktisch unvermeidbare Imperfektionen und Vorverformungen zu einer Knickgefährdung des Tragwerks führen, sodass die Betrachtung des Gleichgewichts am verformten System maßgeblich sein kann. Als Alternative zur Bemessung mit dem Ersatzstabverfahren wird die Tragfähigkeit nach der Theorie II. Ordnung (auch p-Delta Verfahren) nachgewiesen, indem mit einer Vorverformung das Gleichgewichtssystem aufgestellt wird. Bei nichtlinearer Schnittgrößenermittlung kann jedoch zur Vereinfachung weiterhin das linear-elastische Materialverhalten als Idealisierung herangezogen werden (vgl. Neuhaus 2017: 821).

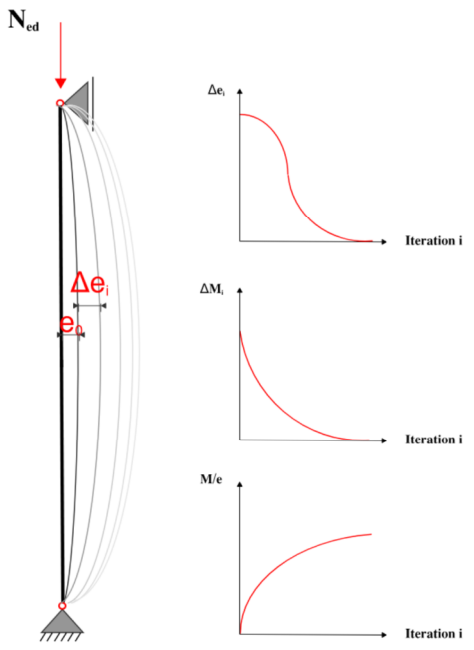


Abbildung 24: Stützenbemessung nach der Theorie II. Ordnung

Das Angreifen der Last am verformten System führt durch die Exzentrizität zu einer Biegebeanspruchung, die wiederum zu einer zusätzlichen Auslenkung führt und das Moment überproportional ansteigen lässt. Dieses geometrisch nicht-lineare Verhalten erfordert bei der Berechnung einen iterativen Ansatz, welcher mit Python gut lösbar ist.

Die Stützenbemessung nach der Theorie II. Ordnung wird anhand desselben vorherigen Beispiels fortgeführt.

Anmerkung: Für das Bemessungsverfahren nach der Theorie II. Ordnung wird im Folgenden nur die Berechnung einer Pendelstütze behandelt. Für weitere Systeme müsste die Formel für die Verformung entsprechend angepasst werden.

3.3.1 Vorgehensweise nach EC5

Die Vorgehensweise der iterativen Ermittlung der Schnittgrößen ist wie folgt:

Theorie I. Ordnung

Zunächst wird anhand des vorgegebenen Imperfektionsbeiwertes die ungewollte Ausmitte der Stütze ermittelt und führt zum Moment infolge der Vorverformung. Mit dem einwirkenden Moment um die y- oder z-Achse ergibt sich das Gesamtmoment

nach Theorie I. Ordnung. Nach der DIN EN 1995-1-1 Abs. 5.4.3 ist für seitlich unverschiebliche Systeme eine Vorverformung von θ=0,0025 anzusetzen.

1. Vorverformung

$$e_0 = \theta * L$$

2. Moment infolge Vorverformung

$$M_0 = e_0 * N_{ed}$$

3. Gesamtmoment

$$M = M_0 + M_{ed}$$

Iteration nach Theorie II. Ordnung

Die Schnittgrößenermittlung nach der Theorie II. Ordnung erfolgt mit der Formel für die Durchbiegung eines Einfeldbalkens. Abhängig ist die Verformung von der Länge und Biegesteifigkeit des Stabes. Mit der ermittelten Verformung errechnet sich das Moment nach Theorie II. Ordnung. Um die Iteration fortzuführen, wiederholt man diesen Vorgang mit dem neu gerechneten Moment, bis die zusätzliche Verformung gegen Null konvergiert.

4. Verformung infolge zusätzlichen Moments

$$e_i = \frac{M_i * L^2}{E * I * \pi^2}$$

5. Moment infolge zusätzlicher Verformung

$$M_i = e_i * N_{ed}$$

Ergebnis

Das Ergebnis der Schnittgrößenermittlung nach Theorie II. Ordnung ergibt sich aus der Summe der Momente.

6. Gesamtverformung

$$e_{tot} = e_0 + \sum e_i$$

7. Gesamtmoment

$$M_{tot} = M_0 + \sum M_i$$

3.3.2 Vorgehensweise in Python

3.3.2.1 Iterationsanweisung

Eines der größten Vorteile eines Computers ist die hohe Rechenleistung und damit einhergehend die hohe Geschwindigkeit, wodurch Rechenaufgaben tausendfach in wenigen Sekunden berechnet werden können.
Diesen Vorteil kann man besonders zum Einsatz bringen, wenn diese Rechenaufgaben in einer Schleife verpackt werden, in denen bestimmte Vorgänge automatisch wiederholt werden.

Schema

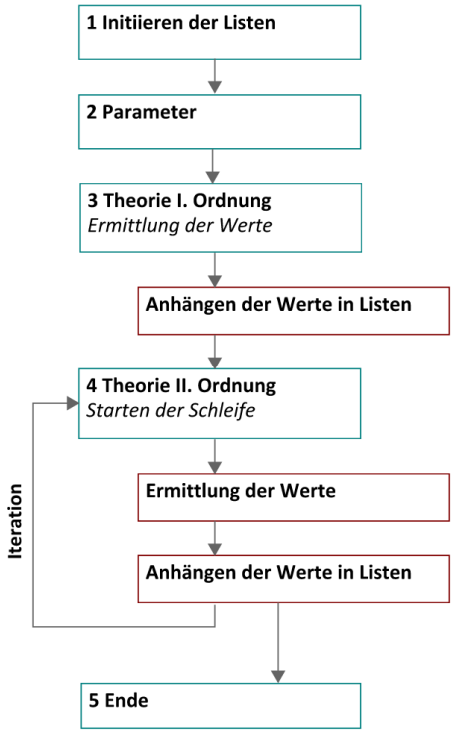


Abbildung 25: Schema zum Iterationsvorgang

Es wird nun die Umsetzung in Python angeführt. Auch hier wird zunächst nur *ein* Iterationsschritt gezeigt. Um die verschiedenen Ergebnisse zu speichern, müssen zunächst Listen erstellt werden, in denen sie mit der `append()`-Funktion beigefügt werden.

1. Initiieren der Listen

```
L_e = []
L_M = []
L_e_total = []
L_M_total = []
```

Theorie I. Ordnung

2. Vorverformung

```
e_0 = theta*L
```

3. Moment infolge Vorverformung

```
M_0 e_0*N_ed
```

4. Gesamtmoment

```
M = M_0+M_ed
```

5. Anhängen der Werte in Listen

```
L_M.append(M)
L_e.append(e_0)
```

Theorie II. Ordnung

Die Momente werden in der Liste `L_M` gespeichert. Mittels `L_M[i]` wird auf das zuletzt hinzugefügte Moment zugegriffen.

6. Verformung infolge zusätzlichem Moment

```
e_i = (L_M[i]*L**2)/(E*I*math.pi**2)
```

7. Moment infolge zusätzlicher Verformung

```
M_i = e_i*N_ed
```

8. Anhängen der Werte in Listen

```
L_e.append(e_i)

L_M.append(M_i)
```

Ergebnis

Für die Ermittlung der Gesamtverformung und -moment wird mit der `sum()`– Funktion die Summe der Listen gebildet.

9. Gesamtverformung

```
e_total = sum(L_e)

L_e_total.append(e_total)
```

10. Gesamtmoment

```
M_total = sum(L_M)

L_M_total.append(M_total)
```

3.3.2.2 Iteration (als Schleife)

Die Wiederholung der Iterationen soll nun in einer Schleife ausgeführt werden. Mit der Definition der Variable `no_iter` kann die Anzahl der Iterationen bestimmt werden.

Es ist nun darauf zu achten, dass die Wertereihe bei `0` anfängt und somit der Wert der Zahl um eine Zahl versetzt ist. Daher muss der Bereich für die Iteration entsprechend mit `n-1` angepasst werden. Geschieht dies nicht, erscheint eine Fehlermeldung, da die Schleife somit ins „*Leere*“ laufen würde.

Zusätzlich soll als Nächstes eine weitere Schleife integriert werden, die nacheinander die Iterationsberechnung für das Moment um die y-Achse und um die z-Achse durchführt. Wieder soll der Code so geschrieben werden, dass er allgemeingültig für alle Fälle anwendbar ist.

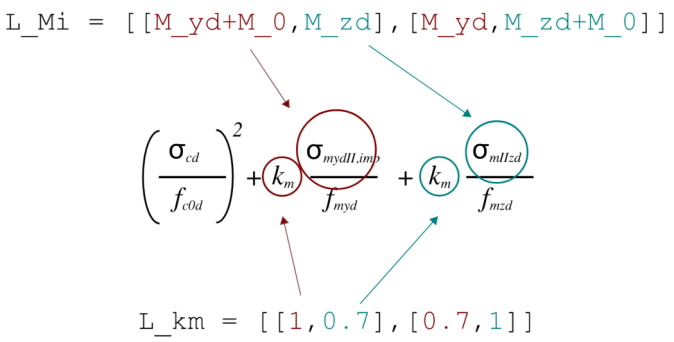


Abbildung 26: Zuordnung von Listenwerten im Nachweis

Der Nachweis erfolgt nach der DIN EN 1995-1-1 Abs. 6.2.4 als Biegespannungsnachweis.

Bei zweiaxialer Biegebeanspruchung ist die Vorverformung nur an der ungünstigen Achse anzusetzen. Um dies zu gewährleisten, muss gleichzeitig der k_m-Beiwert mit der Spannung multipliziert werden, bei der nicht die Vorverformung angesetzt wurde. Um dies im Code widerzuspiegeln, werden alle Parameter analog zu Kapitel 3.2.4.2 in verschachtelten Listen gespeichert, um alle Berechnungen strukturiert durchzuführen. Daraus ergeben sich folgende Kombinationen für die Listenanordnung:

Moment um *y* mit Imperfektion + Moment um *z*

Moment um *y* + Moment um *z* mit Imperfektion

Code 15: Iteration für Schnittgrößenermittlung nach Theorie II. Ordnung

```
#Beiwert k_m
if M_yd == 0 or M_zd == 0:
    L_km = [[1,1],[1,1]]
else:
    L_km = [[1,0.7],[0.7,1]]

#Listen
L_bh = [[b,h],[h,b],[b,h],[h,b]]
L_e = [[e_0*1000],[0],[0],[e_0*1000]]
L_M = [[M_yd+M_0],[M_zd],[M_yd],[M_zd+M_0]]
L_e_total = [[e_0],[0],[0],[e_0]]
L_M_total = [[M_yd+M_0],[M_zd],[M_yd],[M_zd+M_0]]
L_sigma_mIId = []
L_Mi = [M_yd+M_0, M_zd, M_yd, M_zd+M_0]
L_km = [[1,0.7],[0.7,1]]

E = E_0meand

#Schleife 1: Schnittgrößenermittlung nach Theorie II. Ordnung
for n in range(4):

    ##Querschnittsparameter
    A = b*h                                  #$m^2$
    I = (L_bh[n][0]*L_bh[n][1]**3)/12        #$m^4$ - FTM
    w = (L_bh[n][0]*L_bh[n][1]**2)/6         #$m^3$ - Widerstandsmoment
    i = L_bh[n][1]/sqrt(12)                  #$m$ - polares Trägheitsmoment

    ##Theorie II. Ordnung
    #Schleife 2
    for i in range(no_iter):

        ##Ermittlung der Werte
        e_i = (L_M[n][i]*l_ef**2)/(E*I*pi**2)
        M_i = e_i * N_ed

        ##Anhängen der Werte in Listen
        L_e[n].append(round(e_i*1000,1))
        L_M[n].append(round(M_i,2))
        e_total = sum(L_e[n])
        M_total = sum(L_M[n])
        L_e_total[n].append(round(e_total,2))
        L_M_total[n].append(round(M_total,2))

    #Spannungen
    sigma_mIId = L_M_total[n][-1]/w #$kN/m^2$
    L_sigma_mIId.append(round(sigma_mIId,2))
```

Schleife für M_yd_imp, M_zd und M_yd, M_zd_imp

Iterationsschleife

Ermittlung *e* und

Δ*e* und Δ*M*

∑*e* und ∑*M*

L_∑*e* und L_∑*M*

Lässt man nach dem Durchlaufen der Schleife die Liste `L_M_total` ausgeben, sieht man die verschachtelte Struktur der Liste. Auf diese Weise sind gemäß der zuvor definierten Anordnung der Vorverformungen die Unterschiede der Gesamtmomente erkennbar.

```
L_Mi = [[M_yd+M_0,M_zd],[M_yd,M_zd+M_0]]

In [119]:  1  L_M_total

Out[119]: [[65.6, 71.62, 72.17, 72.22, 72.22, 72.22],
           [30, 31.76, 31.86, 31.87, 31.87, 31.87],
           [50, 54.59, 55.01, 55.05, 55.05, 55.05],
           [45.6, 48.28, 48.44, 48.45, 48.45, 48.45]]
```

Abbildung 27: Versagensmechanismus Kippen

3.3.3 Grafische Auswertung der Ergebnisse

3.3.3.1 Tabellen

Für das Visualisieren der Ergebnisse werden zunächst mit der Bibliothek **pandas** Tabellen erstellt. Dies erfolgt, indem für die y- und z-Richtung jeweils Dictionaries erstellt werden. Die Ergebnisse wurden in der Schleife in Listen gespeichert, sodass diese mit dem jeweiligen Index zugegriffen werden können.

Code 16: Tabellen mit pandas-DataFrames

```
#um y-Achse
##Erstellen eines Dictionaries
dict_y = {"e_i [mm]": L_e[0], "M_i [kNm]": L_M[0], "e_tot [mm]": \
          L_e_total[0], "M_tot [kNm]": L_M_total[0]}

#Erstellen eines DataFrames
df_y = pd.DataFrame(dict_y)

#Anzeigen des DataFrames
display(df_y)


#um z-Achse
##Erstellen eines Dictionaries
dict_z = {"e_i [mm]": L_e[1], "M_i [kNm]": L_M[1], "e_tot [mm]": \
          L_e_total[1], "M_tot [kNm]": L_M_total[1]}

#Erstellen eines DataFrames
df_z = pd.DataFrame(dict_z)

#Anzeigen des DataFrames
display(df_z)
```

In der folgenden Tabelle sind die Ergebnisse der Schnittgrößenermittlung nach der Theorie II. Ordnung dargestellt. Mit der Variablen `no_iter = 5` wurde die Anzahl der Iterationsschritte festgelegt. Da in der Informatik die Zahlenreihe bei `0` beginnt, entspricht dies sechs Iterationen. In den ersten beiden Spalten nach den Indices sind die Werte für die Zusatzverformung und dem resultierenden Moment dargestellt. Hier

ist erkennbar, dass bereits nach vier Schritten die Werte gegen Null konvergieren. Die Spalten 3 und 4 beinhalten die aufsummierten Werte.

Schnittgrößenermittlung nach Theorie II. Ordnung

	e_i [mm]	M_i [kNm]	e_tot [mm]	M_tot [kNm]
0	6.0	65.60	0.006	65.60
1	2.3	6.02	8.300	71.62
2	0.2	0.55	8.500	72.17
3	0.0	0.05	8.500	72.22
4	0.0	0.00	8.500	72.22
5	0.0	0.00	8.500	72.22

Ergebnisse für M_yd+M_0

	e_i [mm]	M_i [kNm]	e_tot [mm]	M_tot [kNm]
0	0.0	30.00	0.0	30.00
1	0.7	1.76	0.7	31.76
2	0.0	0.10	0.7	31.86
3	0.0	0.01	0.7	31.87
4	0.0	0.00	0.7	31.87
5	0.0	0.00	0.7	31.87

Ergebnisse für M_zd

```
L_Mi = [[M_yd+M_0,M_zd],[M_yd,M_zd+M_0]]
```

Abbildung 28: Schnittgrößen nach Theorie II. Ordnung

3.3.3.2 Grafik

Mitthilfe der Standardbibliothek **matplotlib** können Ergebnisse mit wenig Aufwand als Grafiken dargestellt werden. Ziel dieser Darstellung soll es sein, zwei Liniendiagramme zu erstellen, die die Zusatzverformungen `e_i` und die Gesamtverformungen `e_tot` untereinander darstellen. Nachdem die Bibliotheken importiert wurden, werden im ersten Schritt die Werte für die x- und y-Achsen definiert. Die y-Werte werden wieder aus der Liste entnommen. Für die x-Achse wird mit dem Modul **numpy** eine passende Zahlenanordnung erstellt. Die Funktion enthält 3 Argumente:

```
np.arange(start, stop, schritte)
```

Das Erstellen von Grafiken erfolgt grundsätzlich immer nach demselben Schema. Im ersten Schritt wird ein leerer Plot erzeugt, der anschließend mit Informationen für die Achsen und für die Beschriftungen gefüllt wird. Bei der Darstellung von mehreren Grafiken muss die Funktion `plt.subplot()` mit der Zahl `211` für die Anzahl der Zeilen, Spalten und Nummer der aktuellen Grafik definiert werden.

```
plt.subplot(Zeilen, Spalten, Nummer)
```

Code 17: Liniendiagramme mit pandas-DataFrames

```
#Importieren
import matplotlib.pyplot as plt

#Daten für x- und y-Achsen
x = np.arange(0, 6, 1)
y1 = L_e[0]
y2 = L_e_total[0]

#Plot 1: Verformung e_i
plt.figure(figsize=(8, 6), dpi=110)     #Initiieren eines Plots
plt.subplot(211)                        #Zeilen, Spalten, Nummer
plt.plot(x, y1,'orange')                #Einfügen der Daten
plt.ylabel('$\delta e_i$ [mm]')         #Beschriftung der y-Achse
plt.xlabel('Iteration')                 #Beschriftung der x-Achse
plt.title('Verformung $\delta e_i$')    #Titel

#leerer Plot als Abstandhalter
plt.figure(figsize=(8, 6), dpi=110)
plt.plot('', '', '')

#Plot 2:Summe der Verformungen e_i
plt.subplot(212)
plt.plot(x, y2, 'orange')
plt.ylabel('$\sum e_i$ [mm]')
plt.xlabel('Iteration')
plt.title('Verformung $\sum e_i$')
```

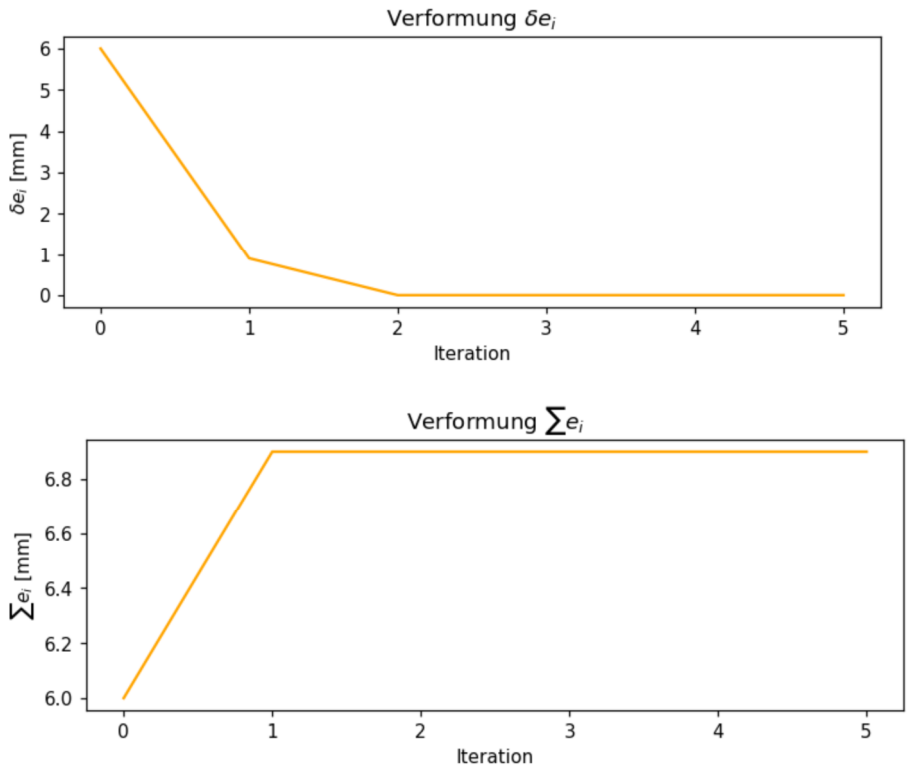


Abbildung 29: Verformungen nach Theorie II. Ordnung

Analog dazu erfolgt die Darstellung der Ergebnisse für die Momente. Der vollständige Code ist dem Anhang zu entnehmen.

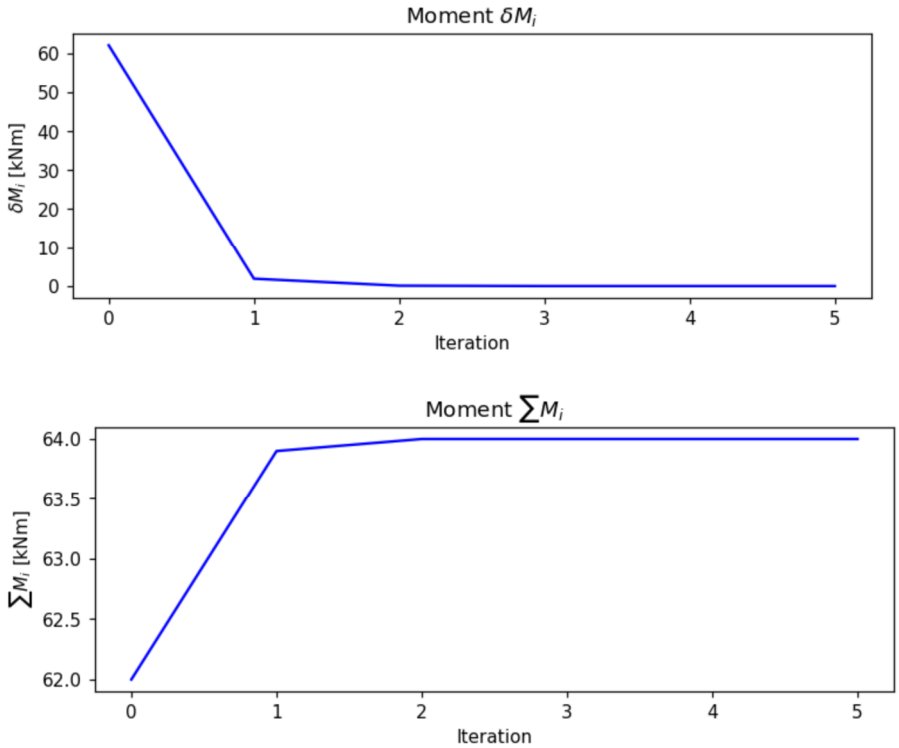


Abbildung 30: Momente nach Theorie II. Ordnung

3.3.4 Nachweise

Im Spannungsnachweis nach DIN EN 1995-1-1 Abs. 6.2.4 des Zahlenbeispiels ist nun ersichtlich, dass die km-Beiwerte richtig zugeordnet wurden, sodass die ungünstigste Überlagerung der Spannungen sichergestellt wurde.

$$\eta_y = \left(\frac{\sigma_{cd}}{f_{c0d}}\right)^2 + k_{m1} \cdot \frac{\sigma_{my2d_{imp}}}{f_{myd}} + k_{m2} \cdot \frac{\sigma_{mz2d}}{f_{mzd}}$$
$$= \left(\frac{13000.00}{17230.77}\right)^2 + 1 \cdot \frac{\boxed{5416.50}}{17230.77} + 0.70 \cdot \frac{\boxed{1912.20}}{17230.77}$$
$$= 0.96$$

$$\eta_z = \left(\frac{\sigma_{cd}}{f_{c0d}}\right)^2 + k_{m3} \cdot \frac{\sigma_{y2d}}{f_{myd}} + k_{m4} \cdot \frac{\sigma_{mz2d_{imp}}}{f_{mzd}}$$
$$= \left(\frac{13000.00}{17230.77}\right)^2 + 0.70 \cdot \frac{\boxed{4128.75}}{17230.77} + 1 \cdot \frac{\boxed{2907.00}}{17230.77}$$
$$= 0.91$$

Abbildung 31: Spannungsnachweis nach DIN EN 1995-1-1 Abs. 6.2.4

Dateien

Die Statikvorlage für eine Stützenbemessung nach der Theorie II. Ordnung als Jupyter Notebook befindet sich im Ordner *01_Stützenbemessung_ec5_63* unter dem Namen *ec5_63_Stützenbemessung_Theorie II. Ordnung.ipynb.* Ein Ausdruck des Bemessungsbeispiels ist dem Anhang A6, der Python Code des Kapitels 3.3 der Datei *3.3_Anwendungsbeispiel_1_Stabilitätsnachweis_Theorie II. Ordnung.ipynb* zu entnehmen.

3.4 Erstellung von Funktionen

Im Jupyter-Arbeitsblatt *3.4_Anwendungsbeispiel_1_Erstellung_von_Funktionen.ipynb* und im Anhang A1 und A2 befinden sich die Funktionen für die Stützenbemessung nach Theorie I. und II. Ordnung. Diese dienen als Grundlage für die weitere Erstellung des Stützenbemessungsprogramms.

3.5 Verknüpfung mit Excel

3.5.1 Bibliothek: xlwings

Es gibt einige Bibliotheken, die sich mit der Automatisierung und Datenverarbeitung mit Excel befassen (OpenPyXL, XlsxWriter). Für das Erstellen des Stützenbemessungsprogramms wird das Modul **xlwings** genutzt, welches im

Gegensatz zu anderen Modulen die Interaktion mit einer offenen Excel-Arbeitsmappe erlaubt und eine große Ähnlichkeit der Vokabeln mit VBA (range, workbook etc.) aufweist.

Code 18: Importieren der Bibliothek xlwings

```
import xlwings as xw    # Interaktion mit Excel
```

Als Nächstes wird der Code mit der Excel-Arbeitsmappe „Stützenbemessung_ec5_63.xlsm“ verknüpft, indem die Arbeitsmappe und das Arbeitsblatt mit den Funktionen `Book` und `sheets` den Variablen `wb` und `ws` zugeordnet werden. Üblicherweise werden für die Variablen die englischen Abkürzungen für `workbook` und `worksheet` verwendet.

```
wb = xlwings.Book(pfad)

ws = xlwings.Book(pfad).sheets[arbeitsblatt]
```

Für die Verknüpfung mit der Arbeitsmappe ist zunächst jedoch der Pfad der Datei notwendig. Nur wenn die Excel-Datei im selben Arbeitsverzeichnis wie das Skript lokalisiert ist, ist nur der Dateiname ausreichend.

3.5.1.1 Pfade

Die Angaben von Pfaden sind notwendig, um die Verknüpfung mit Daten zu erstellen und dienen der Navigation. Sowohl in Windows als auch in Python werden umgekehrte Schrägstriche " \ " als Trennzeichen verwendet. Da in der Programmierung ein umgekehrtes Schrägzeichen ebenfalls die Funktion eines Zeilenumbruchs innerhalb einer Funktion oder Befehls hat, gibt es die Möglichkeit den Pfad wie folgt zu definieren:

Code 19: Pfade

```
pfad = "C:\\Users\\Cal.Mense\\Desktop"    # doppelte Schrägstriche
pfad = r"C:/Users/Cal.Mense/Desktop"    # Raw-String
```

Eine weitere Schreibweise bietet die Verwendung eines Präfixes `r`. Der Pfad wird somit in ein „Python-Raw-String“ konvertiert, sodass die Ausrichtung der Schrägstriche ignoriert werden.

Relative Pfadangabe

Relative Pfade eignen sich für Dateien, dessen Pfade verändert werden können. Die Position wird relativ zum aktuellen Verzeichnis beschrieben und ist somit unabhängig von der Datenstruktur eines Computersystems. Für das Navigieren wird ein Doppelpunkt für die höhere Ebene und ein einzelner Punkt für das aktuelle Verzeichnis verwendet.

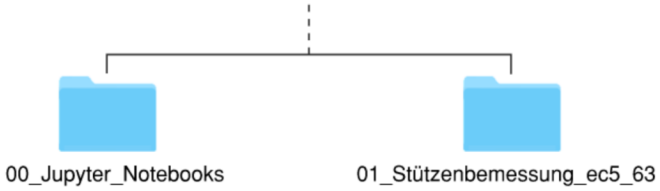


Abbildung 32: Ordnerstruktur

Für die vorhandene Ordnerstruktur befinden sich die Jupyter-Arbeitsblätter im Ordner *00_Jupyter_Notebooks*. Für das Verknüpfen mit der Excel-Datei im Ordner *01_Stützenbemessung_ec5_63* wird der Pfad wie folgt eingegeben. Mit dem Befehl `active()` wird das aktuell offene Arbeitsblatt aktiviert. Als Alternative kann mit dem Namen des Arbeitsblatts in einer eckigen Klammer ein Arbeitsblatt referenziert werden.

Anmerkung: Diese Ordnerstruktur darf nicht verändert werden.

Code 20: Verknüpfung mit Excel

```
pfad = r"..\01_Stützenbemessung_ec5_63\Stützenbemessung_ec5_63.xlsm"
wb = xw.Book(pfad)     # Arbeitsmappe
ws = wb.sheets['Stützenbemessung']     # Arbeitsblatt beim Namen
ws = wb.sheets.active     # Aktives Arbeitsblatt
```

Anmerkung: Die Excel-Datei sollte bereits geöffnet sein, um eine Fehlermeldung zu vermeiden.

3.5.1.2 Range-Objekt

Mit dem `Range`-Objekt können nun einzelne Zellen und Zellbereiche ausgewählt und abgerufen werden. Die Auswahl kann über verschiedene Wege erfolgen.

```
wb.Range('A1')
```

```
wb.Range('A1:C3')

wb.Range((1,1))

wb.Range((1,1), (3,3))

xlwings.Range(Zelle 1, Zelle 2 (optional))

xlwings.Range((Zeile_1, Spalte_1), (Zeile_2, Spalte_2))
```

In Anlehnung an die Schreibweise von Visual Basic wird zunächst die Zeile und dann die Spalte referenziert.

Direkter Verweis

Code 21: Direkter Verweis

```
pos = ws.range('G8').value
pos_nr = ws.range('G10:G15').value
print(str(pos) + ' = ' + str(pos_nr))

Output:
Pos. = ['S-201', 'S-202', 'S-203', 'S-204', 'S-205', 'S-206', 'S-208']
```

Index-Verweis

Ebenso können dieselben Zellen mit der Angabe von Zellen- und Spaltenindizes definiert werden.

Code 22: Index-Verweis

```
pos = ws.range((8,7)).value
pos_nr = ws.range((10,7),(15,7)).value
print(str(pos) + ' = ' + str(pos_nr))

Output:
Pos. = ['S-201', 'S-202', 'S-203', 'S-204', 'S-205', 'S-206', 'S-208']
```

Dynamischer Verweis

Der Vorteil eines Index-Verweises ist die Möglichkeit, die Indizes mit Variablen auszutauschen, sodass der Zellenbezug dynamisch definiert werden kann. Im Folgenden wird einmal, die Variabel `n_pos` an Stelle `Zeile_2` gestellt, um somit die Anzahl der Positionen zu bestimmen. Im zweiten Beispiel wird die Variabel für den Spaltenindex definiert.

Code 23: Dynamischer Verweis

```
# Anzahl der Positionen als Variabel
n_pos = 5
pos = ws.range((8,7)).value
pos_nr = ws.range((10,7),(10+n_pos,7)).value

print(str(pos) + ' = ' + str(pos_nr))

# Spaltenindex als Variabel
n_spalte = 7
pos = ws.range((8,n_spalte)).value
pos_nr = ws.range((10,n_spalte),(10+n_pos,n_spalte)).value

print(str(pos) + ' = ' + str(pos_nr))

Output:
Pos. = ['S-201', 'S-202', 'S-203', 'S-204', 'S-205', 'S-206']
Pos. = ['S-201', 'S-202', 'S-203', 'S-204', 'S-205', 'S-206']
```

Dynamische Anzahl der Tabelleneinträge

Mit der Eigenschaft `current_region.last_cell.row` kann bestimmt werden, wie viele Zeileneinträge in dem Arbeitsblatt vorhanden sind. Bei einer sich verändernden Tabelle passt sich somit die Variabel `row_num` automatisch an.

```
ws.range('B10').current_region.last_cell.row
```

Code 24: Dynamische Anzahl der Tabelleneinträge

```
startrow = 10
rownum = ws.range('G10').current_region.last_cell.row
n_pos = rownum-startrow
print('Die Anzahl der Stützenpositionen beträgt: ' + str(n_pos))

Output:
Die Anzahl der Stützenpositionen beträgt: 62
```

3.5.1.3 Einlesen von Listen

Für das Programmieren einer Anwendung ist es zweckmäßig zunächst die erforderlichen Daten in Listen einzuspeichern. Dieses Vorgehen erhöht die Geschwindigkeit des Codes, da ein ständiges Interagieren mit Excel umgangen wird. Die Listen dienen als Datenbank mit allen benötigten Informationen, auf die jederzeit zugegriffen werden kann. Um die Anzahl der Einträge dynamisch zu definieren, erfolgt das Einlesen wieder mit der Variable `row_num`.

Code 25: Einlesen von Listen

```
startrow = 10
rownum = ws.range('G10').current_region.last_cell.row

# Einlesen der Listen
L_pos = ws.range((startrow,7), (rownum,7)).value
L_ges = ws.range((startrow,8), (rownum,8)).value
L_system = ws.range((startrow,9), (rownum,9)).value
L_L = ws.range((startrow,10), (rownum,10)).value
L_b = ws.range((startrow,11), (rownum,11)).value
L_h = ws.range((startrow,12), (rownum,12)).value
L_N_ed = ws.range((startrow,13), (rownum,13)).value
L_M_yd = ws.range((startrow,14), (rownum,14)).value
L_M_zd = ws.range((startrow,15), (rownum,15)).value
L_h_art = ws.range((startrow,16), (rownum,16)).value
L_f_c0k = ws.range((startrow,17), (rownum,17)).value
L_f_myk = ws.range((startrow,18), (rownum,18)).value
L_E0mean = ws.range((startrow,19), (rownum,19)).value
L_E_05 = ws.range((startrow,20), (rownum,20)).value
L_G_05 = ws.range((startrow,21), (rownum,21)).value
L_k_mod = ws.range((startrow,22), (rownum,22)).value
L_gamma = ws.range((startrow,23), (rownum,23)).value
```

3.5.1.4 Darstellung der Daten

Während des Bearbeitens eines Skriptes ist es zu empfehlen zwischendurch die Ergebnisse zur Kontrolle darzustellen. In der Jupyter Notebook Umgebung eignen sich `DataFrames` (s. 3.3_Anwendungsbeispiel_1_Stabilitätsnachweis_Theorie II. Ordnung.ipynb).

Code 26: Darstellung der Daten in Tabellen mit Pandas DataFrames

```
dict = {'Pos':L_pos, 'Holzart': L_h_art, 'Länge':L_L,
        'b [m]':L_b, 'h [m]':L_h, 'N_ed [kN]':L_N_ed, 'M_yd [kNm]':L_M_yd,
'M_zd [kNm]':L_M_zd}
df = pd.DataFrame(dict)
df.head(10)
```

Tabelle 4: Tabelle mit Pandas DataFrames

	Pos	Holzart	Länge	b [m]	h [m]	N_ed [kN]	M_yd [kNm]	M_zd [kNm]
0	S-201	GL 24h	4.0	0.5	0.5	2400.0	0.0	0.0
1	S-202	GL 24c	4.0	0.5	0.5	2400.0	0.0	0.0
2	S-203	GL 28h	4.0	0.5	0.5	2400.0	0.0	0.0
3	S-204	GL 28c	4.0	0.5	0.5	2400.0	0.0	0.0
4	S-205	GL 32h	4.0	0.5	0.5	2400.0	0.0	0.0
5	S-206	GL 32c	4.0	0.5	0.5	2400.0	0.0	0.0
6	S-207	GL 24h	4.0	0.5	0.5	2400.0	0.0	0.0
7	S-208	GL 24h	4.0	0.5	0.5	2400.0	0.0	0.0
8	S-209	GL 24h	4.0	0.5	0.5	2400.0	0.0	0.0
9	S-210	GL 24h	4.0	0.6	0.6	2800.0	150.0	0.0

3.5.1.5 Iteration

Anhand der eingelesenen Listen soll nun die Stützenbemessung nach der Theorie I. und II. Ordnung nacheinander für alle Stützenpositionen erfolgen. Nachfolgend wird die Vorgehensweise anhand der Bemessung nach dem Ersatzstabverfahren erläutert.

Im *ersten Teil* des Skripts, werden die Vorbereitungen getroffen, indem die Tabellenwerte der Excel-Datei dynamisch eingelesen werden.

Um durch alle Stützenpositionen im *zweiten Teil* zu iterieren, durchläuft die Schleife nacheinander durch den Zahlenbereich der Gesamtanzahl der Positionen. Die Variabel `n_pos` wurde zuvor bereits ermittelt:

```
n_pos = row_num-start_row
```

Code 27: Excel-Iteration: Ersatzstabverfahren – Teil 1

```
# Stützenbemessungsprogramm: Excel-Iteration - Ersatzstabverfahren
# Bibliothek
import xlwings as xw  # Interaktion mit Excel
%run 3.4_Anwendungsbeispiel_1_Erstellung_von_Funktionen.ipynb

# Teil 1: Vorbereitung
# Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws = wb.sheets['Stützenbemessung']

# Dynamisches Einlesen der Zeilenanzahl
startrow = 10  # Startzeile
# Dynamische Anzahl an Reihen
rownum = ws.range('G10').current_region.last_cell.row
ws.range('X10:AH100').value = ""  # Alte Daten LÃ¶schen

# Einlesen der Listen
L_pos = ws.range((startrow, 7), (rownum, 7)).value
L_ges = ws.range((startrow, 8), (rownum, 8)).value
L_system = ws.range((startrow, 9), (rownum, 9)).value
L_L = ws.range((startrow, 10), (rownum, 10)).value
L_b = ws.range((startrow, 11), (rownum, 11)).value
L_h = ws.range((startrow, 12), (rownum, 12)).value
L_N_ed = ws.range((startrow, 13), (rownum, 13)).value
L_M_yd = ws.range((startrow, 14), (rownum, 14)).value
L_M_zd = ws.range((startrow, 15), (rownum, 15)).value
L_h_art = ws.range((startrow, 16), (rownum, 15)).value
L_f_c0k = ws.range((startrow, 17), (rownum, 17)).value
L_f_myk = ws.range((startrow, 18), (rownum, 18)).value
L_E0mean = ws.range((startrow, 19), (rownum, 19)).value
L_E_05 = ws.range((startrow, 20), (rownum, 20)).value
L_G_05 = ws.range((startrow, 21), (rownum, 21)).value
L_k_mod = ws.range((startrow, 22), (rownum, 22)).value
L_gamma = ws.range((startrow, 23), (rownum, 23)).value
...
```

Mit der Variabel i werden nun alle für die Funktion benötigten Werte definiert und in die Funktion für das Ersatzstabverfahren eingefügt.

```
ec5_63_esv(Lagerung,güte,L,b,h,N_ed,M_yd,M_zd,theta, \
                f_c0k,f_myk,f_mzk,E,G_05,k_mod,gamma)
```

Anschließend werden die Ergebnisse ausgewertet und die maßgebenden Zwischenwerte in Excel geschrieben. Um die Zahl zu formatieren, wird das Ergebnis mit `round(Zahl, Nachkommastellen)` auf 2 Nachkommastellen begrenzt.

Code 28: Excel-Iteration: Ersatzstabverfahren – Teil 2

```
# Teil 2: Iteration durch StÃ¼tzenpositionen
for i in range(rownum - startrow + 1):

    # Definition der Variablen des i-ten Elements der Listen
    L = L_L[i]
    b = L_b[i]
    h = L_h[i]
    güte = L_h_art[i]
    Lagerung = L_system[i]
    f_c0k = L_f_c0k[i]*1000
    f_myk = L_f_myk[i]*1000
    f_mzk = f_myk
    E0_mean = L_E0mean[i]*1000
    E_05 = L_E_05[i]*1000
    G_05 = L_G_05[i]*1000
    k_mod = L_k_mod[i]
    gamma = L_gamma[i]
    N_ed = L_N_ed[i]
    M_yd = L_M_yd[i]
    M_zd = L_M_zd[i]

    # Berechnung
    # Kennwerte
    theta = 0
    shi = k_mod/gamma
    f_c0d = f_c0k*shi
    f_myd = f_myk*shi
    E = E_05

    # Stützenbemessung nach Ersatzstabverfahren (Funktion)
    e_0, L_lamb, L_lamb_rel, L_ky, L_kc, k_crit, sigma_cd, L_sigma_md,/
    L_eta, nw = ec5_63_esv(Lagerung, güte, L, b, h, N_ed, M_yd, M_zd,/
                           theta, f_c0k, f_myk, f_mzk, E, G_05, k_mod,/
                           gamma)

    # Auswertung der Ergebnisse
    lamb = max(L_lamb)
    index = L_lamb.index(lamb)
    lamb_rel = max(L_lamb_rel)
    kc = min(L_kc)
    sigma_myd = L_sigma_md[0]
    sigma_mzd = L_sigma_md[1]
    eta = max(L_eta)

    # Schreiben in Excel
    ws.range('Y10').offset(i, 0).value = round(lamb, 2)
    ws.range('Z10').offset(i, 0).value = round(lamb_rel, 2)
    ws.range('AA10').offset(i, 0).value = round(kc, 2)
    ws.range('AB10').offset(i, 0).value = round(k_crit, 2)
    ws.range('AC10').offset(i, 0).value = nw
    ws.range('AD10').offset(i, 0).value = sigma_cd
    ws.range('AE10').offset(i, 0).value = round(sigma_myd, 2)
    ws.range('AF10').offset(i, 0).value = round(sigma_mzd, 2)
    ws.range('AG10').offset(i, 0).value = round(eta, 2)
```

Der vollständige Code ist dem Anhang A3 und A4 und dem Jupyter-Arbeitsblatt *3.5_Anwendungsbeispiel_1_Verknüpfung_mit_Excel.ipynb zu* entnehmen.

3.6 Parameterstudie

3.6.1 Bemessung nach Theorie I. und II. Ordnung

Nachdem die beiden Nachweisverfahren für die Stützenbemessung nach der DIN EN 1995-1-1 für je eine Stützenposition durchgerechnet wurde, soll im Folgenden nun die Verknüpfung mit Excel erfolgen. Dies wird wieder anhand einer Schleife umgesetzt. Um das Skript übersichtlicher zu gestalten, wurden die beiden Nachweisverfahren in Funktionen verpackt und werden gezielt in der Schleife abgerufen.

3.6.2 Variable: Schlankheit

In der folgenden Tabelle sind die Bemessungsergebnisse von zehn Stützen abgebildet, die bei konstanter Belastung und Stützenquerschnitt eine ansteigende Stützenlänge und Schlankheit (λ) aufweisen. In der Spalte I sind die Nachweisverfahren dargestellt, die beim Ersatzstabverfahren verwendet wurden (s. Legende). Die prozentuale Differenz der Ausnutzungen der Bemessungsergebnisse sind in der Spalte M angezeigt.

Nach dem Eurocode Abs. 6.3.2 (2) wird das Knickverhalten erst ab einem bezogenen Schlankheitsgrad von 0,3 maßgebend. Dies entspricht etwa einer Schlankheit von 20. An der Gegenüberstellung der Ausnutzungen η (M) wird deutlich, dass ab der Nachweisführung mit dem Ersatzstabverfahren Eurocode Abs. 6.3.2 (3) ein Knick in den Ergebnissen zu verzeichnen ist und die Ergebnisse ungünstiger ausfallen (s. Diagramm). Der Unterschied Δη liegt ab einem λ-Wert von etwa 20 bei Unterschieden zwischen 13 und 18 Prozent.

Tabelle 5: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (1)

A	B	C	D	E	F	G	H	I	J	K	L	M
		System			Belastung			ESV		Theorie I	Theorie II	
Pos.	Länge	Breite	Höhe	N_{ed}	M_{yd}	M_{zd}	λ	λ_{rel}	Versagen	$\eta_{_Th.1}$	$\eta_{_Th.2}$	$\Delta\eta$
-	[m]	[m]	[m]	[kN]	[kNm]	[kNm]	-	-	-	[%]	[%]	[%]
S-201	1,00	0,60	0,60	2600	30	30	6	0,09	Sp	42%	44%	-2%
S-202	2,00	0,60	0,60	2600	30	30	12	0,19	Sp	42%	45%	-3%
S-203	3,00	0,60	0,60	2600	30	30	17	0,28	Sp	42%	47%	-5%
S-204	4,00	0,60	0,60	2600	30	30	23	0,38	Kn	67%	49%	18%
S-205	5,00	0,60	0,60	2600	30	30	29	0,47	Kn	68%	51%	17%
S-206	6,00	0,60	0,60	2600	30	30	35	0,57	Kn	69%	53%	16%
S-207	7,00	0,60	0,60	2600	30	30	40	0,66	Kn	70%	56%	14%
S-208	8,00	0,60	0,60	2600	30	30	46	0,76	Kn	72%	59%	13%
S-209	9,00	0,60	0,60	2600	30	30	52	0,85	Kn	75%	63%	12%
S-210	10,00	0,60	0,60	2600	30	30	58	0,95	Kn	80%	67%	13%

Legende:
Sp: Spannungsnachweis nach EC5 Abs. 6.2.4
Kn: Knicknachweis nach EC5 Abs. 6.3.2

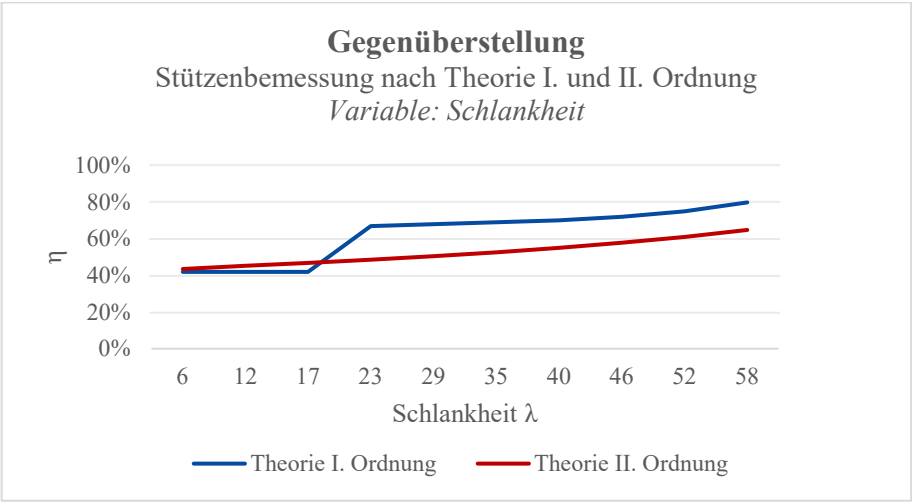


Abbildung 33: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (1)

Bei weiterer Steigerung des Schlankheitsgrades, nähern sich die Ausnutzungskurven an, bis ab einer Schlankheit von ca. 110 die Nachweisführung nach der Theorie II. Ordnung maßgebend wird.

Tabelle 6: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (2)

A	B	C	D	E	F	G	H	I	J	K	L	M
		System			Belastung			ESV		Theorie I	Theorie II	
Pos.	Länge	Breite	Höhe	N_{ed}	M_{yd}	M_{zd}	λ	λ_{rel}	Versagen	$\eta_{_Th.1}$	$\eta_{_Th.2}$	Δη
-	[m]	[m]	[m]	[kN]	[kNm]	[kNm]	-	-	-	[%]	[%]	[%]
S-201	11,00	0,60	0,60	2600	30	30	64	1,04	Kn	**88%**	**70%**	18%
S-202	12,00	0,60	0,60	2600	30	30	69	1,14	Kn	**97%**	**75%**	22%
S-203	13,00	0,60	0,60	2600	30	30	75	1,23	Kn	**109%**	**83%**	26%
S-204	14,00	0,60	0,60	2600	30	30	81	1,33	Kn	**121%**	**92%**	29%
S-205	15,00	0,60	0,60	2600	30	30	87	1,42	Kn	**135%**	**105%**	30%
S-206	16,00	0,60	0,60	2600	30	30	92	1,52	Kn	**151%**	**122%**	29%
S-207	17,00	0,60	0,60	2600	30	30	98	1,61	Kn	**166%**	**145%**	21%
S-208	18,00	0,60	0,60	2600	30	30	104	1,71	Kn	**186%**	**177%**	9%
S-209	19,00	0,60	0,60	2600	30	30	110	1,80	Kn	**204%**	**222%**	-18%
S-210	20,00	0,60	0,60	2600	30	30	115	1,90	Kn	**226%**	**286%**	-60%

Legende:
Sp: Spannungsnachweis nach EC5 Abs. 6.2.4
Kn: Knicknachweis nach EC5 Abs. 6.3.2

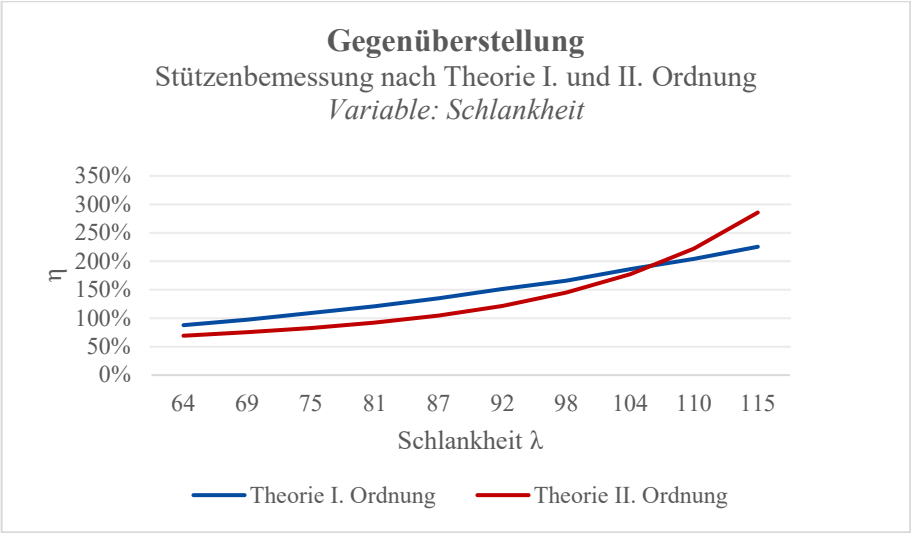


Abbildung 34: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (2)

3.6.3 Theorie II. Ordnung: Gegenüberstellung mit Näherungslösung

Für die Python-gestützte Schnittgrößenermittlung nach der Theorie II. Ordnung wurde der Zuwachs der Verformung durch die aus der Festigkeitslehre bekannten Formeln der Durchbiegung berechnet. Zur Kontrolle soll dieses Verfahren mit einer Näherungslösung gegenübergestellt werden, die nach Neuhaus wie folgt für eine Pendelstütze angeführt wird (2017: 823):

$$\alpha = \frac{1}{(1-\delta)}$$

$$\delta = \frac{N}{N_{crit}}$$

$$N_{crit} = \frac{(\pi^2 * E * I)}{L^2}$$

$$M_{II} = M_I + N * e * \alpha$$

Der α-Vergrößerungsfaktor bestimmt den Momentenzuwachs infolge der unvermeidbaren Vorvervormung des Stabes e. Die Gegenüberstellung der Ergebnisse für das Moment nach Theorie II. Ordnung zeigen eine hohe Genauigkeit für Schlankheitswerte unter 50. Für Schlankheiten über 50 steigt das Moment für das p-Delta Verfahren stärker an (s. Grafik).

Tabelle 7: Gegenüberstellung Theorie II. Ordnung: p-Delta und Näherungsverfahren

Theorie II. Ordnung

	Näherungslösung					p-Delta
λ	I	Ncrit	delta	alpha	M_{II}	M_{II}
-	m^4	kN	-	-	[kNm]	[kNm]
6	0,011	1343056	0,00	1,00	37	37
12	0,011	335764	0,01	1,01	43	44
17	0,011	149228	0,02	1,02	50	51
23	0,011	83941	0,03	1,03	57	59
29	0,011	53722	0,05	1,05	64	68
35	0,011	37307	0,07	1,07	72	78
40	0,011	27409	0,09	1,10	80	89
46	0,011	20985	0,12	1,14	89	102
52	0,011	16581	0,16	1,19	99	118
58	0,011	13431	0,19	1,24	111	138
64	0,011	11100	0,23	1,31	123	148
69	0,011	9327	0,28	1,39	138	172
75	0,011	7947	0,33	1,49	156	202
81	0,011	6852	0,38	1,61	177	241
87	0,011	5969	0,44	1,77	203	293
92	0,011	5246	0,50	1,98	236	363
98	0,011	4647	0,56	2,27	281	460
104	0,011	4145	0,63	2,68	344	594
110	0,011	3720	0,70	3,32	440	782
115	0,011	3358	0,77	4,43	606	1048

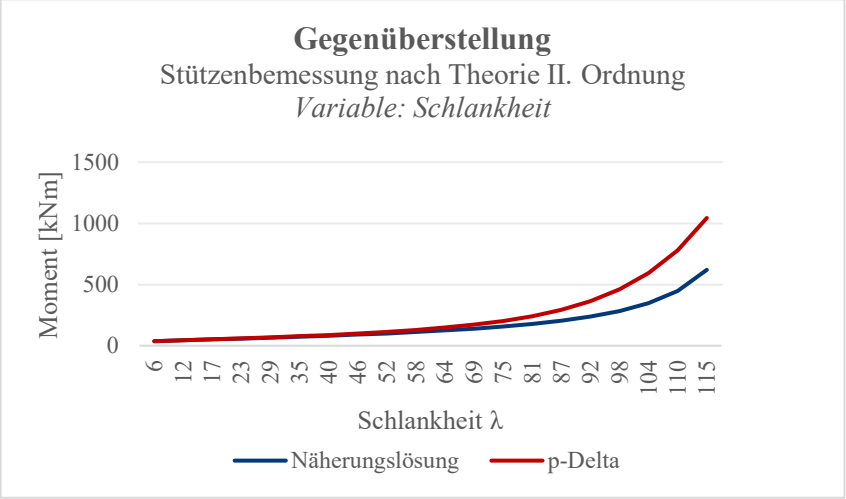


Abbildung 35: Gegenüberstellung Theorie II. Ordnung: p-Delta und Näherungsverfahren

3.6.4 Gegenüberstellung der Bemessungsverfahren

Abschließend lässt sich sagen, dass im Bereich von mittleren Schlankheiten (20> λ < 100) alle drei Bemessungsverfahren vergleichbare Ergebnisse erzielen, wobei die Tragfähigkeiten eines Stabes, welcher nach der Theorie II. Ordnung bemessen wurde 10% bis 20% wirtschaftlicher ausfällt. Der Grund für den Unterschied ist nach Neuhaus auf die Ermittlung des Knickbeiwertes zurückzuführen. Hierbei wird nach der DIN EN 1995-1-1 der bezogene Schlankheitsgrad mit einer charakteristischen Druckfestigkeit berechnet, sodass der Abminderungsbeiwert k_c und resultierend die Tragfähigkeit kleiner ausfällt (2017: 832).

Bei Schlankheitswerten größer als 100 zeigt der Vergleich mit beiden Verfahren, dass bei sehr schlanken Bauteilen die Schnittgrößenermittlung nach der Theorie II. Ordnung von großer Bedeutung ist.

Dateien
Die Parameterstudie ist in der Excel-Datei des Stützenbemessungsprogramms *Stützenbemessung_ec5_63.xlsm* im Reiter „Parameterstudie“ zum eigenen Anwenden bereitgestellt.

3.7 Optimierungsalgorithmus

Besonders leistungsstark ist eine Programmierung, wenn Iterationsprozesse zur Optimierung verwendet werden. Der folgende Algorithmus optimiert die Holzstützenbemessung nach zwei vorgegebenen Parametern, die in der Excel-Oberfläche des Programms definiert werden können.

Mit diesen Einstellungen durchläuft das Skript alle Stützenpositionen und bemisst die Stütze nach Theorie I. und nach Theorie II. Ordnung und passt die Parameter automatisch innerhalb der definierten Grenzen an, bis der Bereich der gewünschten Ausnutzung erreicht wird.

3.7.1 Schematischer Ablauf

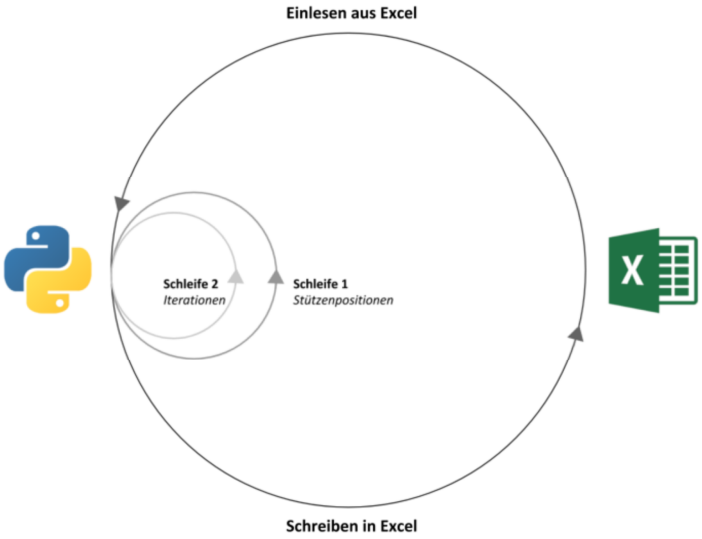


Abbildung 36: Schema zum Optimierungsalgorithmus

Der Algorithmus besteht aus zwei Schleifen, mit denen ein zirkulärer Datenaustausch zwischen dem Code und Excel stattfindet. Dadurch wird nach jedem Iterationsschritt ein Zwischenergebnis in Excel geschrieben und die Daten daraufhin neu eingelesen. Der in der Abbildung dargestellte Ablauf zeigt den schematischen Zusammenhang der Schleifen. Nachdem die Daten anfangs in Python eingelesen wurden, beginnt innerhalb der ersten Schleife die Iteration durch die Stützenpositionen. Mit den eingelesenen Daten der ersten Position erfolgt die Stützenbemessung und die Bewertung der Ergebnisse. Hiermit beginnt die Iteration in der zweiten Schleife. Die Bewertung erfolgt durch fünf if-Abfragen, wodurch zunächst die Holzgüte und schließlich der Stützenquerschnitt erhöht oder verringert wird. Nach jeder Iteration der zweiten Schleife werden die angepassten Parameter und Ausnutzung in Excel als Zwischenergebnis geschrieben und die Bemessung erneut ausgeführt. Erst wenn das Ergebnis innerhalb der definierten Ausnutzungsgrenzen liegt oder die fünfte Iteration erreicht ist, wird Schleife 2 verlassen. Wieder werden die Ergebnisse in Excel übergeben und derselbe Vorgang wird mit der nächsten Stützenposition wiederholt.

3.7.2 Detaillierter Ablauf

1. Verknüpfung mit Excel

Im ersten Teil des Skriptes findet die Verknüpfung mit Excel statt. Wie zuvor wird die Reihenanzahl der Tabelle erfasst und die alten Daten gelöscht. Anschließend werden die Optimierungsparameter und die Listen aller Stützenparameter eingelesen.

2. Sonstige Vorbereitungen

Im zweiten Schritt werden die für die Schleifen benötigten leeren Listen und weitere Parameter wie Statusangaben definiert. Diese werden nach der letzten Bemessungsiteration in Excel geschrieben. Unterschieden wird zwischen:

```
status_1 = 'NW erfüllt'
status_2 = 'NW n. erbracht'
status_3 = 'Grenze'
```

Letzteres wird angegeben, wenn die Grenzen der Parameter erreicht werden und der Nachweis nicht erfüllt wird.

3. Start der Schleife 1

Nachdem die Vorbereitungen getroffen sind und alle notwendigen Informationen eingelesen wurden, kann das Durchlaufen der Stützenpositionen beginnen. Hierzu wird mit einer `for-Schleife` mit der Variablen j durch die Anzahl der Positionen `no_pos` iteriert.

```
for j in range(no_pos):
```

4. Start der Schleife 2

Innerhalb der ersten Schleife wird nun die zweite Schleife betreten und es wird zunächst mittels einer `if-Abfrage` geprüft, ob die Position bereits berechnet wurde. Ist dies beispielsweise der zweite Optimierungsvorgang und der Status der Stütze entspricht `status_2`, wird diese Position übersprungen. Enthält sie jedoch keine Daten, beginnt das Einlesen der Stützenparameter.

Hierbei ist zu unterscheiden, dass die statischen Werte, wie die Länge, Einwirkung etc. anhand des `j`-ten Elementes der Listen eingelesen werden und die sich verändernden Werte, wie die Abmessungen und Holzgüte dynamisch aus Excel eingelesen werden.

5. Stützenbemessung

Nun wird mittels der definierten Funktionen aus Kapitel 3.5 die Stützenbemessung nach Theorie I. und II. Ordnung geführt und der maximale Ausnutzungswert definiert. Anhand dessen, kann nun die Bewertung des Ergebnisses erfolgen.

6. Bewertung der Ausnutzung

Bei der Bewertung werden mit `if-Abfragen` fünf Fälle unterschieden:

- **FALL 1: Iteration fertig / Ausnutzung nicht erreicht**
  Beim ersten Fall ist nach der fünften Iteration die Zielausnutzung nicht erreicht und es wird der `Status 2` ausgegeben.

- **FALL 2: Ausnutzung erreicht**
  Im Fall zwei liegt die Ausnutzung im gewünschten Bereich, sodass alle Ergebnisse in die Excel-Tabelle geschrieben werden kann und die Schleife 2 mit dem `break()`-Befehl verlassen wird.

- **FALL 3: Ausnutzung über 100%**
  Bei den Optimierungsparametern wird zunächst die Anpassung der Holzgüte priorisiert. Erst wenn die Grenze der Güte erreicht wird, beginnt die Anpassung der Querschnittswerte.
  Beim `Fall 3` liegt die Ausnutzung bei über 100%. Die Güte wird um einen Wert erhöht. Für den Querschnitt wird die Abmessung um zwei Werte erhöht.

- **FALL 4: Ausnutzung über `eta_top`**
  Die Ausnutzung liegt über dem Grenzwert der Ausnutzung und die Güte wird um einen Wert erhöht und die Abmessungen um einen Wert erhöht.

- **FALL 5: Ausnutzung unter `eta_bot`**

  Im fünften Fall liegt die Ausnutzung unterhalb des Grenwertes, sodass die Parameter entsprechend um einen Wert verringert werden.

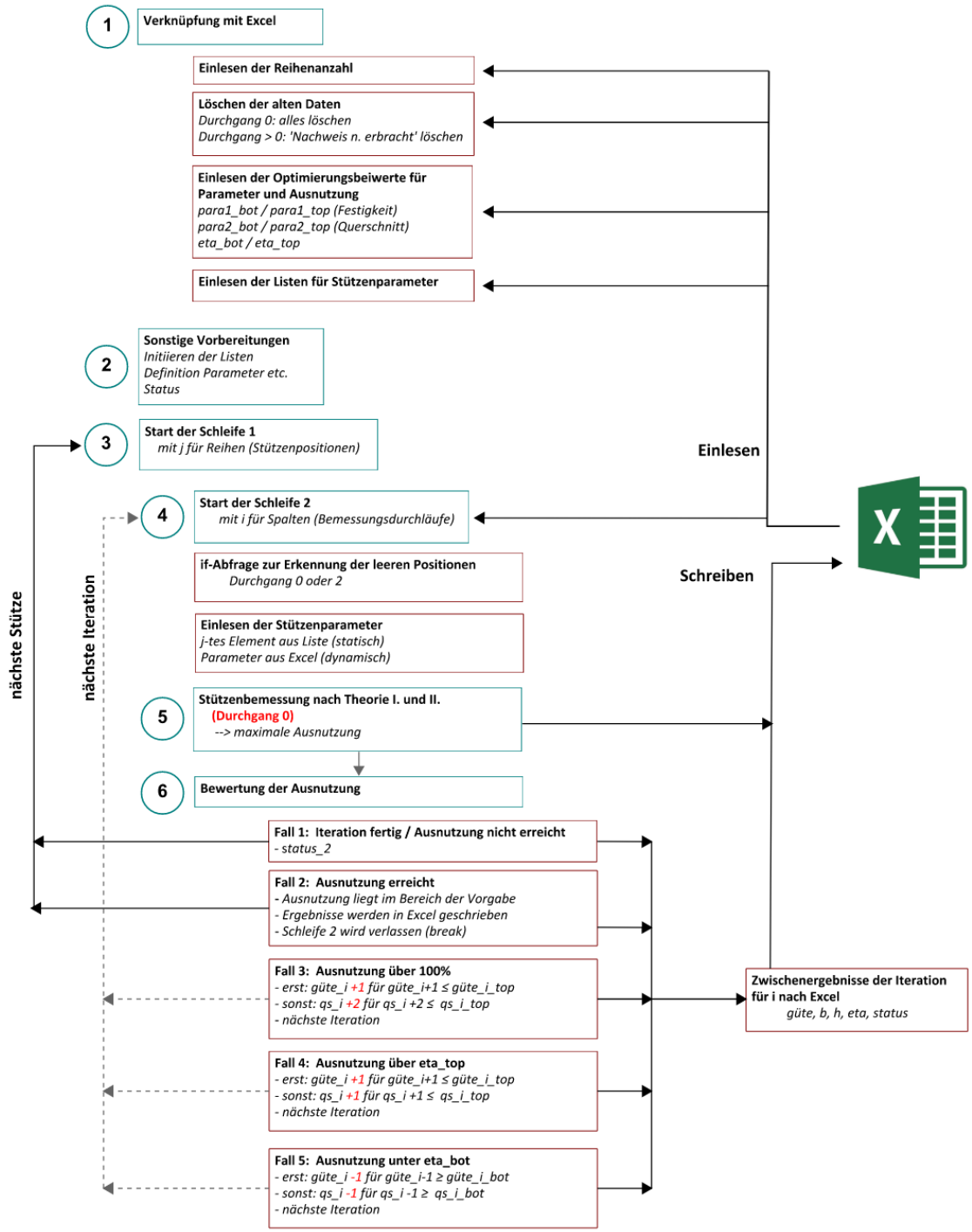


Abbildung 37: Detailliertes Schema zum Optimierungsalgorithmus

Der vollständige Code ist dem Anhang A7 und dem Jupyter-Arbeitsblatt *3.7_Anwendungsbeispiel_1_Optimierungsalgorithmus.ipynb zu* entnehmen.

3.8 Dokumentation

Um die Ergebnisse einer Bemessung verständlich zu dokumentieren, ist es als Ingenieur von großer Bedeutung übersichtliche und nachvollziehbare Grafiken zu erstellen. Im Folgenden wird einerseits demonstriert, wie es mit der Interaktion von Python und AutoCAD möglich ist, eine automatisierte grafische Auswertung zu erzeugen und wie anhand von xlwings schriftliche Einzelnachweise als PDF exportiert werden können.

3.8.1 Grafische Dokumentation

Grafische Übersichten eignen sich auf einer verständlichen und übersichtlichen Art und Weise Informationen zu präsentieren. Grade bei Zwischenergebnissen kann es zu ständigen Veränderungen und Anpassungen kommen, sodass die Erstellung einer grafischen Übersicht eine zeitintensive Aufgabe darstellen kann. Zusätzlich besteht bei manuell auszuführenden Arbeiten das hohe Risiko von Fehlern, die mit einer automatisierten Arbeitsweise minimiert werden können.

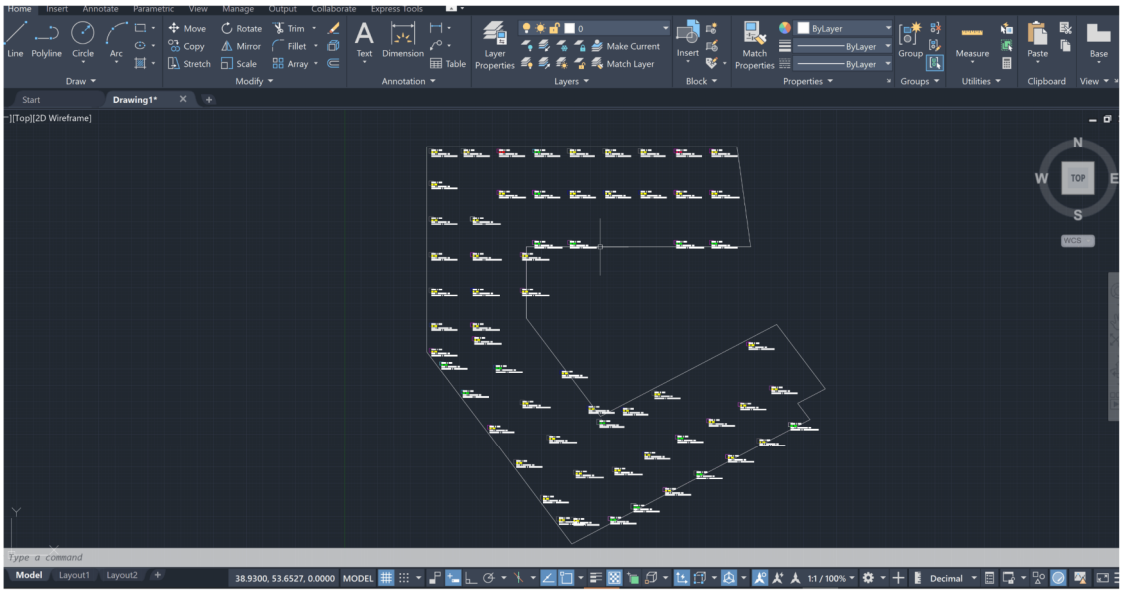


Abbildung 38: Grafische Übersicht in AutoCAD

Link zur Online-Dokumentation:
https://pyautocad.readthedocs.io/en/latest/

3.8.1.1 Bibliothek: pyautocad

Mit der Bibliothek **pyautocad** wird es ermöglicht, mit einfachen Befehlen mit Autodesk AutoCAD zu interagieren und simple geometrische Formen und Textelemente zu überführen.
Die sonstige Vorgehensweise ist analog zu den bisherigen Skripten und wird im Folgenden nur kurz erläutert.

1. Verknüpfung mit Excel
2. Einlesen der Listen
3. Öffnen einer AutoCAD-Datei

```
acad = Autocad(create_if_not_exists=True)
```

4. Zeichnen der Decke Anhand der Schleife durch die Liste der Deckenkoordinaten werden je Durchgang zwei Punkte definiert, die anschließend als Linie verbunden werden.

```
p1 = APoint(xx_coord[i], yy_coord[i])     # Punkt 1
p2 = APoint(xx_coord[i+1], yy_coord[i+1])     # Punkt 2
line1 = acad.model.AddLine(p1, p2)     # Linie zw P. 1 und 2
```

5. Zeichnen der Stützen Gleichermaßen wird durch die Stützenkoordinaten iteriert und gemäß der Bemessungsergebnisse die Koordinaten des Stützenquerschnittes berechnet und gezeichnet. Im selben Durchgang werden die zugehörigen Positionsinformationen in der Liste aufgerufen und als Textelement an die Stütze geschrieben. Um die Querschnitte und die Holzgüte farblich abzusetzen, wird mit dem zuvor ermittelten Index die Farbe gewählt.

```
text = acad.model.AddText(pos[i]+'-'+ geschoss[i], pi, 0.2)
text.Color = L_color_güte[index_güte]
```

Der vollständige Code ist dem Anhang A8 und dem Jupyter-Arbeitsblatt *3.8_Anwendungsbeispiel_1_Dokumentation.ipynb zu* entnehmen.

3.8.2 Schriftliche Dokumentation

Nachdem man mit dem Excel-Programm die Stützen eines ganzen Geschosses berechnet hat, ist es im weiteren Verlauf notwendig, ausformulierte und vom beispielsweise Prüfstatiker nachvollziehbare Nachweise auszudrucken, die man im Bericht verwenden kann. Hierbei ist es nun nicht zweckmäßig das gesamte Geschoss auszudrucken, sodass die Möglichkeit angeboten werden sollte, einzelne Positionen auszuwählen. Dies wird ermöglicht, indem hinter einer Stützenposition ein „x" hingesetzt werden kann, sodass nur die gekennzeichneten Positionen vom Code erfasst werden.

Beispielhaft ist in der Excel-Datei eine Vorlage für die Stützenbemessung nach der Theorie II. Ordnung hinterlegt, die für den PDF-Export verwendet wird. Das Skript funktioniert, indem die Schleife nacheinander die gekennzeichneten Positionen durchläuft und die Bemessungsergebnisse in die Vorlage einfügt und das Arbeitsblatt anschließend als PDF exportiert. Für jede Position wird demnach dieselbe Vorlage verwendet und die Daten für die nächste Position ersetzt. Somit wird gewährleistet, dass auf Knopfdruck ein Nachweis exportiert werden kann, die Datei selbst jedoch „schlank" bleibt.

In der Zelle D41 der Excel-Datei wird ein Ordner für die PDF-Exporte angegeben. Mit dem Befehl `to_pdf` wird die Vorlage exportiert.

```
pfad = ws_1.range("D41").value

pdf_file_name = pfad + "Stützenbemessung_ThII_" + L_pos[j]

ws_2.to_pdf(pdf_file_name)
```

Der vollständige Code und ein Bespiel eines PDF-Ausdrucks sind dem Anhang A9 und dem Jupyter-Arbeitsblatt *3.8_Anwendungsbeispiel_1_Dokumentation.ipynb zu* entnehmen.

4. Anwendungsbeispiel 2: Schraubenbemessung

4.0 Einführung

Im Folgenden soll ein Web-basiertes Bemessungstool für Vollgewindeschrauben entwickelt werden, welches mit der Bibliothek Streamlit erstellt wird.

Abbildung 39: Streamlit Logo

Streamlit ist darauf spezialisiert, schnell und einfach interaktive Webseiten zu programmieren. Besonders gut ist es geeignet, um Ergebnisse auf eine moderne und interaktive Art und Weise darzustellen. Dabei können verschiedene Module, wie *matplotlib*, *plotly* oder *pandas* integriert und grafische Darstellungen und Tabellen eingefügt werden. Außerdem wird eine einfache Methode angeboten, um mit Latex formatierte Formeln darzustellen. In Verbindung mit Github ist es bei Bedarf zusätzlich möglich, die Anwendung als eigenständige Applikation zu exportieren und mit Anderen freizugeben (vgl. Streamlit Inc.: 2022).

Link zum Schraubenbemessungsprogramm:
https://c-02-schraubenbemessung-ec5-87schraubenbemessung-ec5-87-bixfit.streamlitapp.com/

Das Programm wird so konzipiert, dass die Ermittlung der benötigten Schraube in drei Schritten erfolgt.

1) Zunächst können in der Seitenleiste die **Eingangsparameter** hinsichtlich der Verbindung, der Einwirkung und der vorhandenen Bauteildicken eingegeben werden.

Abbildung 40: Schraubenbemessungsprogramm

2) Im zweiten Schritt wird die Tragfähigkeit aller angebotenen Schrauben in Bemessungstabellen und -diagrammen dargestellt. Anhand von interaktiven **Filtern** wird farblich angezeigt, welche Schraube für die angegebene Einwirkung und Bauteilgeometrie zur Bemessung herangezogen werden kann.

L	F_axRk	F_vRk
[mm]	[kN]	[kN]
120	0.56	1.93
140	1.13	2.65
160	1.69	3.09
180	2.26	3.23
200	2.82	3.37
220	2.82	3.37
240	2.82	3.37
260	2.82	3.37
280	2.82	3.37
300	2.82	3.37
330	2.82	3.37

Abbildung 41: Anwendung der interaktiven Filter in Tabelle

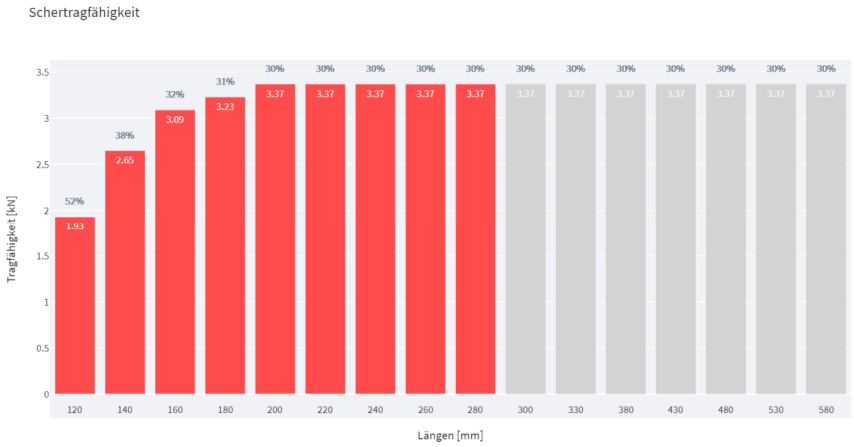


Abbildung 42: Anwendung der interaktiven Filter in Diagramm

3) Ist eine Schraube gewählt worden, kann im letzten Schritt ein **Einzelnachweis** geführt werden. Dieser beinhaltet die ausführliche Berechnung der Tragfähigkeit und kann für die Dokumentation verwendet werden.

Schertragfähigkeit

einschnittige Holz-Holz Verbindung.

DIN EN 1995-1-1 Abs. 8.2.2 (8.6)

$$F_{vk1} = f_{hk} * t_1 * d = 10.92kN$$

$$F_{vk2} = f_{hk} * t_2 * d = 2.18kN$$

$$F_{vk3} = \frac{f_{hk} * t_1 * d}{1+\beta} * \left(\sqrt{1 + 2\beta^2 * (1 + \frac{t_2}{t_1} + \left(\frac{t_2}{t_1}\right)^2) + \beta^3 \left(\frac{t_2}{t_1}\right)^2} - \beta\left(1 + \frac{t_2}{t_1}\right)\right) + \frac{F_{axRk}}{4} = 3.8kN$$

$$F_{vk4} = 1.05 * \frac{f_{hk} * t_1 * d}{2+\beta} * \left(\sqrt{2\beta * (1+\beta) + \frac{4 * \beta * (2+\beta) * M_{yRk}}{f_{hk} * d * t_1^2}} - \beta\right) + \frac{F_{axRk}}{4} = 4.03kN$$

$$F_{vk5} = 1.05 * \frac{f_{hk} * t_2 * d}{1+2\beta} * \left(\sqrt{2\beta^2 * (1+\beta) + \frac{4 * \beta * (1+2\beta) * M_{yRk}}{f_{hk} * d * t_2^2}} - \beta\right) + \frac{F_{axRk}}{4} = 1.31kN$$

$$F_{vk6} = 1.15 * \sqrt{\frac{2\beta}{1+\beta}} * \sqrt{2 * M_{yRk} * f_{hk} * d} + \frac{F_{axRk}}{4} = 1.77kN$$

$$F_{vRk} = min(10.92, 2.18, 3.8, 4.03, 1.31, 1.77) = 1.31kN$$

Abbildung 43: Einzelnachweis mit Latex-Formeln

4.1 Grundlagen zur Bemessung von Holzschrauben

4.1.1 Einleitung

In den folgenden Kapiteln werden verschiedene Anwendungsbeispiele angeführt, die die Tragfähigkeit von Vollgewindeschrauben nach der DIN EN 1995-1-1: 2010-12 Absatz 8 ermitteln. Vollgewindeschrauben gehören mit Stabdübeln, Passbolzen, Bolzen, Gewindestangen und Nägeln zu den metallischen stiftförmigen Verbindungsmitteln. Die Tragfähigkeit wird grundsätzlich hinsichtlich einer Beanspruchung rechtwinklig (Abscheren) und in Richtung (Herausziehen) der Längsachse untersucht. Beim Abscheren wird das Verbindungsmittel auf Biegung beansprucht und überträgt Scherkräfte über Lochleibungsspannungen in das Holz. In Stabachse werden Zugkräfte durch die Verzahnung des Gewindes übertragen (vgl. Neuhaus 2017: 393).

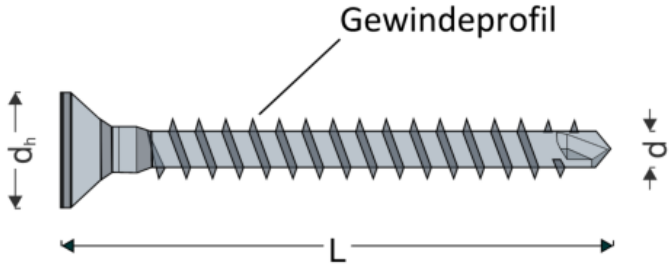


Abbildung 44: Vollgewindeschraube

Holzschrauben müssen den Standards der Produktnormen DIN EN 14592: 2009-02 und 2012-11.1 entsprechen und benötigen bei Abweichungen einen bauaufsichtlichen Verwendbarkeitsnachweis (vgl. Neuhaus 2017: 466). Für die nachfolgenden Betrachtungen wird eine selbstbohrende Vollgewindeschraube der Firma Würth (ASSY plus VG) verwendet, die in der Europäischen Technischen Zulassung ETA-11/0190 zertifiziert ist.

Im Holzbau ist die Wahl und die Art der Verbindungsmittel maßgebend entscheidend für die Bauteilabmessungen. Sie vereinen Tragelemente zu statischen Tragstrukturen und können auf unterschiedlicher Weise ausgeführt werden. Grundsätzlich können Verbindungsmittel nach ihrer Anwendung und nach ihrer Nachgiebigkeit unterschieden

werden. Im Holzingenieurbau kommen meist Verbindungen, wie Stahlformen, und mechanische Verbindungen, wie Nägel und Schrauben zum Einsatz. Die Nachgiebigkeit einer Verbindung entspricht der Steifigkeit bei mechanischer Belastung und hat einen entscheidenden Einfluss auf die globalen Verformungen eines Bauwerks (vgl. Becker/Rautenstrauch 2012: 92).

Hinzu trägt die Konzeptionierung der Holzverbindungen zur Zirkularität eines Gebäudes bei, da bei der Verwendung von einfachen Stahlverbindungen eine Rückbaubarkeit ermöglicht wird, sodass nach dem Lebenszyklus die Materialien in die Kreislaufwirtschaft wieder eingeführt werden können.

4.1.2 Axialtragfähigkeit

Die folgende Abbildung zeigt eine Verbindung eines Balkens mit einem Unterzug und den Einsatz einer Zugschraube. Der Balken lagert auf einer Auflagerfläche und überträgt quer zur Faserrichtung eine Druckkraft. Durch die exzentrische Krafteinleitung entsteht ein Torsionsmoment im Unterzug **(1)**. Dieses Moment löst eine Kippbewegung aus, welches das Auflager des Balkens entziehen würde. Durch die Platzierung der Holzschraube im unteren Zugbereich **(2)**, wird das Torsionsmoment über einen vertikalen Hebelarm in eine Zug- und Druckkomponente abgebaut, sodass eine biegesteife Verbindung gewährleistet wird.

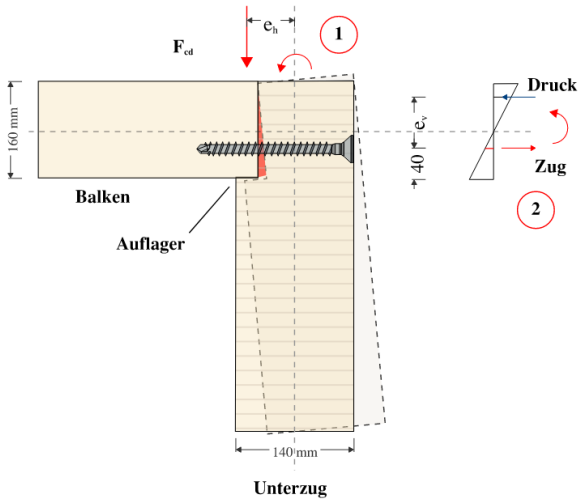


Abbildung 45: Beispiel einer Zugschraube

Die Bemessung des Widerstandes einer Holzschraube in Stabachse nach dem Eurocode 5 Abs. 8.7.2 erfolgt, indem verschiedene Versagensfälle miteinander verglichen werden, wobei der kleinste Wert maßgebend ist (vgl. DIN EN 1995-1-1:2010-12 2010: 81).

1. Herausziehen des Gewindes
2. Zugtragfähigkeit des Stahls
3. Durchziehwiderstand des Schraubenkopfes
4. Abreißwiderstand des Kopfes bei kombinierten Verbindungen mit Stahlblechen

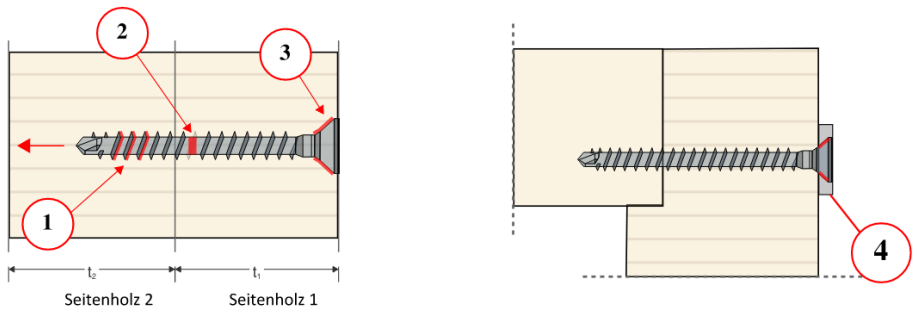


Abbildung 46: Versagensmechanismen von Vollgewindeschrauben

Im Wesentlichen sind folgende Parameter für die Axialtragfähigkeit erforderlich (vgl. Becker/Rautenstrauch 2012: 96):

- charakteristischer Wert des Ausziehparameters f_{axk}
- charakteristischer Wert des Kopfdurchziehparameters f_{headk}
- Winkel zur Faserrichtung α
- Rohdichtekennwert des Baustoffes ρ_k
- Durchmesser des Verbindungsmittels d
- Bauteildicken t_1 und die effektive Länge des Gewindes l_{ef}

Diese Parameter sind teils herstellerspezifisch und müssen der Europäischen Technischen Zulassung ETA-11/0190 für Würth Schrauben entnommen werden (vgl. Deutsches Institut für Bautechnik 2013: 11 ff.).

4.1.2.1 Eingangswerte

Bibliotheken

Zu Beginn werden die Bibliotheken wie bereits erläutert geordnet und alphabetisch sortiert importiert.

Code 29: Importieren der Bibliotheken

```
from math import sqrt, cos, sin, pi    # Mathematische Funktionen
import pandas as pd    # Datenstrukturen und DataFrames
import handcalcs.render    # Darstellung formatierter Gleichungen
```

Kennwerte

Es ist zu beachten, dass sich der Kopfdurchmesser und charakteristische Kopfdurchziehparameter je nach Ausführung der Schraube unterscheiden. Die Kennwerte der Schraube werden zunächst in verschachtelten Listen gespeichert und nach dem Durchmesser sortiert.

Code 30: Listen der herstellerspezifischen Kennwerte für Würth ASSY plus VG

```
# Listen der herstellerspezifischen Kennwerte für Würth ASSY plus VG
## Liste der Durchmesser [mm]
L_d = [6, 8, 10, 12]

## Liste der Schraubenlängen [mm]
L_L = [[120,140,160,180,200,220,240,260],
       [120,140,160,180,200,220,240,260,280,300,330,380,430,480,530,580],
       [120,140,160,180,200,220,240,260,280,300,320,340,360,380,400,430, \
        480,530,580,600,650,700,750,800],
       [120,140,160,180,200,220,240,260,280,300,380,480,600]]

## Liste der char. Ausziehparameter [N/mm^2]
L_f_axk = [11.5, 11, 10, 10]

## Liste der Zugtragfähigkeiten [kN]
L_f_tensk = [11, 20, 32, 45]

## Liste der Kopfdurchmesser [mm]
L_d_h = [14, 22, 25.2, 29.4]

## Liste der char. Kopfdurchziehparameter [N/mm^2]
L_f_head = [13, 13, 10, 10]
```

Kennwerte

Für eine Beispielberechnung werden folgende Eingabewerte definiert:

Code 31: Eingabeparameter

```
# Eingabewerte
d = 6     # mm
L = 240     # mm
t_1 = 100     # mm
t_Blech = 0     # mm
rho_k = 350     # kg/m^3
alpha = 15
```

Mit der Ermittlung des Indexes für den Schraubendurchmesser, also der Position in der Liste, können die entsprechenden Kennwerte aus den Listen entnommen werden. Zusätzlich wird die effektive Eindringtiefe der Schraube berechnet. Die effektive Länge l_{ef} entspricht dem Mindestwert aus der Gewindelänge im zweiten Seitenholz und der Dicke des ersten Seitenholzes $min(L-t_1, t_1)$ (s. Abb. 37).

Code 32: Kennwerte durch Index

```
# Bearbeitung der Eingabwerte
index = L_d.index(d)

d_h = L_d_h[index]
f_headk = L_f_head[index]
f_axk = L_f_axk[index]
f_tensk = L_f_tensk[index]
l_ef = min(L-t_1, t_1)
alph = alpha*pi/180
```

Für eine Zugbeanspruchung der Schraube werden nun die folgenden drei Versagensmechanismen untersucht und die Tragfähigkeit berechnet.

4.1.3 Ausziehwiderstand

In Abschnitt A.1.3.1 der ETA ist der charakteristische Wert der Ausziehtragfähigkeit für Vollholz, Brettschichtholz, Brettsperrholz oder Furnierschichtholz definiert. Der Winkel zur Faserrichtung von 0° ≤ α ≤ 90° wird mit einem Faktor k_{axk} berücksichtigt.

Code 33: Ausziehwiderstand

```
# Ausziehwiderstand
if alpha <= 45:
    k_axk = 0.3+(0.7*alpha)/45
else:
    k_axk = 1

F_axrk1 = (k_axk*f_axk*d*l_ef) * (rho_k/350)**0.8     # N
F_axRk1 = F_axrk1/1000     # kN
```

$$F_{axrk1} = \left(k_{axk} \cdot f_{axk} \cdot d \cdot l_{ef}\right) \cdot \left(\frac{\rho_k}{350}\right)^{0.8} = (0.53 \cdot 11.50 \cdot 6 \cdot 100) \cdot \left(\frac{350}{350}\right)^{0.8} \qquad = 3680.00 \quad (\mathrm{N})$$

$$F_{axRk1} = \frac{F_{axrk1}}{1000} = \frac{3680.00}{1000} \qquad = 3.68 \quad (\mathrm{kN})$$

4.1.4 Kopfdurchziehwiderstand

Der Widerstand der Schraube gegen das Kopfdurchziehen F_{headRk} ist abhängig von der Querdruckspannung, die zwischen Schraubenkopf und Holz auftritt und errechnet sich aus dem Kopfdurchziehparameter f_{headk} und dem Kopfdurchmesser d_h.

Code 34: Kopfdurchziehwiderstand

```
# Kopfdurchziehwiderstand
F_headrk = f_headk*d_h**2*(rho_k/350)**0.8     # N
F_headRk = F_headrk/1000     # kN
```

$$F_{headrk} = f_{headk} \cdot (d_h)^2 \cdot \left(\frac{\rho_k}{350}\right)^{0.8} = 13 \cdot (14)^2 \cdot \left(\frac{350}{350}\right)^{0.8} \qquad = 2548.00 \quad (\mathrm{N})$$

$$F_{headRk} = \frac{F_{headrk}}{1000} = \frac{2548.00}{1000} \qquad = 2.55 \quad (\mathrm{kN})$$

4.1.5 Zugfestigkeit

Die Zugfestigkeit einer Schraube ergibt sich für die entsprechende Festigkeitsklasse aus dem Produkt der Zugfestigkeit f_{ub} und dem Schraubenquerschnitt A.

Code 35: Zugfestigkeit

```
# Zugfestigkeit
F_tRk = f_tensk     # kN
```

$$F_{tRk} = 11 \quad (\mathrm{kN})$$

4.1.6 Resultierende Axialtragfähigkeit

Bei Kombinationen mit Stahlblechen und Unterlegscheiben oder bei einer Seitenholzdicke größer als 4-mal der Schraubendurmesser $t_1 > 4d$ ist das Versagen infolge Kopfdurchziehen zu vernachlässigen.

Code 36: Resultierende Axialtragfähigkeit

```
# Axialtragfähigkeit einer Vollgewindeschraube
## Bei Kombinationen mit Stahlblechen/Unterlegscheiben oder bei t_1 > 4d
ist das Versagen infolge Kopfdurchziehen zu vernachlässigen

if t_Blech != 0 or t_1 > 4*d:
    F_axRk = min(F_axRk1, F_tRk)     # kN
    F_axrk = min(F_axrk1, F_tRk*1000)     # N

elif t_Blech == 0 or t_1 < 4*d:
    F_axRk = min(F_axRk1, F_headRk, F_tRk)     # kN
    F_axrk = min(F_axrk1, F_headrk, F_tRk*1000)     # N
```

4.1.7 Tragfähigkeit auf Abscheren

Rechtwinklig zur Stabachse wird bei der Ermittlung der Tragfähigkeit unterschieden in reine Holz-Holzverbindungen und Stahlblech-Holzverbindungen, die ein- oder mehrschnittig ausgeführt werden können. Die Tragfähigkeit auf Abscheren wird in der Scherfuge nachgewiesen und berücksichtigt die Einflüsse der Lochleibungsfestigkeit, die Fließgrenze und des Ausziehwiderstandes des Verbindungsmittels. Als Berechnungsmodell wird für diese Versagensmechanismen ein ideal-plastisches Verhalten zugrunde gelegt. Diese Vereinfachung beinhaltet, dass elastische Verformungen vernachlässigt werden. Im Eurocode 5 dient die sogenannte Fließgelenktheorie nach Johansen die Grundlage für die Bemessung von stiftförmigen Verbindungsmitteln (vgl. Becker/Rautenstrauch 2012: 96).

Es sind folgende Parameter erforderlich:

- charakteristischer Wert der Lochleibungsfestigkeit f_{hk}
- charakteristisches Fließmoment M_{yRk}
- Winkel zur Faserrichtung α
- Rohdichtekennwert des Baustoffes ρ_k
- Bauteildicken t_1
- Durchmesser des Verbindungsmittels d

In den kommenden Beispielen wird von derselben Holzrohdichte in beiden Seitenhölzern ausgegangen, sodass für die Lochleibungsfestigkeit $f_{hk}=f_{1hk}=f_{2hk}$ gilt. Der Winkel zur Faserrichtung α muss als Bogenmaß umgerechnet werden.

Der Nachweis eines Verbindungsmittels je Scherfuge berücksichtigt verschiedene Versagensmechanismen. Für eine einschnittige Holz-Holzverbindung werden folgende mögliche Mechanismen betrachtet:

Versagen durch Überschreitung der Lochleibungsfestigkeit des Holzes und keine Verformung des Verbindungsmittels:

1) Lochleibung Seitenholz 1
2) Lochleibung Seitenholz 2
3) Lochleibung in beiden Seitenhölzern

Versagen durch Fließgelenkbildung: plastisches Biegeversagen mit plastischer Verformung (bei größeren Holzdicken, höchste Tragfähigkeit und duktiles Bauteilversagen)

4) Plastisches Moment im Seitenholz 2
5) Plastisches Moment im Seitenholz 1
6) Plastisches Moment in beiden Seitenhölzern

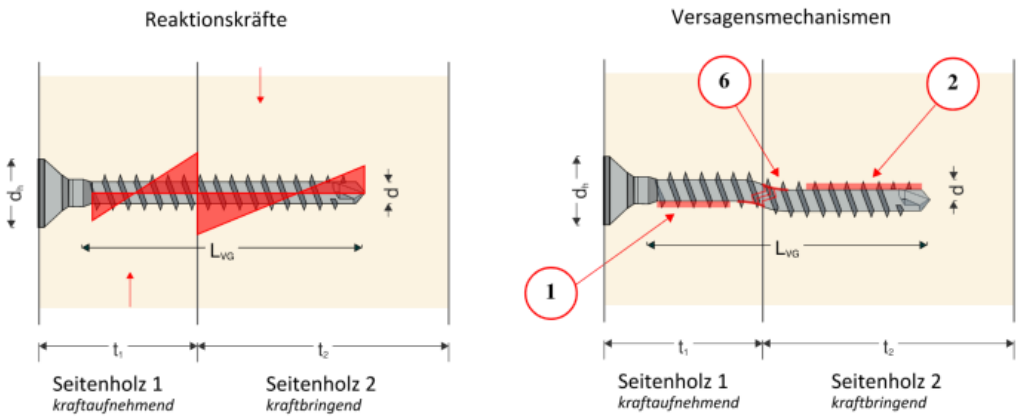


Abbildung 47: Versagensmechanismen von Vollgewindeschrauben auf Abscheren

Winkel zwischen Kraft- und Faserrichtung

Vor allem bei der Betrachtung der Lochleibungsfestigkeit spielt der Winkel zwischen Kraft und Faserrichtung eine bedeutende Rolle, da die Lochleibungsfestigkeit mit

steigendem Winkel abnimmt. Folglich ist sie bei 0 Grad am größten. Dies wird wie folgt berücksichtigt.

```
f_hk = (0.082*rho_k*d**(-0.3))/(2.5*cos(alph)**2+sin(alph)*
*2)
```

Code 37: Kennwerte für die Ermittlung der Schertragfähigkeit

```
# Ermittlung der Kennwerte
f_hk = (0.082*rho_k*d**(-0.3))/(2.5*cos(alph)**2+sin(alph)**2)
M_yrk = 0.15*600*d**(2.6)
```

$$f_{hk} = \frac{0.082 \cdot \rho_k \cdot (d)^{(-0.3)}}{2.5 \cdot (\cos(\text{alph}))^2 + (\sin(\text{alph}))^2} = \frac{0.082 \cdot 350 \cdot (6)^{(-0.3)}}{2.5 \cdot (\cos(0.262))^2 + (\sin(0.262))^2} \qquad = 6.987 \quad (N/mm^2)$$

$$M_{yrk} = 0.15 \cdot 600 \cdot (d)^{2.6} = 0.15 \cdot 600 \cdot (6)^{2.6} \qquad = 9493.71 \quad (Nmm \text{ - Plastisches Moment})$$

4.1.8 Einschnittige Holz-Holzverbindung

4.1.8.1 Lochleibungsversagen

Beim Lochleibungsversagen findet keine Verformung im Holz statt und führt zu einem spröden Bauteilversagen im Holz. Dieses Versagen kann durch eine Steigerung der Bauteildicken vermieden werden.

Code 38: Lochleibungswiderstand 1

```
# Lochleibungswiderstand im Seitenholz 1
F_vk1 = f_hk*t_1*d
```

$$F_{vk1} = f_{hk} \cdot t_1 \cdot d = 6.99 \cdot 100 \cdot 6 \qquad = 4192.41 \quad (N \text{ - Lochleibungsw. im SH1})$$

$$F_{Vk1} = \frac{F_{vk1}}{1000} = \frac{4192.41}{1000} \qquad = 4.19 \quad (\text{kN})$$

Code 39: Lochleibungswiderstand 2

```
# Lochleibungswiderstand im Seitenholz 2
F_vk2 = f_hk*l_ef*d
```

$$F_{vk2} = f_{hk} \cdot l_{ef} \cdot d = 6.99 \cdot 100 \cdot 6 \qquad = 4192.41 \quad (N \text{ - Lochleibungsw. im SH2})$$

$$F_{Vk2} = \frac{F_{vk2}}{1000} = \frac{4192.41}{1000} \qquad = 4.19 \quad (\text{kN})$$

Code 40: Lochleibungswiderstand 3

```
# Lochleibungswiderstand in beiden Seitenhölzern
F_vk3 = (f_hk*t_1*d)/2 * (
sqrt(1+2*(1+l_ef/t_1+(l_ef/t_1)**2)+(l_ef/t_1)**2)-(1+l_ef/t_1))+F_axrk/4
```

$$F_{vk3} = \frac{f_{hk} \cdot t_1 \cdot d}{2} \cdot \left(\sqrt{1 + 2 \cdot \left(1 + \frac{l_{ef}}{t_1} + \left(\frac{l_{ef}}{t_1}\right)^2\right) + \left(\frac{l_{ef}}{t_1}\right)^2} - \left(1 + \frac{l_{ef}}{t_1}\right)\right) + \frac{F_{axrk}}{4}$$

$$= \frac{6.99 \cdot 100 \cdot 6}{2} \cdot \left(\sqrt{1 + 2 \cdot \left(1 + \frac{100}{100} + \left(\frac{100}{100}\right)^2\right) + \left(\frac{100}{100}\right)^2} - \left(1 + \frac{100}{100}\right)\right) + \frac{3680.00}{4}$$

$$= 2656.55 \quad (N \text{ - Lochleibungswiderstand in beiden Seitenhölzern})$$

$$F_{Vk3} = \frac{F_{vk3}}{1000} = \frac{2656.55}{1000} \qquad = 2.66 \quad (\text{kN})$$

4.1.8.2 Entstehung eines Fließgelenks

Bei größeren Holzdicken erfolgt ein plastisches Biegeversagen mit plastischer Verformung. Dieser Versagensmechanismus resultiert in höhere Tragfähigkeiten und einem duktilen Bauteilversagen.

Code 41: Fließgelenk 1

```
# Fließgelenk im Seitenholz 1
F_vk4 = 1.05*(f_hk*t_1*d)/(3) * (sqrt(4+(12*M_yrk)/(f_hk*t_1**2*d))-
1)+F_axrk/4
```

$$F_{vk4} = 1.05 \cdot \frac{f_{hk} \cdot t_1 \cdot d}{3} \cdot \left(\sqrt{4 + \frac{12 \cdot M_{yrk}}{f_{hk} \cdot (t_1)^2 \cdot d}} - 1 \right) + \frac{F_{axrk}}{4}$$

$$= 1.05 \cdot \frac{6.99 \cdot 100 \cdot 6}{3} \cdot \left(\sqrt{4 + \frac{12 \cdot 9493.71}{6.99 \cdot (100)^2 \cdot 6}} - 1 \right) + \frac{3680.00}{4}$$

$$= 2485.39 \quad (N \text{ - Fließgelenk im Seitenholz 1})$$

$$F_{Vk4} = \frac{F_{vk4}}{1000} = \frac{2485.39}{1000} \qquad = 2.49 \quad (\text{kN})$$

Code 42: Fließgelenk 2

```
# Fließgelenk im Seitenholz 2
F_vk5 = 1.05*(f_hk*l_ef*d)/(3) * (sqrt(4+(12*M_yrk)/(f_hk*l_ef**2*d))-
1)+F_axrk/4
```

$$F_{vk5} = 1.05 \cdot \frac{f_{hk} \cdot l_{ef} \cdot d}{3} \cdot \left(\sqrt{4 + \frac{12 \cdot M_{yrk}}{f_{hk} \cdot (l_{ef})^2 \cdot d}} - 1 \right) + \frac{F_{axrk}}{4}$$

$$= 1.05 \cdot \frac{6.99 \cdot 100 \cdot 6}{3} \cdot \left(\sqrt{4 + \frac{12 \cdot 9493.71}{6.99 \cdot (100)^2 \cdot 6}} - 1 \right) + \frac{3680.00}{4}$$

$$= 2485.39 \quad (N \text{ - Fließgelenk im Seitenholz 2})$$

$$F_{Vk5} = \frac{F_{vk5}}{1000} = \frac{2485.39}{1000} \qquad = 2.49 \quad (\text{kN})$$

Code 43: Fließgelenk 3

```
# Fließgelenke in beiden Seitenhölzern
F_vk6 = 1.15*sqrt(2*M_yrk*f_hk*d)+F_axrk/4
```

$$F_{vk6} = 1.15 \cdot \sqrt{2 \cdot M_{yrk} \cdot f_{hk} \cdot d} + \frac{F_{axrk}}{4} = 1.15 \cdot \sqrt{2 \cdot 9493.71 \cdot 6.99 \cdot 6} + \frac{3680.00}{4} \qquad = 1946.04 \quad (N \text{ - beide Seitenhölzer})$$

$$F_{Vk6} = \frac{F_{vk6}}{1000} = \frac{1946.04}{1000} \qquad = 1.95 \quad (\text{kN})$$

4.1.8.3 Resultierende Schertragfähigkeit

Code 44: Resultierende Schertragfähigkeit

```
# Resultierende Schertragfähigkeit
F_vrk = min(F_vk1, F_vk2, F_vk3, F_vk4, F_vk5, F_vk6)
```

$$F_{vrk} = \min(F_{vk1},\ F_{vk2},\ F_{vk3},\ F_{vk4},\ F_{vk5},\ F_{vk6})$$
$$= \min(4192.41,\ 4192.41,\ 2656.55,\ 2485.39,\ 2485.39,\ 1946.04)$$
$$= 1946.04 \quad (\text{N - Resultierende Schertragfähigkeit})$$

$$F_{Vrk} = \frac{F_{vrk}}{1000} = \frac{1946.04}{1000} \qquad = 1.95 \quad (\text{kN})$$

4.1.9 Einschnittige Holz-Stahlverbindung (dickes Blech)

Bei einer Verbindung, bei der die Dicke des Stahlblechs größer als der Schraubendurchmesser ist, wird von einem dicken Stahlblech ausgegangen (t_{Blech}>d). In diesem Fall wird das Verbindungsmittel vom Stahlblech eingespannt.

Code 45: Fließgelenk im Seitenholz

```
# Fließgelenk im Seitenholz
F_vk2 = f_hk*t_1*d*(sqrt(2+(4*M_yrk)/(f_hk*d*t_1**2))-1)+F_axrk/4
```

$$F_{vk1} = f_{hk} \cdot t_1 \cdot d = 6.99 \cdot 100 \cdot 6 \qquad = 4192.41 \quad (N \text{ - Fließgelenk im Seitenholz 1})$$

$$F_{Vk1} = \frac{F_{vk1}}{1000} = \frac{4192.41}{1000} \qquad = 4.19 \quad (\text{kN})$$

Code 46: 2 Fließgelenke

```
# 2 Fließgelenke
F_vk3 = 2.3*sqrt(M_yrk*f_hk*d)+F_axrk/4
```

$$F_{vk2} = f_{hk} \cdot t_1 \cdot d \cdot \left(\sqrt{2 + \frac{4 \cdot M_{yrk}}{f_{hk} \cdot d \cdot (t_1)^2}} - 1 \right) + \frac{F_{axrk}}{4}$$

$$= 6.99 \cdot 100 \cdot 6 \cdot \left(\sqrt{2 + \frac{4 \cdot 9493.71}{6.99 \cdot 6 \cdot (100)^2}} - 1 \right) + \frac{3680.00}{4}$$

$$= 2789.33 \quad (N \text{ - Fließgelenk im Seitenholz 2})$$

$$F_{Vk2} = \frac{F_{vk2}}{1000} = \frac{2789.33}{1000} \qquad = 2.79 \quad (\text{kN})$$

Code 47: Resultierende Schertragfähigkeit

```
# Resultierende Schertragfähigkeit
F_vrk = min(F_vk1, F_vk2, F_vk3)
```

$$F_{vrk} = \min(F_{vk1}, F_{vk2}, F_{vk3}) = \min(4192.41, 2789.33, 2371.03) \qquad = 2371.03 \quad (N \text{ - Resultierende Schertragfähigkeit})$$

$$F_{Vrk} = \frac{F_{vrk}}{1000} = \frac{2371.03}{1000} \qquad = 2.37 \quad (\text{kN})$$

4.1.10 Einschnittige Holz-Stahlverbindung (dünnes Blech)

Ein dünnes Stahlblech ist definiert mit einer Dicke kleiner als die Hälfte des Schraubendurchmessers (t_{Blech}<0,5d).

Code 48: Lochleibungswiderstand im Seitenholz 1

```
# Lochleibungswiderstand im Seitenholz 1
F_vk1 = 0.4*f_hk*t_1*d
```

$$F_{vk1} = 0.4 \cdot f_{hk} \cdot t_1 \cdot d = 0.4 \cdot 6.99 \cdot 100 \cdot 6 \qquad = 1676.96 \ \ (N \text{ - Lochleibungswiderstand im Seitenholz 1})$$

$$F_{Vk1} = \frac{F_{vk1}}{1000} = \frac{1676.96}{1000} \qquad = 1.68 \ \ (\text{kN})$$

Code 49: Fließgelenk im Seitenholz

```
# Fließgelenk im Seitenholz
F_vk2 = 1.15*sqrt(2*M_yrk*f_hk*d)+F_axrk/4
```

$$F_{vk2} = 1.15 \cdot \sqrt{2 \cdot M_{yrk} \cdot f_{hk} \cdot d} + \frac{F_{axrk}}{4} = 1.15 \cdot \sqrt{2 \cdot 9493.71 \cdot 6.99 \cdot 6} + \frac{3680.00}{4} \qquad = 1946.04 \ \ (N \text{ - Fließgelenk im SH})$$

$$F_{Vk2} = \frac{F_{vk2}}{1000} = \frac{1946.04}{1000} \qquad = 1.95 \ \ (\text{kN})$$

Code 50: Resultierende Schertragfähigkeit

```
# Resultierende Schertragfähigkeit
F_vrk = min(F_vk1, F_vk2)
```

$$F_{vrk} = \min(F_{vk1}, F_{vk2}) = \min(1676.96, 1946.04) \qquad = 1676.96 \ \ (N \text{ - Resultierende Schertragfähigkeit})$$

$$F_{Vrk} = \frac{F_{vrk}}{1000} = \frac{1676.96}{1000} \qquad = 1.68 \ \ (\text{kN})$$

4.2 Erstellung einer Funktion

Diese Berechnung wird in den folgenden Kapiteln als Grundlage verwendet, sodass nun eine Funktion erstellt wird, die die Axial- und Schertragfähigkeit einer Vollgewindeschraube berechnet.

```
ec5_87_tragfähigkeit_vg(Hersteller, d, L, t_1, t_2, t_Blech, rho_k
                        , alpha)
```

Als Eingabeparameter werden die folgenden Variablen eingesetzt:

- **Hersteller:** 'Würth' (als Platzhalter für eine mögliche Erweiterung)
- **d:** Durchmesser der Schraube in mm (6,8,10,12)
- **L:** Länge der Schraube in mm
- **t_1:** Dicke des Seitenholzes 1 in mm
- **t_2:** Dicke des Seitenholzes 2 in mm
- **t_Blech:** Dicke des Stahlblechs in mm
- **rho_k:** Rohdichte des Holzes

- **alpha:** Winkel der Kraft zur Faserrichtung in Grad (0-90)

Beim Definieren der Funktion müssen alle Möglichkeiten abgedeckt werden, die beim Ausführen zu Fehlermeldungen führen können. Eine **Fehlermeldung** wird der Liste `nw` angehängt und ausgegeben. Folgende Szenarien müssen beachtet werden:

Es werden nur einschnittige Verbindungen berechnet. Es dürfen nicht alle drei Dicken ungleich 0 sein.

```
t_Blech != 0 and t_1 != 0 and t_2 != 0
```

Es dürfen nicht alle drei Dicken gleich 0 sein.

```
t_Blech == 0 and t_1 == 0 and t_2 == 0
```

Die Schraubenlänge darf nicht gleich der Dicke des Seitenholzes 1 oder des Blechs sein.

```
L in [t_1, t_Blech]
```

Für eine einschnittige Verbindung darf die Dicke des Seitenholzes 2 und des Blechs gleich 0 sein.

```
t_Blech == 0 and t_2 == 0
```

Holz-Holzverbindung

Für eine Holz-Holzverbindung darf die Dicke des Seitenholzes 1 nicht größer als die Schraubenlänge sein.

```
t_Blech == 0 and t_1 > L
```

Für eine Holz-Holzverbindung darf die Summe der Dicken des Seitenholzes 1 und 2 nicht größer als die Schraubenlänge sein.

```
L >= (t_1+t_2) and t_Blech == 0
```

Stahl-Holzverbindung

Für eine Stahl-Holzverbindung darf die Dicke des Seitenholzes 1 nicht gleich null sein.

```
t_Blech != 0 and t_1 == 0 and t_2 != 0
```

Für eine Stahl-Holzverbindung darf die Schraubenlänge nicht größer als die Summe der Dicke des Blechs und des Seitenholzes 1 sein.

```
L >= (t_Blech+t_1) and t_2 == 0
```

Der Code ist dem Anhang A8 und dem Jupyter-Arbeitsblatt *4.2_Anwendungsbeispiel_2_Schraubenbemessung_Funktion.ipynb* zu entnehmen.

4.3 Entwicklung einer Web-App mit Streamlit

4.3.1 Einführung

Nachdem die Grundlagen für Vollgewindeschrauben durchgesprochen und eine Funktion für die Bemessung erstellt wurde, kann nun schrittweise eine Anwendung aufgebaut werden, welche mit der Bibliothek **Streamlit** erstellt wird.

4.3.2 Erste Schritte

Die Installation von Streamlit erfolgt ebenso mit dem Modul-Installierer von Python (pip) und ist in der Entwicklungsumgebung nach dem Importieren sofort verfügbar. Für das Entwickeln einer Web-App wird nun eine Python-Datei (.py) in Spyder erstellt (.py).

Für den Entwicklungsprozess hilft ein zeitgleiches Öffnen der Entwicklungsumgebung und der Applikation. Um die Applikation zu starten, muss im Arbeitsverzeichnis der Befehl `streamlit run datei.py` ausgeführt werden, sodass sich ein Browserfenster mit der Anwendung öffnet.

Sobald in der Python-Datei weitere Elemente und Befehle hinzugefügt werden, werden die Änderungen nach einem Aktualisieren der Seite (Tastenbefehl "r") sofort umgesetzt. Diese Arbeitsweise ermöglicht ein schnelles und intuitives Entwickeln der Anwendung.

Beispiel für die Ausführung im Terminal

```
cd C:\Users\Cal.Mense\Desktop\01_Streamlit

streamlit run streamlit.py
```

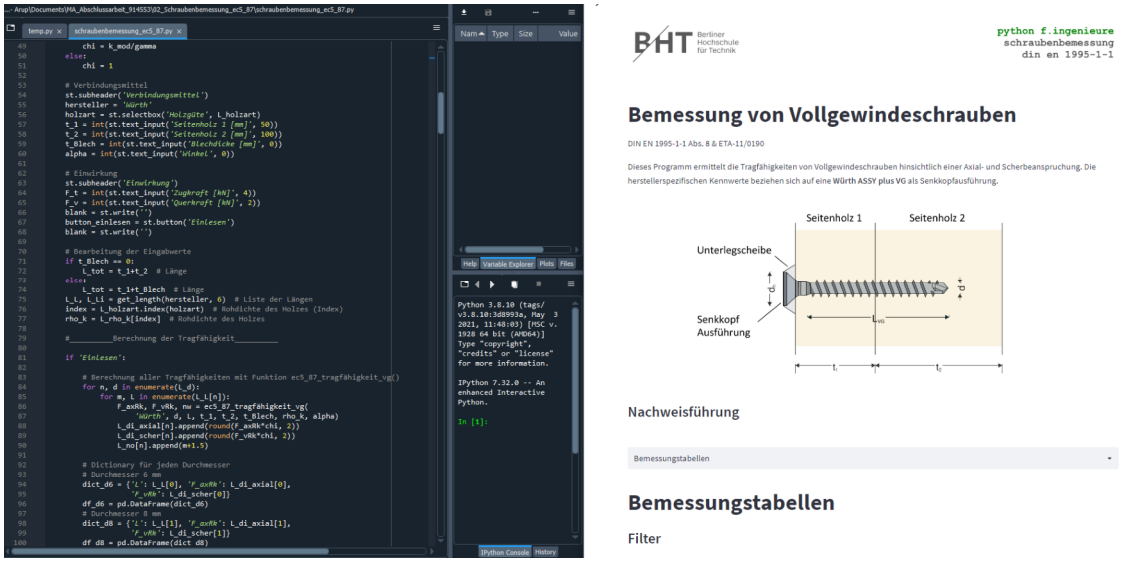


Abbildung 48: Entwicklungsprozess mit Spyder und Streamlit

4.3.3 Elemente in Streamlit

Die Nutzeroberfläche besteht aus einer schlichten Darstellung und kann unterteilt werden in ein Seitenfenster ("sidebar") und einem Hauptfenster. Dort können zeilenweise durch einfache Befehle verschiedene Eingabefelder, Texte, Tasten, Tabellen und Grafiken eingefügt werden.
In der Online-Dokumentation ist eine ausführliche Übersicht aller möglichen Funktionalitäten aufgeführt.

Link zur Online-Dokumentation:
https://docs.streamlit.io/library/api-reference

Es folgt zudem eine Übersicht einiger Elemente, die für das Programm verwendet werden.

4.3.3.1 Textelemente

```
st.title(Einzelnachweis)     # 1

st.subheader(Eingabewerte)     # 2

st.write('Durchmesser [mm]')     # 3
```

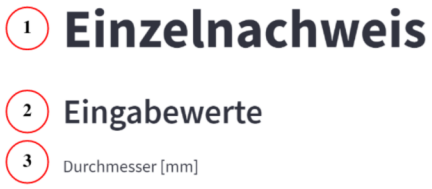


Abbildung 49: Streamlit Textelemente

4.3.3.2 Eingabeelemente

Für die Eingabeelemente wird für das erste Argument die Überschrift und für das zweite Argument die Auswahl angegeben.

```
st.text_input('Seitenholz 1 [mm]',50)
st.radio('Nachweisführung',('Charakteristische Werte',\
'Bemessungswerte'), horizontal=True)
st.selectbox('Nutzungsklasse', [1,2,3])
st.button('Einlesen')
```

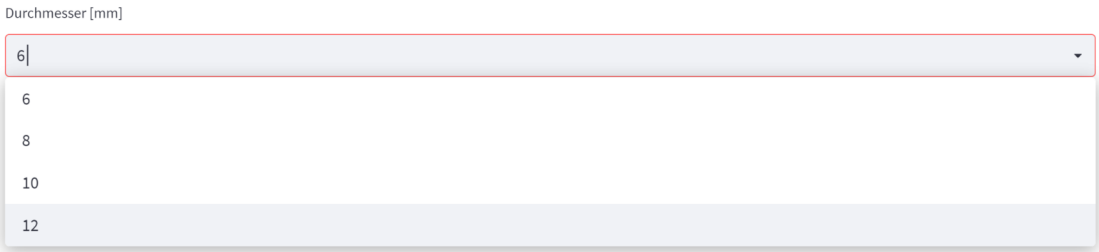


Abbildung 50: Streamlit Eingabeelement "selectbox"

4.3.3.3 Darstellungselemente

Um eine übersichtliche Darstellung zu wahren, können die Inhalte auf verschiedenen Registerseiten verteilt werden.

```
col_d6, col_d8= st.tabs(["6 mm", "8 mm"])
```

Abbildung 51: Streamlit Darstellungselement "tabs"

4.3.4 Bibliothek: Plotly

Plotly gehört zu den beliebtesten Visualisierungs-Bibliotheken in Python und bietet eine große Auswahl an interaktiven Grafiken, die modern und übersichtlich aufbereitet werden können.

Grundsätzlich gibt es dabei zwei Arten, um Grafiken zu erzeugen. Über die sogenannten `graph objects` können Grafiken mit einer großen Vielzahl an Einstellungsmöglichkeiten gestaltet und modifiziert werden. Die zweite Variante bietet mit der `express`-Methode eine schnellere Umsetzung mit weniger Gestaltungsfreiheit und wird für den Einstieg empfohlen.

Im folgenden Link gibt es eine Übersicht verschiedener Grafiken mit zahlreichen Beispielen. Die simplen Beispiele eignen sich für einen einfachen Einstieg und die Adaption für eigene Anwendungen (vgl. Plotly Graphic Libraries 2022).

Link zur Online-Dokumentation:
https://plotly.com/python/

4.3.4.1 Anwendungsbeispiel

Die Nutzung von Tabellen und Diagrammen in Plotly soll im Folgenden anhand der in Kapitel 4.2 erstellten Funktion `ec5_87_tragfähigkeit_vg()` demonstriert werden (s. Anhang A8).

Code 51: Schraubenbemessung - Skript für Tabelle und Grafik

```
# Eingabeparameter
Hersteller = 'Würth'
d = 6    # mm
t_1 = 120    # mm
t_2 = 500    # mm
t_Blech = 0    # mm
rho_k = 350    # kg/m3
alpha = 90    # Grad

# Listen
L_L = [80,100,120,140,160,180,200,220,240,260]
L_axial = []
L_scher = []
L_anmerk = []

# Schleife durch Liste der Längen
for n,L in enumerate(L_L):
  F_axRk, F_vRk, nw = ec5_87_tragfähigkeit_vg(Hersteller,d,L,t_1,t_2,t_Blech, \
                                    rho_k, alpha)

  L_axial.append(F_axRk)
  L_scher.append(F_vRk)
  L_anmerk.append(nw)
```

4.3.4.2 Tabellen

Die allgemeine Vorgehensweise für das Kreieren einer Tabelle in plotly besteht darin, zuerst ein `Figure`-Objekt zu erstellen. Dieses Objekt kann dann beliebig mit der Funktion `add_traces()` mit Graphen erweitert bzw. befüllt werden. Hier wird für das erste Argument `data` eine Liste zugewiesen, die schließlich ein Tabellen-Objekt beinhaltet. Das Tabellen-Objekt besitzt die Argumente für die Überschrift und Zellen der Tabelle, die mit einem Dictionary hinzugefügt werden. Mit der Funktion `update_layout()` können nun alle möglichen grafischen Eigenschaften modifiziert werden.

```
# Initiieren eines Figure-Objektes
tabelle = go.Figure()
# Hinzufügen einer Tabelle
tabelle.add_traces((data=[go.Table(header, cells)]))
# Aktualisieren von grafischen Elementen
```

```
tabelle.update_layout(width, height, title_text)
```

Ausführliche Tutorials sind unter der offiziellen API-Dokumentation zu finden:

Link zur API-Dokumentation:

https://plotly.com/python/table/

Code 52: Plotly-Tabelle

```
# Bibliothek
import plotly.graph_objects as go

# Dictionaries
dict_header={'values':[('L','[mm]'),  ('F_axRk','[kN]'), ('F_vRk','[kN]')]}
dict_cells={'values':[L_L,  L_axial, L_scher]}

# Initiieren eines Figure-Objektes
tabelle = go.Figure()

# Hinzufügen einer Tabelle
tabelle.add_traces([go.Table(header=dict_header, cells=dict_cells)])
```

Tabelle 8: Plotly-Tabelle: Axial- und Schertragfähigkeit

L	F_axRk	F_vRk
[mm]	[kN]	[kN]
80	2.07	1.77
100	2.07	1.77
120	2.07	1.77
140	2.07	1.77
160	2.07	1.77
180	2.07	1.77
200	2.07	1.77
220	2.07	1.77
240	0	0
260	0	0

Modifizieren der Farben

Das Modifizieren der Farben erfolgt getrennt für die Überschriftszeile und der Tabellenzellen, indem jedes Dictionary mit folgenden Schlüsseln und Werten erweitert werden.

```
# Schrift

'font':{'size':15, 'color':'white'}

# Füllfarbe

'fill_color':dark_grey

# Linienfarbe

'line_color':'white'
```

Code 53: Plotly-Tabelle mit Modifizierung

```
# Farben
grey = '#bfced8'
dark_grey = 'lightslategrey'
red = '#ff4b4b'

# Dictionaries
dict_header={'values':[('L','[mm]'),  ('F_axRk','[kN]'), ('F_vRk','[kN]')],
             'align':'center','font':{'size':15, 'color':'white'},
             'fill_color':dark_grey, 'line_color':'white'}

dict_cells={'values':[L_L,  L_axial, L_scher],
            'height':30,'font':{'size':15, 'color':'black'},
            'fill_color':grey}

# Initiieren eines Figure-Objektes
tabelle = go.Figure()

# Hinzufügen einer Tabelle
tabelle.add_traces([go.Table(header=dict_header, cells=dict_cells)])
```

Tabelle 9: Plotly-Tabelle mit Modifizierung

L	F_axRk	F_vRk
[mm]	[kN]	[kN]
80	0	0
100	0	0
120	0	0
140	1.38	1.48
160	2.76	2.28
180	4.14	2.62
200	5.52	2.97
220	6.9	3.31
240	8.28	3.66
260	8.28	3.66

4.3.4.3 Diagramme

Das folgende Balkendiagramm wird auf eine etwas andere Art erstellt. Hierfür wird die `plotly express` Methode angewendet, bei der zunächst die verwendeten Daten in einem Dictionary (oder DataFrame) verpackt werden. Schließlich wird mit dem Befehl `px.bar()` das Diagramm initiiert.

```
# Erstellen eines Dictionaries

dict = {'x':x-Werte, 'y':y-Werte}
```

```
# Hinzufügen einer Tabelle

px.bar(Dictionary, x=x-Werte, y=y-Werte)
```

Weitere Tutorials zu Diagrammen sind unter der offiziellen API-Dokumentation zu finden:

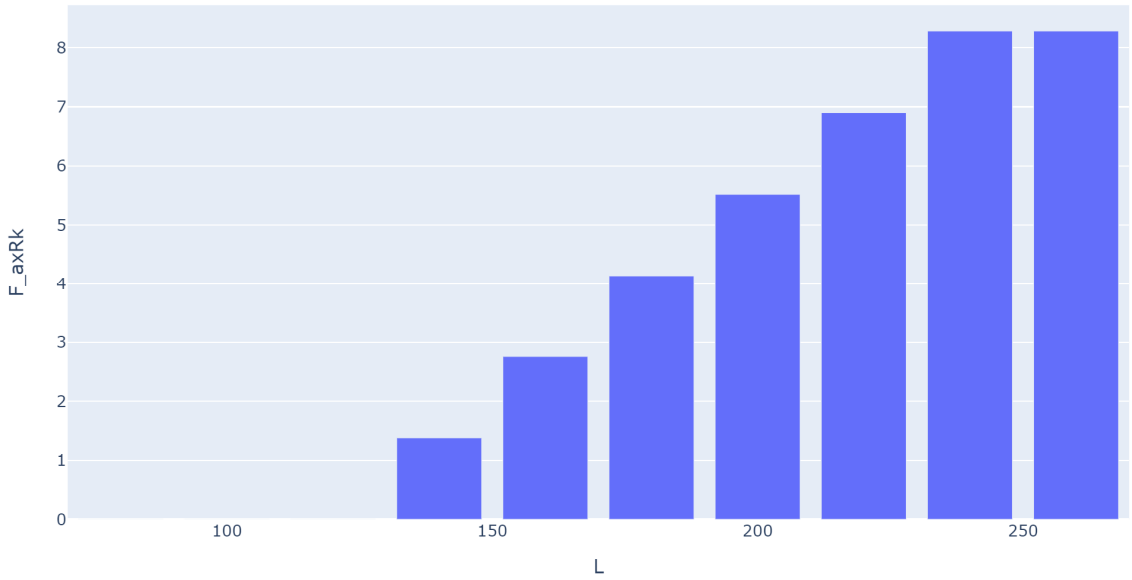
Link zur API-Dokumentation:

https://plotly.com/python/bar-charts/

Code 54: Plotly-Balkendiagramm

```
# Bibliothek
import plotly.express as px

# Erstellen eines Dictionaries
dict_d6 = {'L': L_L, 'F_axRk': L_axial, 'F_vRk': L_scher}

# Hinzufügen einer Tabelle
fig =  px.bar(dict_d6, x='L', y='F_axRk')

# Anzeigen der Tabelle
fig.show()
```

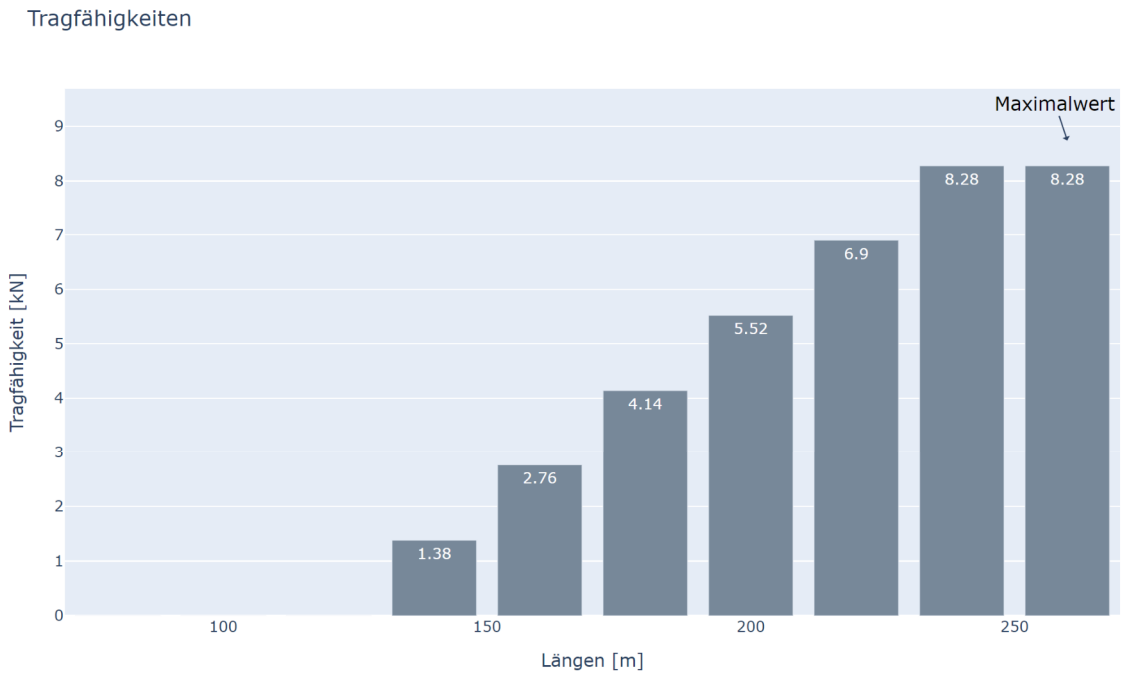
Tabelle 10: Plotly-Balkendiagramm



Code 55: Plotly-Balkendiagramm mit Modifizierung

```
# Bibliothek
import plotly.express as px

# Farben
grey = '#bfced8'
dark_grey = 'lightslategrey'
red = '#ff4b4b'

# Erstellen eines Dictionaries
dict_d6 = {'L': L_L, 'F_axRk': L_axial, 'F_vRk': L_scher}

# Hinzufügen einer Tabelle
fig =  px.bar(dict_d6, x='L', y='F_axRk', text_auto=True,
              title='Tragfähigkeiten', height=600, width=1000)

# Hinzufügen einer Axenbeschriftung
fig.update_yaxes(title_text="Tragfähigkeit [kN]")
fig.update_xaxes(title_text="Längen [m]")

# Hinzufügen eines Textelementes
fig.add_annotation(text='Maximalwert', x=L_L[-1], y=max(L_axial), \
                   arrowhead=1, showarrow=True, yshift=+20, \
                   font={'size':15, 'color':'black'})

# Modifizieren der Farben
fig.update_traces(marker_color=dark_grey, textfont={'color':'white'})

# Anzeigen der Grafik
fig.show()
```

Tabelle 11: Plotly-Balkendiagramm mit Modifizierung



Code 56: Plotly-Tabelle: Axial- und Schertragfähigkeit mit farblicher Absetzung

```
# Einwirkung
F_td = 5 #kN
F_vd = 1 #kN

# Tabelle
L_t_fill2 = []
L_t_fill3 = []

# Farben
light_grey = '#f0f2f6'
grey = '#bfced8'
dark_grey = 'lightslategrey'
red = '#ff4b4b'

# Dictionaries
dict_header={'values':[('L','[mm]'),  ('F_axRk','[kN]'), ('F_vRk','[kN]')],
             'align':'center','font':{'size':15, 'color':'black'},
'fill_color':grey, 'line_color':'white'}

dict_cells={'values':[L_L,  L_axial, L_scher],
            'height':30,'font':{'size':15, 'color':'black'},
'fill_color':[light_grey, L_t_fill2, L_t_fill3]}

# Schleife für Farbgebung
for i in range(len(L_L)):
    if L_axial[i] < F_td:
        dict_cells['fill_color'][1].append(light_grey)
    else:
        dict_cells['fill_color'][1].append(red)

    if L_scher[i] < F_vd:
        dict_cells['fill_color'][2].append(light_grey)
    else:
        dict_cells['fill_color'][2].append(dark_grey)

# Initiieren eines Figure-Objektes
tabelle = go.Figure()

# Hinzufügen einer Tabelle
tabelle.add_traces([go.Table(header=dict_header, cells=dict_cells)])
```

Tabelle 12: Plotly-Tabelle: Axial- und Schertragfähigkeit mit farblicher Absetzung

L	F_axRk	F_vRk
[mm]	[kN]	[kN]
80	0	0
100	0	0
120	0	0
140	1.38	1.48
160	2.76	2.28
180	4.14	2.62
200	5.52	2.97
220	6.9	3.31
240	8.28	3.66
260	8.28	3.66

4.3.5 Erstellung eines Einzelnachweises

Neben der Darstellung der Tragfähigkeiten der Vollgewindeschrauben in Form von Tabellen und Diagrammen soll die Anzeige eines Einzelnachweises ermöglicht werden. Dabei sollen die Berechnungen mit Hilfe der in Streamlit integrierten Funktion `st.latex` in Latex formatiert angezeigt werden (vgl. Streamlit Documentation 2022).

4.3.5.1 Latex

Hierzu werden im Groben die Grundlagen in Latex erläutert.

Tiefstellungen

Grundsätzlich werden mathematische Formeln in Latex innerhalb von Dollarzeichen `$` geschrieben. Tiefstellungen werden mit einem Unterstrich getrennt und werden in geschweiften Klammern definiert.

```
$f_{headk}$
```

4.3.5.2 Mathematische Operatoren

Wird ein Latex-Befehl initiiert, wird ein Rückwärtsschrägstrich verwendet. Darauf können mathematische Operatoren, wie Brüche, Wurzeln oder griechische Symbole folgen.

Brüche

```
$\frac{f_{hk}*t_{1}*d}{2+\beta}$
```

Wurzeln

```
$\sqrt{2*(1+f_{hk})}$
```

Griechische Symbole

```
$\pi$ $\omega$ $\lambda$ $\phi$
```

Weitere Operatoren

```
$\sin(\alpha))$
```

Komplexe Brüche

```
$\left(\sqrt{2\beta*(1+\beta)+\frac{4*\beta*(2+\beta)*M_{yRk}}{f_{hk}*d*t_{1}^2
}}-\beta\right)$
```

4.4 Vollständiger Code

4.4.1 Schematischer Ablauf

Wie bereits in der Kurzbeschreibung erläutert, lässt sich das Programm unterteilen in einzelne Abschnitte. Der vollständige Code besteht aus den einzelnen Elementen, die in den vorherigen Kapiteln besprochen wurden und ist dem Anhang A14 zu entnehmen.

Aufgrund der Länge und Abstraktheit des Codes wird der Ablauf des Algorithmus anhand eines Schemas skizziert.

In den Punkten **1** bis **2** wird mit Streamlit die Seitenleiste konstruiert, die mit verschiedenen Eingabeelementen die Eingangsparameter des Nutzers speichert. Um ein möglichst ruckelfreies Nutzen der Bemessungstabellen zu gewährleisten, wird bereits nach der Eingabe der Parameter die Tragfähigkeiten für alle Längen berechnet und in Dictionaries gespeichert **(3)**. Auf der Hauptseite kann nun zwischen der Anzeige der Bemessungstabellen und des Einzelnachweises gewählt werden **(4)**. Die Bemessungstabellen und -diagramme sind für die Durchmesser in Registerkarten aufgeteilt. Jeder Reiter beinhaltet eine Tabelle und Diagramme für die Axial- und Schertragfähigkeiten der Schrauben **(5)**. Hier besteht die Möglichkeit über die Wahl von Filtern Schrauben auszublenden, die hinsichtlich der Einwirkung und Geometrie unzulässig sind. Für die Anzeige eines Einzelnachweises **(6)** wird ein Durchmesser und eine Länge der Schraube gewählt.

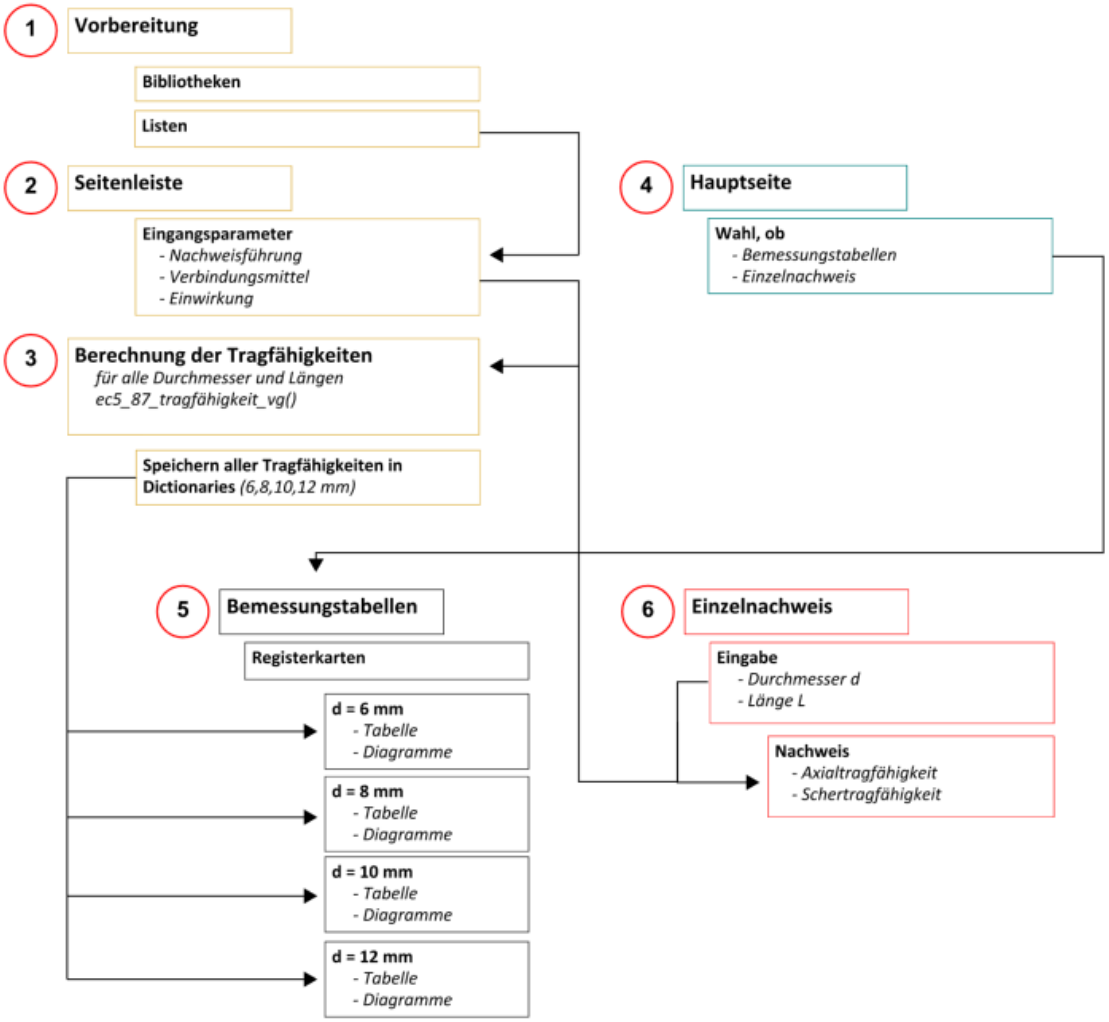


Abbildung 52: Schematischer Ablauf des Codes

5. Fazit

Python ist besonders durch seine Übersichtlichkeit und das einfache Erlernen beliebt. Die Bearbeitung dieser Abschlussarbeit hat diese Eigenschaften Pythons zweifellos bestätigt. Ohne jegliche Vorkenntnisse war es innerhalb der Bearbeitungszeit von fünf Monaten möglich, ein umfangreiches Verständnis im Programmieren mit Python zu erlernen. Dennoch ist zu sagen, dass dies vor allem am Anfang mit einem hohen Maß an Geduld verbunden war und die Fortschritte trotz umfangreicher Dokumentation im Internet nur mit teils langwierigen Recherchen erzielt wurden.

Ähnlich wie in den Ingenieurswissenschaften ist für das Programmieren ein geschultes Auge und Gespür für das Detail notwendig. Komplexe Zusammenhänge werden anhand der Mathematik in logischen Reihenfolgen umgesetzt. Es geht um das Lösen von Problemen, wobei fortwährend das Bestreben nach einer Optimierung und einer Steigerung der Effizienz besteht. Das Programmieren ist ein kognitiv anspruchsvoller und fordernder Prozess, der zeitgleich große Erfolgserlebnisse und Freude herbeiführt. Aus diesem Grund denke ich, dass vor allem die Statik für eine Anwendung und Integration von Programmierungen prädestiniert ist und dass ein Statiker alle nötigen Kompetenzen mitbringt, um eine Programmiersprache zu erlernen.

Die anfänglichen Mühen beim Erlernen von Python resultieren schließlich in immer schnelleren Fortschritten und führen zu einem Verständnis von Grundtechniken und -prinzipien des Programmierens, die immer wieder zum Einsatz kommen und erstaunlich vielseitig einsetzbar sind. Neben einer Handvoll an Methodiken, wie der Einsatz von Schleifen in Verbindung mit if-Abfragen, geht es größtenteils um die Verarbeitung von Datenstrukturen, wie Listen und *Dictionaries*. Ein gutes Verständnis dieser wenigen Fertigkeiten ermöglicht einen sicheren Umgang mit großen Datenmengen, die effizient in einer Anwendung genutzt werden können.

Die verschiedenen Skripte im Anwendungsbeispiel 1 verdeutlichen, dass immer derselbe Aufbau zum Einsatz kommt. Zunächst erfolgt die Verknüpfung mit der Excel-Datei. Anschließend werden Informationen in Listen gespeichert und in allen möglichen Formen weiterverarbeitet. Durch das Iterieren in Schleifenstrukturen ist es beim Stützenbemessungsprogramm möglich, dutzende Positionen in wenigen Sekunden zu berechnen. In Verbindung mit der übersichtlichen Nutzeroberfläche von Excel ist es

einerseits möglich, alle Ergebnisse zu überblicken, andererseits wird dem Nutzer stets die Möglichkeit geboten, auf eine schnelle Art und Weise die Stützenparameter zu ändern. Besonders wichtig ist der Aspekt der Kontrolle. Die Möglichkeit der Eingabeparameter und das Anzeigen der Zwischenergebnisse bietet ein hohes Maß an Kontrolle und Vertrauen beim Nutzer. Das Nutzen von Programmierungen bzw. Automatisierungen sollten niemals das kritische Denken des Ingenieurs ersetzen, sondern sollten nur eine Hilfe darstellen, die die Arbeit vereinfacht.

Beim zweiten Anwendungsbeispiel wurde mit der Bibliothek *Streamlit* ein Web-basiertes Programm entwickelt, welches die Tragfähigkeiten von Vollgewindeschrauben errechnet. Damit wurde eine Alternative zu den herstellerspezifischen Produktblättern aufgezeigt, die die Ergebnisse anhand aller möglichen Eingangsparameter in langen PDF-Dateien darstellen. Die recht unhandliche Nachweisführung von Holzschrauben wurde stattdessen genutzt, um eine Anwendung mit interaktiven Tabellen, Grafiken und Latex-formatierten Einzelnachweisen zu erstellen. *Streamlit* ermöglicht dabei durch beeindruckend einfache Befehle das Entwickeln von ganzen Nutzoberflächen, die in wenigen Stunden erzeugt und als Programm mit Anderen geteilt werden können.

Vor allem im Zeitalter der Digitalisierung und des Klimawandels entwickelt sich die Bauindustrie dahingehend, dass immer präziser und nachhaltiger gebaut werden muss. Die Integration von Python stellt dafür ein perfektes Werkzeug dar, mit dem statische Berechnungen anhand weniger Grundprinzipien um ein Vielfaches effizienter aufgestellt werden können. Der anfängliche Mehraufwand beim Erlernen von Python bringt große Vorteile hinsichtlich der Produktivität, Kosten und Fehlerminimierung mit sich und kann mit den Kompetenzen eines Statikers leicht erlernt werden. Um den aktuellen Anforderungen auch in Zukunft gerecht zu werden, lässt sich abschließend sagen, dass das Anwenden von unterstützenden Programmierungen in der Statik in Deutschland weitreichender gefördert werden sollte.

Abbildungsverzeichnis

Abbildung 1: Ausschnitt aus GitHub 11
Abbildung 2: Verwendete Programme für Anwendungsbeispiele 13
Abbildung 3: Schematischer Ablauf des Stützenbemessungsprogramms mit Python 14
Abbildung 4: Verwendete Programme für Anwendungsbeispiele 15
Abbildung 5: Ausschnitt aus einem Jupyter Notebook 16
Abbildung 6: Funktion zur Ausblendung der Code-Zeilen 16
Abbildung 7: Startseite der Entwicklungsumgebung Anaconda 20
Abbildung 8: Eingabeaufforderung in Windows 21
Abbildung 9: xlwings Register in Excel 23
Abbildung 10: VBA Einstellungen für die xlwings Erweiterung 24
Abbildung 11: Reiter „Lastabtrag“ des Stützenbemessungsprogramms 26
Abbildung 12: Eingabeparameter des Stützenbemessungsprogramms 26
Abbildung 13: Gegenüberstellung der Bemessungsverfahren 27
Abbildung 14: Eingabefelder von Optimierungsparameter 27
Abbildung 15: Optimierungsalgorithmus des Stützenbemessungsprogramms 28
Abbildung 16: PDF-Plot der grafischen Übersicht in AutoCAD 29
Abbildung 17: Ausschnitt des AutoCAD-Plots 29
Abbildung 18: Richtungsabhängigkeiten der Holzfestigkeiten 31
Abbildung 19: Spannungs-Dehnungs-Linie von Holz 32
Abbildung 20: Geometrische Linearität und Nicht-Linearität 33
Abbildung 21: Stützenquerschnitt des Rechenbeispiels 35
Abbildung 22: Euler-Stäbe 40
Abbildung 23: Versagensmechanismus Kippen 43
Abbildung 24: Stützenbemessung nach der Theorie II. Ordnung 49
Abbildung 25: Schema zum Iterationsvorgang 51
Abbildung 26: Zuordnung von Listenwerten im Nachweis 54
Abbildung 27: Versagensmechanismus Kippen 56
Abbildung 28: Schnittgrößen nach Theorie II. Ordnung 57
Abbildung 29: Verformungen nach Theorie II. Ordnung 58
Abbildung 30: Momente nach Theorie II. Ordnung 59
Abbildung 31: Spannungsnachweis nach DIN EN 1995-1-1 Abs. 6.2.4 60
Abbildung 32: Ordnerstruktur 62
Abbildung 33: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (1) 70
Abbildung 34: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (2) 71

Abbildung 35: Gegenüberstellung Theorie II. Ordnung: p-Delta und Näherungsverfahren 72
Abbildung 36: Schema zum Optimierungsalgorithmus 74
Abbildung 37: Detailliertes Schema zum Optimierungsalgorithmus 77
Abbildung 38: Grafische Übersicht in AutoCAD 78
Abbildung 39: Streamlit Logo 81
Abbildung 40: Schraubenbemessungsprogramm 82
Abbildung 41: Anwendung der interaktiven Filter in Tabelle 82
Abbildung 42: Anwendung der interaktiven Filter in Diagramm 83
Abbildung 43: Einzelnachweis mit Latex-Formeln 83
Abbildung 44: Vollgewindeschraube 84
Abbildung 45: Beispiel einer Zugschraube 85
Abbildung 46: Versagensmechanismen von Vollgewindeschrauben 86
Abbildung 47: Versagensmechanismen von Vollgewindeschrauben auf Abscheren 91
Abbildung 48: Entwicklungsprozess mit Spyder und Streamlit 99
Abbildung 49: Streamlit Textelemente 100
Abbildung 50: Streamlit Eingabeelement "selectbox" 100
Abbildung 51: Streamlit Darstellungselement "tabs" 100
Abbildung 52: Schematischer Ablauf des Codes 110

Tabellenverzeichnis

Tabelle 1: Druckfestigkeiten nach der DIN EN 14080 31
Tabelle 2: Theorie I. und II. Ordnung bei der Stützenbemessung 33
Tabelle 3: Mögliche Nachweise bei der Stützenbemessung 44
Tabelle 4: Tabelle mit Pandas DataFrames 66
Tabelle 5: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (1) 70
Tabelle 6: Gegenüberstellung Bemessung nach Theorie I. und II. Ordnung (2) 71
Tabelle 7: Gegenüberstellung Theorie II. Ordnung: p-Delta und Näherungsverfahren 72
Tabelle 8: Plotly-Tabelle: Axial- und Schertragfähigkeit 103
Tabelle 9: Plotly-Tabelle mit Modifizierung 104
Tabelle 10: Plotly-Balkendiagramm 105
Tabelle 11: Plotly-Balkendiagramm mit Modifizierung 106
Tabelle 12: Plotly-Tabelle: Axial- und Schertragfähigkeit mit farblicher Absetzung 107

Code-Verzeichnis

Code 1: Installation der xlwings-Erweiterung für Excel (Add-in) 23
Code 2: Aufruf eines Python-Codes in VBA 24
Code 3: Importieren von Bibliotheken 35
Code 4: Geometrie 36
Code 5: Einwirkung 37
Code 6: Charakteristische Festigkeitswerte 37
Code 7: Charakteristische Steifigkeitswerte 37
Code 8: Widerstandsbeiwerte 38
Code 9: Bemessungswerte der Festigkeit 38
Code 10: Querschnittsparameter 39
Code 11: Spannungen 39
Code 12: Knicklänge 40
Code 13: Schlankheit 42
Code 14: Ersatzstabverfahren – Nachweis 47
Code 15: Iteration für Schnittgrößenermittlung nach Theorie II. Ordnung 55
Code 16: Tabellen mit pandas-DataFrames 56
Code 17: Liniendiagramme mit pandas-DataFrames 58
Code 18: Importieren der Bibliothek xlwings 61
Code 19: Pfade 61
Code 20: Verknüpfung mit Excel 62
Code 21: Direkter Verweis 63
Code 22: Index-Verweis 63
Code 23: Dynamischer Verweis 64
Code 24: Dynamische Anzahl der Tabelleneinträge 64
Code 25: Einlesen von Listen 65
Code 26: Darstellung der Daten in Tabellen mit Pandas DataFrames 65
Code 27: Excel-Iteration: Ersatzstabverfahren – Teil 1 67
Code 28: Excel-Iteration: Ersatzstabverfahren – Teil 2 68
Code 29: Importieren der Bibliotheken 87
Code 30: Listen der herstellerspezifischen Kennwerte für Würth ASSY plus VG 87
Code 31: Eingabeparameter 88
Code 32: Kennwerte durch Index 88
Code 33: Ausziehwiderstand 88
Code 34: Kopfdurchziehwiderstand 89
Code 35: Zugfestigkeit 89

Code 36: Resultierende Axialtragfähigkeit 90
Code 37: Kennwerte für die Ermittlung der Schertragfähigkeit 92
Code 38: Lochleibungswiderstand 1 92
Code 39: Lochleibungswiderstand 2 92
Code 40: Lochleibungswiderstand 3 93
Code 41: Fließgelenk 1 93
Code 42: Fließgelenk 2 93
Code 43: Fließgelenk 3 94
Code 44: Resultierende Schertragfähigkeit 94
Code 45: Fließgelenk im Seitenholz 95
Code 46: 2 Fließgelenke 95
Code 47: Resultierende Schertragfähigkeit 95
Code 48: Lochleibungswiderstand im Seitenholz 1 95
Code 49: Fließgelenk im Seitenholz 96
Code 50: Resultierende Schertragfähigkeit 96
Code 51: Schraubenbemessung - Skript für Tabelle und Grafik 102
Code 52: Plotly-Tabelle 103
Code 53: Plotly-Tabelle mit Modifizierung 104
Code 54: Plotly-Balkendiagramm 105
Code 55: Plotly-Balkendiagramm mit Modifizierung 106
Code 56: Plotly-Tabelle: Axial- und Schertragfähigkeit mit farblicher Absetzung 107

Literaturverzeichnis

- Becker, Klausjürgen / Karl Rautenstrauch (2012): Ingenieurholzbau nach Eurocode 5, Konstruktion, Berechnung, Ausführung, 1. Aufl., Berlin, Deutschland: Ernst & Sohn.
- Bonacina, Michael (2018): Python 3, Programmieren für Einsteiger, 2. Aufl., Landshut, Deutschland: BMU Verlag.
- Deutsches Institut für Bautechnik (2013): Europäische Technische Zulassung ETA-11/0190, Würth Schrauben, Berlin, Deutschland: Beuth-Verlag.
- DIN EN 1995-1-1:2010-12 (2010): Eurocode 5: Bemessung und Konstruktion von Holzbauten - Teil 1-1: Allgemeines-Allgemeines regeln und Regeln für den Hochbau, Berlin, Deutschland: Beuth-Verlag.
- DIN EN 1995-1-1/NA:2010-12 (2010): Nationaler Anhang – National festgelegte Parameter – Eurocode 5: Bemessung und Konstruktion von Holzbauten – Teil 1-1: Allgemeines – Allgemeine Regeln und Regeln für den Hochbau, Berlin, Deutschland: Beuth-Verlag.
- DIN EN 14080:2013 (2013):Holzbauwerke-Brettschichtholz und Balkenschichtholz-Anforderungen; Deutsche Fassung EN 14080:2013, Berlin, Deutschland: Beuth-Verlag.
- Mahringer, Walter (2013): Baustatik, Planen - Berechnen - Begutachten von Tragwerken, 1. Aufl., Haan-Gruiten, Deutschland: Europa Lehrmittel.
- Neuhaus, Helmuth (2017): Ingenieurholzbau, Grundlagen – Bemessung – Nachweise - Beispiele, 4. Aufl., Münster, Deutschland: Springer Vieweg.
- Pilgrim, Mark / Florian Wollenschein (2009): Python 3 Intensivkurs, 1. Aufl., Marktheidenfeld, Deutschland: Springer.
- Plotly Graphic Libraries (2022): [online] https://plotly.com/python/ [abgerufen am 06.09.2022].
- Spyder 5 documentation (2022): Welcome to Spyder’s Documentation, [online] https://docs.spyder-ide.org/current/index.html [abgerufen am 07.09.2022].
- Steinkamp, Veit (2021): Der Python-Kurs für Ingenieure und Naturwissenschaftler, 1. Aufl., Bonn, Deutschland: Rheinwerk Verlag.

- Steyer, Ralph (2017): Programmierung in Python, 1. Aufl., Bodenheim, Deutschland: Springer Vieweg.
- Streamlit Documentation (2022): [online] https://docs.streamlit.io/ [abgerufen am 06.09.2022].
- The Jupyter Notebook (2022): The Jupyter Notebook - Jupyter 6.4.12 documentation, [online] https://jupyter-notebook.readthedocs.io/en/stable/index.html [abgerufen am 07.09.2022].
- xlwings (2022): xlwings - Make Excel Fly!, [online] https://docs.xlwings.org/en/stable/index.html [abgerufen am 07.07.2022].

Anhang

Anhangsverzeichnis

A1. Python-Code: Funktion – Ersatzstabverfahren 3

A2. Python-Code: Funktion – Theorie II. Ordnung 7

A3. Python-Code: Excel – Ersatzstabverfahren 10

A4. Python-Code: Excel – Theorie II. Ordnung 13

A5. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Stützenbemessung Ersatzstabverfahren 16

A6. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Stützenbemessung Theorie II. Ordnung 19

A7. Python-Code: Excel – Optimierungsalgorithmus 22

A8. Python-Code: Dokumentation – AutoCAD 28

A9. Python-Code: Dokumentation – PDF 31

A10. Excel PDF-Ausdruck: Einzelnachweis – Stützenbemessung Theorie II. Ordnung 35

A11. Excel CAD-Ausdruck 39

A12. Python-Code: Funktion – Vollgewindeschrauben 41

A13. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Schraubenbemessung 47

A14. Python-Code: Web-App mit Streamlit 50

A1. Python-Code: Funktion – Ersatzstabverfahren

```python
# Stützenbemessungsprogramm: Funktion - Ersatzstabverfahren
# Bibliothek
from math import sqrt, pi  # Mathematische Funktionen

# Funktion
def ec5_63_esv(Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k, f_myk, f_mzk, E, G_05, k_mod, gamma):

    # Vorbereitung
    # Listen
    L_M = [M_yd, M_zd]
    L_bh = [[b, h], [h, b]]
    L_fmk = [f_myk, f_mzk]
    L_k_crit = [1, 1]
    L_lamb = []
    L_lamb_rel = []
    L_ky = []
    L_kc = []
    L_sigma_md = []
    L_eta = []
    L_nw = []
    L_w = []
    L_I = []

    # Beiwert km
    if M_yd == 0 or M_zd == 0:
        L_km = [[1, 1], [1, 1]]
    else:
        L_km = [[1, 0.7], [0.7, 1]]

    # if-Abfrage zur Bestimmung von beta
    if Lagerung == 'Pendelstütze':
        beta = 1
    elif Lagerung == 'Kragstütze':
        beta = 2
    elif Lagerung in ['Eingespannt (o)', 'Eingespannt (u)']:
        beta = 0.7
    else:
        beta = 0.5

    # Berechnung
    # Knicklänge
    l_ef = L * beta
    e_0 = 0

    # Bemessungswerte der Festigkeit
    xi = k_mod / gamma
    f_c0d = f_c0k * xi  # $N/mm^2$ - Bemessungswert der Druckfestigkeit
    f_myd = f_myk * xi  # $N/mm^2$ - Bemessungswert der Biegefestigkeit
    f_mzd = f_mzk * xi  # $N/mm^2$ - Bemessungswert der Biegefestigkeit

    # Schleife für Momente um y- und z-Achse
    for n, M in enumerate(L_M):

        # Querschnittsparameter
        A = b * h  # $m^2$
        I = (L_bh[n][0] * L_bh[n][1]**3) / 12  # $m^4$ - FTM
        w = (L_bh[n][0] * L_bh[n][1]**2) / 6  # $m^3$ - Widerstandsmoment
```

```
            i = L_bh[n][1] / sqrt(12)  # $m$ - polares Trägheitsmoment
            L_w.append(w)
            L_I.append(I)

            # Knicken
            lamb = (beta * l_ef) / i  # Schlankheitsgrad
            lamb_rel = lamb / pi * sqrt(f_c0k / E)  # bezogene Schlankheit
            k_y = 0.5 * (1 + 0.1 * (lamb_rel - 0.3) + lamb_rel**2)  # Beiwert
            k_c = 1 / (k_y + sqrt(k_y**2 - lamb_rel**2))  # Knickbeiwert

            # Spannungen
            sigma_cd = N_ed / A  # $kN/m^2$ - Druckspannung
            sigma_md = M / w  # $kN/m^2$ - Biegespannung

            # Anhängen an Listen
            L_lamb.append(round(lamb, 2))
            L_lamb_rel.append(round(lamb_rel, 2))
            L_ky.append(round(k_y, 2))
            L_kc.append(round(k_c, 2))
            L_sigma_md.append(round(sigma_md, 2))

        # Kippen um starke Achse
        # Ermittlung des maßgebenden Widerstandsmoments
        index = L_w.index(max(L_w))
        w = L_w[index]
        b = L_bh[index][0]
        h = L_bh[index][1]
        f_mk = L_fmk[index]

        lamb_relm = sqrt(l_ef / (pi * b**2)) * sqrt(f_mk / sqrt(E * G_05))  # bez. Schlankheitsgrad

        # Kippbeiwert
        if lamb_relm <= 0.75:
            k_crit = 1
        elif lamb_relm > 0.75 and lamb_relm < 1.4:
            k_crit = 1.56 - 0.75 * lamb_relm
        elif lamb_relm > 1.4:
            k_crit = 1 / lamb_relm**2

        if index == 0:
            L_k_crit[0] = k_crit
            L_k_crit[1] = 1

        elif index == 1:
            L_k_crit[0] = 1
            L_k_crit[1] = k_crit

        # Listen für Nachweise
        L_Md = [M_yd, M_zd]
        lamb_rel = min(L_lamb_rel)
        L_pot = [[1, 2], [2, 1]]
        k_crit_index = L_w.index(max(L_w))

        # Schleife für Nachweise
        for n, M in enumerate(L_Md):
            # EC5 Abs. 6.2.4: Biegung m/o Druck ohne Knicken ohne Kippen
            if k_crit == 1 and lamb_rel < 0.3:
                eta = (sigma_cd / f_c0d)**2 + \
                    L_km[n][0] * L_sigma_md[0] / f_myd + \
```

```
                    L_km[n][1] * L_sigma_md[1] / f_mzd
                L_eta.append(round(eta, 2))
                nw = ('/')

            # EC5 Abs. 6.3.2: Biegung m/o Druck mit Knicken ohne Kippen
            elif k_crit == 1 and lamb_rel > 0.3:
                eta = (sigma_cd) / (f_c0d * L_kc[n]) + L_km[n][0] * \
                    L_sigma_md[0] / f_myd + L_km[n][1] * L_sigma_md[1] / f_mzd
                L_eta.append(round(eta, 2))
                nw = ('Kn')

            # EC5 Abs. 6.3.3: Biegung m/o Druck mit Knicken und Kippen
            elif k_crit < 1 and lamb_rel < 0.3:
                eta = (sigma_cd) / (f_c0d * L_kc[n]) + (L_sigma_md[0] / (f_myd * L_k_crit[0])
                                                        )**L_pot[n][0] + (L_sigma_md[1] / (f_mzd * L_k_crit[1]))**L_pot[n][1]
                L_eta.append(round(eta, 2))
                nw = ('Kn/Ki')

            else:
                eta = 0
                L_eta.append(round(eta, 2))
                nw = ('N/A')

        return e_0, L_lamb, L_lamb_rel, L_ky, L_kc, k_crit, sigma_cd, L_sigma_md, L_eta, nw
```

A2. Python-Code: Funktion – Theorie II. Ordnung

```python
# Stützenbemessungsprogramm: Funktion - Theorie II. Ordnung
# Bibliotheken
from math import sqrt  # Mathematische Funktionen

# Funktion
def ec5_63_th2(Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k, f_myk, f_mzk, E, k_mod, gamma, no_iter):

    # Listen
    L_eta = []

    # Bemessungswerte der Festigkeit
    xi = k_mod/gamma
    f_c0d = f_c0k*xi  # $N/mm^2$ - Bemessungswert der Druckfestigkeit
    f_myd = f_myk*xi  # $N/mm^2$ - Bemessungswert der Biegefestigkeit
    f_mzd = f_mzk*xi  # $N/mm^2$ - Bemessungswert der Biegefestigkeit

    # Beiwert k_m
    if M_yd == 0 or M_zd == 0:
        L_km = [[1, 1], [1, 1]]
    else:
        L_km = [[1, 0.7], [0.7, 1]]

    # if-Abfrage zur Bestimmung von beta
    if Lagerung == 'Pendelstütze':
        beta = 1
    elif Lagerung == 'Kragstütze':
        beta = 2
    elif Lagerung == 'Eingespannt (u/o)':
        beta = 0.7
    else:
        beta = 0.5

    # Ausmitte nach Theorie I. Ordnung
    l_ef = L*beta
    e_0 = round(theta*l_ef, 8)
    M_0 = e_0 * N_ed

    # Listen
    L_bh = [[b, h], [h, b], [b, h], [h, b]]
    L_e = [[e_0*1000], [0], [0], [e_0*1000]]
    L_M = [[M_yd+M_0], [M_zd], [M_yd], [M_zd+M_0]]
    L_e_total = [[e_0*1000], [0], [0], [e_0*1000]]
    L_M_total = [[M_yd+M_0], [M_zd], [M_yd], [M_zd+M_0]]
    L_sigma_mIId = []

    # Schleife 1: Durchlaufen von [M_yIId_imp, M_zIId, M_yIId, M_zIId_imp ]
    for n in range(4):

        # Querschnittsparameter
        A = b*h  # $m^2$
        I = (L_bh[n][0]*L_bh[n][1]**3)/12  # $m^4$ - FTM
        w = (L_bh[n][0]*L_bh[n][1]**2)/6  # $m^3$ - Widerstandsmoment
        i = L_bh[n][1]/sqrt(12)  # $m$ - polares Trägheitsmoment

        # Schleife 2: Schnittgrößenermittlung nach Theorie II. Ordnung
        for i in range(no_iter):

            # Ermittlung der Werte
            #e_i = (L_M[n][i]*l_ef**2)/(E*I*pi**2)
            e_i = (40*L_M[n][i]*L**2)/(E*I*384)
```

```
                M_i = e_i * N_ed

                # Anhängen der Werte in Listen
                L_e[n].append(round(e_i*1000, 1))
                L_M[n].append(M_i)
                e_total = sum(L_e[n])
                M_total = sum(L_M[n])
                L_e_total[n].append(round(e_total, 2))
                L_M_total[n].append(round(M_total, 2))

            # Spannungen
            sigma_mIId = L_M_total[n][-1]/w  # $kN/m^2$
            L_sigma_mIId.append(round(sigma_mIId, 2))

        # Ergebnisse der Schnittgrößenermittlung nach Theorie II. Ordnung
        # Verformungen und Momente
        L_e = [[L_e[0], L_e[1]], [L_e[2], L_e[3]]]
        L_M = [[L_M[0], L_M[1]], [L_M[2], L_M[3]]]
        L_e_total = [[L_e_total[0], L_e_total[1]], [L_e_total[2], L_e_total[3]]]
        L_M_total = [[L_M_total[0], L_M_total[1]], [L_M_total[2], L_M_total[3]]]

        # Spannungen
        sigma_cd = N_ed/A  # $kN/m^2$ - Druckspannung
        L_sigma_mIId = [[L_sigma_mIId[0], L_sigma_mIId[1]],
                        [L_sigma_mIId[2], L_sigma_mIId[3]]]

        # Schleife für y- und z-Achse
        for i in range(2):

            # Nachweis
            eta = (sigma_cd/f_c0d)**2 + \
                L_km[i][0]*L_sigma_mIId[i][0]/f_myd + \
                L_km[i][1]*L_sigma_mIId[i][1]/f_mzd
            L_eta.append(round(eta, 6))

        # maximale Ausnutzung
        eta_max = max(L_eta)
        index = L_eta.index(eta_max)

        return L_e[index], L_M[index], L_e_total[index], L_M_total[index], sigma_cd, L_sigma_mIId[index], eta_max, index
```

A3. Python-Code: Excel – Ersatzstabverfahren

```python
# Stützenbemessungsprogramm: Excel-Iteration - Ersatzstabverfahren
# Bibliothek
import xlwings as xw  # Interaktion mit Excel
from python_code_funktion_esv import ec5_63_esv

# Teil 1: Vorbereitung
# Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws = wb.sheets['Stützenbemessung']

# Dynamisches Einlesen der Zeilenanzahl
startrow = 10  # Startzeile
# Dynamische Anzahl an Reihen
rownum = ws.range('G10').current_region.last_cell.row
ws.range('X10:AH100').value = ""  # Alte Daten LÃ¶schen

# Einlesen der Listen
L_pos = ws.range((startrow, 7), (rownum, 7)).value
L_ges = ws.range((startrow, 8), (rownum, 8)).value
L_system = ws.range((startrow, 9), (rownum, 9)).value
L_L = ws.range((startrow, 10), (rownum, 10)).value
L_b = ws.range((startrow, 11), (rownum, 11)).value
L_h = ws.range((startrow, 12), (rownum, 12)).value
L_N_ed = ws.range((startrow, 13), (rownum, 13)).value
L_M_yd = ws.range((startrow, 14), (rownum, 14)).value
L_M_zd = ws.range((startrow, 15), (rownum, 15)).value
L_h_art = ws.range((startrow, 16), (rownum, 15)).value
L_f_c0k = ws.range((startrow, 17), (rownum, 17)).value
L_f_myk = ws.range((startrow, 18), (rownum, 18)).value
L_E0mean = ws.range((startrow, 19), (rownum, 19)).value
L_E_05 = ws.range((startrow, 20), (rownum, 20)).value
L_G_05 = ws.range((startrow, 21), (rownum, 21)).value
L_k_mod = ws.range((startrow, 22), (rownum, 22)).value
L_gamma = ws.range((startrow, 23), (rownum, 23)).value

# Teil 2: Iteration durch StÃ¼tzenpositionen
for i in range(rownum - startrow + 1):

    # Definition der Variablen des i-ten Elements der Listen
    L = L_L[i]
    b = L_b[i]
    h = L_h[i]
    güte = L_h_art[i]
    Lagerung = L_system[i]
    f_c0k = L_f_c0k[i]*1000
    f_myk = L_f_myk[i]*1000
    f_mzk = f_myk
    E0_mean = L_E0mean[i]*1000
    E_05 = L_E_05[i]*1000
    G_05 = L_G_05[i]*1000
    k_mod = L_k_mod[i]
    gamma = L_gamma[i]
    N_ed = L_N_ed[i]
    M_yd = L_M_yd[i]
    M_zd = L_M_zd[i]

    # Berechnung
    # Kennwerte
    theta = 0
    shi = k_mod/gamma
    f_c0d = f_c0k*shi
    f_myd = f_myk*shi
```

```
        E = E_05

        # StÃ¼tzenbemessung nach Ersatzstabverfahren (Funktion)
        e_0, L_lamb, L_lamb_rel, L_ky, L_kc, k_crit, sigma_cd,
        L_sigma_md, L_eta, nw = ec5_63_esv(
            Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k,
            f_myk, f_mzk, E, G_05, k_mod, gamma)

        # Auswertung der Ergebnisse
        lamb = max(L_lamb)
        index = L_lamb.index(lamb)
        lamb_rel = max(L_lamb_rel)
        kc = min(L_kc)
        sigma_myd = L_sigma_md[0]
        sigma_mzd = L_sigma_md[1]
        eta = max(L_eta)

        # Schreiben in Excel
        ws.range('Y10').offset(i, 0).value = round(lamb, 2)
        ws.range('Z10').offset(i, 0).value = round(lamb_rel, 2)
        ws.range('AA10').offset(i, 0).value = round(kc, 2)
        ws.range('AB10').offset(i, 0).value = round(k_crit, 2)
        ws.range('AC10').offset(i, 0).value = nw
        ws.range('AD10').offset(i, 0).value = sigma_cd
        ws.range('AE10').offset(i, 0).value = round(sigma_myd, 2)
        ws.range('AF10').offset(i, 0).value = round(sigma_mzd, 2)
        ws.range('AG10').offset(i, 0).value = round(eta, 2)
```

A4. Python-Code: Excel – Theorie II. Ordnung

```python
# Stützenbemessungsprogramm: Excel-Iteration - Theorie II. Ordnung
# Bibliothek
import xlwings as xw  # Interaktion mit Excel
from python_code_funktion_th2 import ec5_63_th2

# Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws = wb.sheets['Stützenbemessung']

# Dynamisches Einlesen der Zeilenanzahl
startrow = 10
rownum = ws.range('G10').current_region.last_cell.row
colnum = ws.range((startrow, 1)).current_region.last_cell.column

# Alte Daten löschen
ws.range('AI10:AQ100').value = ""

# Einlesen der Listen
L_pos = ws.range((startrow, 7), (rownum, 7)).value
L_ges = ws.range((startrow, 8), (rownum, 8)).value
L_system = ws.range((startrow, 9), (rownum, 9)).value
L_L = ws.range((startrow, 10), (rownum, 10)).value
L_b = ws.range((startrow, 11), (rownum, 11)).value
L_h = ws.range((startrow, 12), (rownum, 12)).value
L_N_ed = ws.range((startrow, 13), (rownum, 13)).value
L_M_yd = ws.range((startrow, 14), (rownum, 14)).value
L_M_zd = ws.range((startrow, 15), (rownum, 15)).value
L_h_art = ws.range((startrow, 16), (rownum, 15)).value
L_f_c0k = ws.range((startrow, 17), (rownum, 17)).value
L_f_myk = ws.range((startrow, 18), (rownum, 18)).value
L_E0mean = ws.range((startrow, 19), (rownum, 19)).value
L_E_05 = ws.range((startrow, 20), (rownum, 20)).value
L_G_05 = ws.range((startrow, 21), (rownum, 21)).value
L_k_mod = ws.range((startrow, 22), (rownum, 22)).value
L_gamma = ws.range((startrow, 23), (rownum, 23)).value

e_0 = ws.range('D24').value

for j in range(rownum - startrow + 1):

    # Definition der Variablen des j-ten Elements der Listen
    L = L_L[j]
    b = L_b[j]
    h = L_h[j]
    f_c0k = L_f_c0k[j]*1000
    f_myk = L_f_myk[j]*1000
    f_mzk = f_myk
    E0_mean = L_E0mean[j]*1000
    E_05 = L_E_05[j]*1000
    k_mod = L_k_mod[j]
    gamma = L_gamma[j]
    N_ed = L_N_ed[j]
    M_yd = L_M_yd[j]
    M_zd = L_M_zd[j]

    güte = L_h_art[j]
    Lagerung = L_system[j]
    no_iter = 5
    E = E0_mean/1.3

    # Kennwerte
    shi = k_mod/gamma
```

```
        f_c0d = f_c0k*shi
        f_myd = f_myk*shi

        A = b*h
        I = b*h**3/12
        w = (b*h**2)/6
        i = (I/A)**0.5

        # Stützenbemessung nach Theorie I. Ordnung (Funktion)
        L_e, L_M, L_e_total, L_M_total, sigma_cd, L_sigma_mIId, eta, index = ec5_63_th2(
            Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, e_0, f_c0k, f_myk, f_mzk, E, k_mod, gamma, no_iter)

        #Schreiben in Excel
        ws.range('AI10').offset(j, 0).value = L_e_total[0][0]
        ws.range('AJ10').offset(j, 0).value = L_M_total[0][0]
        ws.range('AK10').offset(j, 0).value = L_e_total[0][-1]
        ws.range('AL10').offset(j, 0).value = L_M_total[0][-1]
        ws.range('AM10').offset(j, 0).value = L_e[1][0]
        ws.range('AN10').offset(j, 0).value = L_M[1][0]
        ws.range('AO10').offset(j, 0).value = L_e_total[1][-1]
        ws.range('AP10').offset(j, 0).value = L_M_total[1][-1]
        ws.range('AQ10').offset(j, 0).value = eta
```

A5. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Stützenbemessung Ersatzstabverfahren

A6. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Stützenbemessung Theorie II. Ordnung

A7. Python-Code: Excel – Optimierungsalgorithmus

```python
# Stützenbemessungsprogramm: Optimierungsalgorithmus
# Bibliotheken
import xlwings as xw  # Interaktion mit Excel
from python_code_funktion_esv import ec5_63_esv
from python_code_funktion_th2 import ec5_63_th2

# Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws = wb.sheets['Stützenbemessung']

# Startreihe
startrow = 10
rownum = ws.range('G10').current_region.last_cell.row
colnum = ws.range((startrow, 1)).current_region.last_cell.column
n = rownum-startrow

# Alte Daten löschen
for i in range(n-1):
    if ws.range('BH10').offset(i, 0).value == 'NW n. erbracht':
        ws.range((startrow+i, 45), (startrow+i, 61)).value = ''
    else:
        print('')

# Status
ws.range('BH8').value = 'läuft..'

# Optimierung
# Ausnutzung
eta_target_top = ws.range('D37').value
eta_target_bot = ws.range('D38').value

# Querschnitt
para1_target_top = ws.range('D32').value
para1_target_bot = ws.range('D33').value
para1_steps = ws.range('D34').value
n_steps = (para1_target_top-para1_target_bot)/para1_steps

# Güte
para2_target_top = ws.range('D28').value
para2_target_bot = ws.range('D29').value

L_güte = ['GL 24h', 'GL 24c', 'GL 28h', 'GL 28c', 'GL 32h', 'GL 32c']
index_top = L_güte.index(para2_target_top)
index_bot = L_güte.index(para2_target_bot)

# Einlesen der Listen
L_pos = ws.range((startrow, 7), (rownum, 7)).value
L_ges = ws.range((startrow, 8), (rownum, 8)).value
L_system = ws.range((startrow, 9), (rownum, 9)).value
L_L = ws.range((startrow, 10), (rownum, 10)).value
L_b = ws.range((startrow, 11), (rownum, 11)).value
L_h = ws.range((startrow, 12), (rownum, 12)).value
L_N_ed = ws.range((startrow, 13), (rownum, 13)).value
L_M_yd = ws.range((startrow, 14), (rownum, 14)).value
L_M_zd = ws.range((startrow, 15), (rownum, 15)).value
L_h_art = ws.range((startrow, 16), (rownum, 16)).value
L_f_c0k = ws.range((startrow, 17), (rownum, 17)).value
L_f_myk = ws.range((startrow, 18), (rownum, 18)).value
L_E0mean = ws.range((startrow, 19), (rownum, 19)).value
L_E_05 = ws.range((startrow, 20), (rownum, 20)).value
L_G_05 = ws.range((startrow, 21), (rownum, 21)).value
L_k_mod = ws.range((startrow, 22), (rownum, 22)).value
```

```
L_gamma = ws.range((startrow, 23), (rownum, 23)).value

a1 = 'NW erfüllt'
a2 = 'NW n. erbracht'
a3 = 'Grenze'
L_n_iter = []
L_n_erf = []
L_n_nerf = []

# Starten der Schleife
# j für Reihen
for j in range(rownum - startrow + 1):
    # for j in range(30):
    ws.range('BH9').value = str(j) + '/' + str(n)
    güte = L_h_art[j]

    # i für Spalten
    for k, i in enumerate(range(5)):

        # Rechnet nur die leeren Positionen
        if ws.range('BH10').offset(j, 0).value == a1:
            break

        elif ws.range('BG10').offset(j, 0).value == None or ws.range('BH10').offset(j, 0).value == a2:

            # Definition der Variablen des i-ten Elements der Listen
            L = L_L[j]
            Lagerung = L_system[j]
            k_mod = L_k_mod[j]
            gamma = L_gamma[j]
            N_ed = L_N_ed[j]
            M_yd = L_M_yd[j]
            M_zd = L_M_zd[j]

            b = ws.range('K10').offset(j, 0).value
            h = ws.range('L10').offset(j, 0).value

            güte = ws.range('P10').offset(j, 0).value
            index = L_güte.index(güte)
            f_c0k = ws.range('Q10').offset(j, 0).value*1000
            f_myk = ws.range('R10').offset(j, 0).value*1000
            f_mzk = f_myk
            E0_mean = ws.range('S10').offset(j, 0).value*1000
            E_05 = ws.range('T10').offset(j, 0).value*1000
            G_05 = ws.range('U10').offset(j, 0).value*1000

            # Theorie I. Ordnung
            theta = 0
            E = E_05

            # Funktion
            e_0, L_lamb, L_lamb_rel, L_ky, L_kc, k_crit, sigma_cd, L_sigma_md, L_eta, nw = ec5_63_esv(
                Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k, f_myk, f_mzk, E, G_05, k_mod, gamma)

            # Auswertung der Ergebnisse
            lamb = max(L_lamb)
            lamb_rel = max(L_lamb_rel)
            kc = min(L_kc)
            sigma_myd = L_sigma_md[0]
```

```
            sigma_mzd = L_sigma_md[1]
            eta_esv = max(L_eta)

            # Theorie II. Ordnung
            theta = 0.0025
            E = E0_mean/1.3
            no_iter = 5

            # Funktion
            L_e, L_M, L_e_total, L_M_total, sigma_cd, L_sigma_mIId, eta_th2, ii = ec5_63_th2(
                Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k, f_myk, f_mzk, E, k_mod, gamma, no_iter)

            # Auswertung der Ergebnisse
            L_e_y = L_e[0]
            L_M_y = L_M[0]
            L_e_total_y = L_e_total[0]
            L_M_total_y = L_M_total[0]

            L_e_z = L_e[1]
            L_M_z = L_M[1]
            L_e_total_z = L_e_total[1]
            L_M_total_z = L_e_total[1]

            # maximale Ausnutzung
            eta = max(eta_esv, eta_th2)
            ws.range('AS10').offset(j, i).value = güte
            ws.range('AX10').offset(j, i).value = round(b, 2)
            ws.range('BC10').offset(j, i).value = round(eta, 2)

            # FALL 1: Iteration fertig / Ausnutzung bereits erreicht
            if i == 4 and (eta > eta_target_top or eta < eta_target_bot):

                # Status
                ws.range('BH10').offset(j, 0).value = a2

                # ESV
                ws.range('Y10').offset(j, 0).value = round(lamb, 2)
                ws.range('Z10').offset(j, 0).value = round(lamb_rel, 2)
                ws.range('AA10').offset(j, 0).value = round(kc, 2)
                ws.range('AB10').offset(j, 0).value = round(k_crit, 2)
                ws.range('AC10').offset(j, 0).value = nw
                ws.range('AD10').offset(j, 0).value = sigma_cd
                ws.range('AE10').offset(j, 0).value = round(sigma_myd, 2)
                ws.range('AF10').offset(j, 0).value = round(sigma_mzd, 2)
                ws.range('AG10').offset(j, 0).value = round(eta_esv, 2)

                # TH2
                ws.range('AI10').offset(j, 0).value = L_e[0][0]
                ws.range('AJ10').offset(j, 0).value = L_M[0][0]
                ws.range('AK10').offset(j, 0).value = L_e_total[0][-1]
                ws.range('AL10').offset(j, 0).value = L_M_total[0][-1]

                ws.range('AM10').offset(j, 0).value = L_e[1][0]
                ws.range('AN10').offset(j, 0).value = L_M[1][0]
                ws.range('AO10').offset(j, 0).value = L_e_total[1][-1]
```

```
                ws.range('AP10').offset(j, 0).value = L_M_total[1][-1]
                ws.range('AQ10').offset(j, 0).value = eta_th2

            # FALL 2: Ausnutzung erreicht
            elif eta <= eta_target_top and eta >= eta_target_bot:

                # ESV
                ws.range('Y10').offset(j, 0).value = round(lamb, 2)
                ws.range('Z10').offset(j, 0).value = round(lamb_rel, 2)
                ws.range('AA10').offset(j, 0).value = round(kc, 2)
                ws.range('AB10').offset(j, 0).value = round(k_crit, 2)
                ws.range('AC10').offset(j, 0).value = nw
                ws.range('AD10').offset(j, 0).value = sigma_cd
                ws.range('AE10').offset(j, 0).value = round(sigma_myd, 2)
                ws.range('AF10').offset(j, 0).value = round(sigma_mzd, 2)
                ws.range('AG10').offset(j, 0).value = round(eta_esv, 2)

                # TH2
                ws.range('AI10').offset(j, 0).value = L_e[0][0]
                ws.range('AJ10').offset(j, 0).value = L_M[0][0]
                ws.range('AK10').offset(j, 0).value = L_e_total[0][-1]
                ws.range('AL10').offset(j, 0).value = L_M_total[0][-1]

                ws.range('AM10').offset(j, 0).value = L_e[1][0]
                ws.range('AN10').offset(j, 0).value = L_M[1][0]
                ws.range('AO10').offset(j, 0).value = L_e_total[1][-1]
                ws.range('AP10').offset(j, 0).value = L_M_total[1][-1]
                ws.range('AQ10').offset(j, 0).value = eta_th2

                # PARAMETER
                ws.range('K10').offset(j, 0).value = round(b, 2)
                ws.range('L10').offset(j, 0).value = round(h, 2)
                ws.range('P10').offset(j, 0).value = güte

                ws.range('AS10').offset(j, i).value = güte
                ws.range('AX10').offset(j, i).value = round(b, 2)
                ws.range('BC10').offset(j, i).value = round(eta, 2)

                # ITERATION
                ws.range('BH10').offset(j, 0).value = a1
                break

            # FALL 3: Ausnutzung über 100%
            elif eta > 1:
                print('here')
                # Güte: Grenze bereits erreicht
                if güte == para2_target_top:
                    print(güte, para2_target_top)
                    # Querschnitt raufgesetzt
                    b = min(b+2*para1_steps, para1_target_top)
                    h = b
                    print(b, h)
                    if b == para1_target_top and i < 4:
                        ws.range('BH10').offset(j, 0).value = a3
                        break

                else:
                    # Querschnitt bleibt / Güte wird raufgesetzt
```

```
                        b = b
                        h = h
                        print(güte)
                        güte = L_güte[min(index+1, index_top)]
                        print(güte)

                # FALL 4: Ausnutzung über eta_top
                elif eta > eta_target_top:
                    # Güte: Grenze bereits erreicht
                    if güte == para2_target_top:
                        # Querschnitt raufgesetzt
                        b = min(b+para1_steps, para1_target_top)
                        h = b

                        if b == para1_target_top and i < 4:
                            ws.range('BH10').offset(j, 0).value = a3
                            break

                    else:
                        # Querschnitt bleibt / Güte wird raufgesetzt
                        b = b
                        h = h
                        print(güte)
                        print(index)
                        güte = L_güte[min(index+1, index_top)]
                        print(güte)

                # FALL 5: Ausnutzung unter eta_bot
                elif eta < eta_target_bot:
                    # Güte: Grenze bereits erreicht
                    if güte == para2_target_bot:
                        b = max(b-para1_steps, para1_target_bot)
                        h = b

                        if b == para1_target_bot and i < 4:
                            ws.range('BH10').offset(j, 0).value = a3
                            break
                    else:
                        b = b
                        h = h
                        güte = L_güte[max(index-1, index_bot)]

                # ITERATION
                ws.range('P10').offset(j, 0).value = güte
                ws.range('K10').offset(j, 0).value = round(b, 2)
                ws.range('L10').offset(j, 0).value = round(h, 2)

            else:
                continue

    # Status
    ws.range('BH8').value = 'abgeschlossen'
```

A8. Python-Code: Dokumentation – AutoCAD

```python
# Stützenbemessungsprogramm: Dokumentation - Grafische Übersicht mit AutoCAD
# Bibliotheken
from pyautocad import Autocad, APoint, aDouble, ACAD  # Interaktion mit AutoCAD
import xlwings as xw  # Interaktion mit Excel

# Verknüpfung mit Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws_1 = wb.sheets['Lastabtrag']
ws_2 = wb.sheets['Stützenbemessung']

# Dynamisches Einlesen der Zeilenanzahl
startrow = 10
rownum = ws_2.range('G10').current_region.last_cell.row

# Koordinaten der Stützen und Decken
# Stützen
x_coord = ws_1.range((10, 13), (rownum, 13)).value
y_coord = ws_1.range((10, 14), (rownum, 14)).value

# Gebäudeaußenkante
xx_coord = ws_1.range((10, 17), (22, 17)).value
yy_coord = ws_1.range((10, 18), (22, 18)).value

# Einlesen der Stützenkennwerte in Listen
pos = ws_2.range((startrow, 7), (rownum, 7)).value
geschoss = ws_2.range((startrow, 8), (rownum, 8)).value
material = ws_2.range((startrow, 16), (rownum, 16)).value
L = ws_2.range((startrow, 10), (rownum, 10)).value
b = ws_2.range((startrow, 11), (rownum, 11)).value
h = ws_2.range((startrow, 12), (rownum, 12)).value
Ned = ws_2.range((startrow, 13), (rownum, 13)).value
Myd = ws_2.range((startrow, 14), (rownum, 14)).value
Mzd = ws_2.range((startrow, 15), (rownum, 15)).value

# Auswertung für die farbliche Gruppierung
# Güte
L_güte = ['GL 24h', 'GL 24c', 'GL 28h', 'GL 28c', 'GL 32h', 'GL 32c']
L_color_güte = [1, 1, 2, 2, 3, 3]

# Querschnitt
L_qs = []
b_oben = ws_2.range('D32').value
b_unten = ws_2.range('D33').value
b_steps = ws_2.range('D34').value
n_steps = (b_oben-b_unten)/b_steps
L_color_qs = []

for i in range(int(n_steps)):
    L_qs.append(round(b_unten+b_steps*i, 2))
    L_color_qs.append(i)

L_qs[-1] = b_oben

# AutoCAD
# Öffnen einer neuen Datei
acad = Autocad(create_if_not_exists=True)
#pii = APoint(xx_coord[0], yy_coord[0])

# Zeichnen der Decke
for i in range(0, 12):
```

```
61          p1 = APoint(xx_coord[i], yy_coord[i])  # Punkt 1
62          p2 = APoint(xx_coord[i+1], yy_coord[i+1])  # Punkt 2
63          line1 = acad.model.AddLine(p1, p2)  # Linie zwischen Punkt 1 und 2
64
65      # Stützen
66      for i in range(len(x_coord)):
67
68          # Index für Güte und Querschnitt
69          index_güte = L_güte.index(material[i])
70          index_qs = L_qs.index(round(b[i], 2))
71
72          # Punktkoordinaten
73          pi = APoint(x_coord[i], y_coord[i])
74          p1 = APoint(x_coord[i]+b[i]*0.5, y_coord[i]+h[i]*0.5)  # Punkt 1
75          p2 = APoint(x_coord[i]+b[i]*0.5, y_coord[i]-h[i]*0.5)  # Punkt 2
76          p3 = APoint(x_coord[i]-b[i]*0.5, y_coord[i]-h[i]*0.5)  # Punkt 3
77          p4 = APoint(x_coord[i]-b[i]*0.5, y_coord[i]+h[i]*0.5)  # Punkt 4
78
79          # Zeichnen der Stütze
80          line1 = acad.model.AddLine(p1, p2)  # Linie zwischen Punkt 1 und 2
81          line2 = acad.model.AddLine(p2, p3)  # Linie zwischen Punkt 2 und 3
82          line3 = acad.model.AddLine(p3, p4)  # Linie zwischen Punkt 3 und 4
83          line4 = acad.model.AddLine(p4, p1)  # Linie zwischen Punkt 4 und 1
84
85          # Anpassung der Farbe gemäß Index
86          line1.Color = L_color_qs[index_qs]
87          line2.Color = L_color_qs[index_qs]
88          line3.Color = L_color_qs[index_qs]
89          line4.Color = L_color_qs[index_qs]
90
91          # Textelemente
92          text = acad.model.AddText(pos[i] + ' - ' + geschoss[i], pi, 0.2)
93          text1 = acad.model.AddText(
94              material[i], APoint(x_coord[i], y_coord[i]-0.3), 0.2)
95          text2 = acad.model.AddText('L/B/H = '+str(round(L[i], 2))+'/'+str(
96              b[i]) + '/' + str(h[i])+' [m]', APoint(x_coord[i],
                y_coord[i]-0.6), 0.2)
97          text3 = acad.model.AddText('Ned/Myd/Mzd = '+str(round(Ned[i],
            2))+'/'+str(
98              Myd[i]) + '/' + str(Mzd[i]), APoint(x_coord[i],
                y_coord[i]-0.9), 0.2)
99
100         # Anpassung der Farbe gemäß Index
101         text1.Color = L_color_güte[index_güte]
```

A9. Python-Code: Dokumentation – PDF

```python
# Stützenbemessungsprogramm: Dokumentation - Schriftlicher Einzelnachweis als PDF
# Bibliotheken
import xlwings as xw  # Interaktion mit Excel
from python_code_funktion_th2 import ec5_63_th2

# Verknüpfung mit Excel
wb = xw.Book('Stützenbemessung_ec5_63.xlsm')
ws_1 = wb.sheets['Stützenbemessung']
ws_2 = wb.sheets['Theorie II. Ordnung']

# Dynamisches Einlesen der Zeilenanzahl
startrow = 10
rownum = ws_1.range('G10').current_region.last_cell.row
colnum = ws_1.range((startrow, 1)).current_region.last_cell.column

# Einlesen der Listen
L_pos = ws_1.range((startrow, 7), (rownum, 7)).value
L_ges = ws_1.range((startrow, 8), (rownum, 8)).value
L_system = ws_1.range((startrow, 9), (rownum, 9)).value
L_L = ws_1.range((startrow, 10), (rownum, 10)).value
L_b = ws_1.range((startrow, 11), (rownum, 11)).value
L_h = ws_1.range((startrow, 12), (rownum, 12)).value
L_N_ed = ws_1.range((startrow, 13), (rownum, 13)).value
L_M_yd = ws_1.range((startrow, 14), (rownum, 14)).value
L_M_zd = ws_1.range((startrow, 15), (rownum, 15)).value
L_h_art = ws_1.range((startrow, 16), (rownum, 15)).value
L_f_c0k = ws_1.range((startrow, 17), (rownum, 17)).value
L_f_myk = ws_1.range((startrow, 18), (rownum, 18)).value
L_E0mean = ws_1.range((startrow, 19), (rownum, 19)).value
L_E_05 = ws_1.range((startrow, 20), (rownum, 20)).value
L_G_05 = ws_1.range((startrow, 21), (rownum, 21)).value
L_k_mod = ws_1.range((startrow, 22), (rownum, 22)).value
L_gamma = ws_1.range((startrow, 23), (rownum, 23)).value

# Schleife durch Stützenpositionen
for j in range(rownum - startrow + 1):

    # Ermittlung der gekennzeichneten Positionen
    if ws_1.range("BJ10").offset(j, 0).value == "x":

        # Einlesen der Kennwerte gemäß Index
        L = L_L[j]
        b = L_b[j]
        h = L_h[j]
        f_c0k = L_f_c0k[j]*1000
        f_myk = L_f_myk[j]*1000
        f_mzk = f_myk
        E0_mean = L_E0mean[j]*1000
        E_05 = L_E_05[j]*1000
        k_mod = L_k_mod[j]
        gamma = L_gamma[j]
        N_ed = L_N_ed[j]
        M_yd = L_M_yd[j]
        M_zd = L_M_zd[j]
        güte = L_h_art[j]
        Lagerung = L_system[j]
        theta = ws_1.range("D24").value
        no_iter = 9
        E = E0_mean/1.3

        # Auswertung der Kennwerte
```

```
            shi = k_mod/gamma
            f_c0d = f_c0k*shi
            f_myd = f_myk*shi

            A = b*h
            I = b*h**3/12
            w = (b*h**2)/6
            i = (I/A)**0.5

            # Bemessung mit Funktion
            L_e, L_M, L_e_total, L_M_total, sigma_cd, L_sigma_mIId, eta, index = ec5_63_th2(
                Lagerung, güte, L, b, h, N_ed, M_yd, M_zd, theta, f_c0k, f_myk, f_mzk, E, k_mod, gamma, no_iter)

            #Schreiben in bemessungsvorlage
            # Geometrie
            ws_2.range('I4').value = L_pos[j]
            ws_2.range('T49').value = index

            # Geometrie
            start = ws_2.range('E14')
            start.offset(0, 0).value = L
            start.offset(0, 13).value = b
            start.offset(0, 26).value = h

            # Bemessungswerte der Einwirkung
            start = ws_2.range('E17')
            start.offset(0, 0).value = N_ed
            start.offset(0, 13).value = M_yd
            start.offset(0, 26).value = M_zd

            # Charakteristische Festigkeitswerte
            start = ws_2.range('F21')
            start.offset(0, 0).value = f_c0k
            start.offset(0, 16).value = f_myk
            start.offset(1, 0).value = E0_mean

            # Widerstandsbeiwerte
            start = ws_2.range('F25')
            start.offset(0, 0).value = k_mod
            start.offset(0, 16).value = gamma

            # Charakteristische Festigkeitswerte
            start = ws_2.range('F34')
            start.offset(0, 0).value = f_c0d
            start.offset(0, 16).value = f_myd
            start.offset(1, 0).value = E0_mean/1.3

            # Schnittgrößenermittlung nach Theorie II. Ordnung
            start = ws_2.range('E61')
            start.offset(0, 0).options(transpose=True).value = L_e[0]
            start.offset(0, 6).options(transpose=True).value = L_M[0]
            start.offset(0, 12).options(transpose=True).value = L_e_total[0]
            start.offset(0, 18).options(transpose=True).value = L_M_total[0]

            start = ws_2.range('E80')
            start.offset(0, 0).options(transpose=True).value = L_e[1]
            start.offset(0, 6).options(transpose=True).value = L_M[1]
            start.offset(0, 12).options(transpose=True).value =
```

```
            L_e_total[1]
            start.offset(0, 18).options(transpose=True).value =
            L_M_total[1]

            # PDF export
            pfad = ws_1.range("D41").value
            pdf_file_name = pfad + "Stützenbemessung_ThII_" + L_pos[j]
            ws_2.to_pdf(pdf_file_name)
```

A10. Excel PDF-Ausdruck: Einzelnachweis – Stützenbemessung Theorie II. Ordnung

Projektdaten

Projekt:		Bauteil:		Leistungsphase:
Position:	**S-201**	Verfasser:	CM	

DIN EN 1995-1-1: 6.3 Stabilität von Bauteilen

6.3.1 Allgemeines

(1) Die Biegespannungen infolge spannungsloser Vorverformungen und Anfangskrümmungen, Ausmittigkeiten und eingeprägter Durchbiegungen sind zusätzlich zu solchen infolge Querlasten zu

Eingangswerte

Geometrie

L	=	3,00	m	b	=	0,50	m	h	=	0,40

Bemessungswerte der Einwirkung

N_{ed}	=	2.000	kN	M_{yd}	=	50	kNm	M_{zd}	=	30

Charakteristische Festigkeitswerte

Güte:

$f_{c,0,k}$	=	28.000	kN/m²	$f_{m,y,k}$	=	28.000	kN/m²
E_{mean}	=	12.500.000	kN/m²				

Widerstandsbeiwerte

k_{mod}	=	0,60	-	γ_m	=	1,30	-

Berechnung

Querschnittsparameter

w_y	=	0,013	m^3	w_z	=	0,017	m^3
i_y	=	0,115	m^4	i_z	=	0,144	m^4
I_y	=	0,003	m²	I_z	=	0,004	m²

Bemessungswerte der Festigkeit

$f_{c,0,d}$	=	12.923	kN/m²	$f_{m,y,d}$	=	12.923	kN/m²
E_{meand}	=	9.615.385	kN/m²				

Schnittgrößenermittlung
Schnittgrößen nach Theorie I. Ordnung

θ	=	0,0025			angesetzt in		y
e0	=	θ	*	L	=	0,0075	m
M0	=	e0	*	Ned	=	15,0	kNm

Schnittgrößen nach Theorie II. Ordnung
Iteration um die y-Achse

mit $e_i = \frac{M_i * L_{eff}{}^2}{E_{0,05} * I * \pi^2}$ und $M_i = e_i * N_{ed} + M_{yd}$

i	ei	Mi	e	M_{yd}
-	*mm*	*kNm*	*mm*	*kNm*
0	7,500	65,0	7,500	65,0
1	2,400	4,8	9,900	69,8
2	0,200	0,3	10,100	70,1
3	0,000	0,0	10,100	70,1
4	0,000	0,0	10,100	70,1
5	0,000	0,0	10,100	70,1
6	0,000	0,0	10,100	70,1
7	0,000	0,0	10,100	70,1
8	0,000	0,0	10,100	70,1
9	0,000	0,0	10,100	70,1
SUM	**10,1**	**70,1**	**10,1**	**70,1**

σ_{myd} = 5.260 kN/m²

Iteration um die z-Achse

i	ei	Mi	e	M_{zd}
-	*mm*	*kNm*	*mm*	*kNm*
0	0,0	30,0	0,00	30,0
1	0,7	1,4	0,70	31,4
2	0,0	0,1	0,70	31,5
3	0,0	0,0	0,70	31,5
4	0,0	0,0	0,70	31,5
5	0,0	0,0	0,70	31,5
6	0,0	0,0	0,70	31,5
7	0,0	0,0	0,70	31,5
8	0,0	0,0	0,70	31,5
9	0,0	0,0	0,70	31,5
SUM	**0,7**	**31,5**	**0,7**	**31,5**

σ_{mzd} = 1.888 kN/m²

Spannungen

σ_{cd} = ##### kN/m²

σ_{myd} = 5.260 kN/m²

σ_{mzd} = 1.888 kN/m²

Nachweis
Gl. (6.19)

$$\eta = \left(\frac{10.000}{12.923}\right)^2 + \frac{5.260 * 1}{12.923} + \frac{1.888 * 0,7}{12.923} = 1,11$$

$$\eta = \left(\frac{10.000}{12.923}\right)^2 + \frac{5.260 * 0,7}{12.923} + \frac{1.888 * 1}{12.923} = 1,03$$

A11. Excel CAD-Ausdruck

A12. Python-Code: Funktion – Vollgewindeschrauben

```python
# Schraubenbemessungsprogramm: Funktionen - Axial- und Schertragfähigkeit von Würth Vollgewindeschrauben
# Bibliotheken
import streamlit as st
import pandas as pd
from math import sqrt, pi, cos, sin

# Funktion
def ec5_87_tragfähigkeit_vg(Hersteller, d, L, t_1, t_2, t_Blech, rho_k, alpha):

    # Nachweisführung
    nw = []

    #__________Fehlermeldungen__________

    if t_Blech != 0 and t_1 != 0 and t_2 != 0:
        fehler = "Error: Es werden keine zweischnittigen Verbindungen berechnet."
        hinweis = "Hinweis: t_Blech <> 0 and t_1 <> 0 and t_2 <> 0"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

    elif t_Blech == 0 and t_1 == 0 and t_2 == 0:
        fehler = "Error: Die Dicken sind unzulässig."
        hinweis = "Hinweis: t_Blech = 0 and t_1 = 0 and t_2 = 0"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

    elif L in [t_1, t_Blech]:
        fehler = "Error: Die Dicken sind unzulässig."
        hinweis = "Hinweis: L = t_1 or L = t_Blech"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

    elif t_Blech == 0 and t_2 == 0:
        fehler = "Error: Die Dicken sind unzulässig."
        hinweis = "Hinweis: t_Blech = 0 and t_2 = 0"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

    elif t_Blech == 0 and t_1 > L:
        fehler = "Error: Die Dicken sind unzulässig."
        hinweis = "Hinweis: t_Blech = 0 and t_1 > L"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

```

```
    ## Holz-Stahl-Verbindungen
    elif t_Blech != 0 and t_1 == 0 and t_2 != 0:
        fehler = "Bei Stahl-Holz Verbindungen darf die Seitenholzdicke 1 nicht gleich null sein."
        hinweis = "Hinweis: t_Blech <> 0 and t_1 = 0"
        nw.append(fehler)
        nw.append(hinweis)
        F_axRk = 0
        F_vRk = 0
        return F_axRk, F_vRk, nw

    else:
        #__________Vorbereitung__________

        nw.append('success')

        # Listen
        # herstellerspezifische Kennwerte
        if Hersteller == 'Würth':
            # WÜrth
            L_d = [6, 8, 10, 12]
            L_f_axk = [11.5, 11, 10, 10]
            L_f_tensk = [11, 20, 32, 45]
            L_d_h = [14, 22, 25.2, 29.4]
            L_f_head = [13, 13, 10, 10]

        # Bearbeitung der Eingabwerte
        # Index
        index = L_d.index(d)

        # Zugriff auf Listenelemente
        d_h = L_d_h[index]
        f_head = L_f_head[index]
        f_axk = L_f_axk[index]
        f_1k = 80 * 10**(-6) * rho_k**2  # N/mm2
        f_2k = 80 * 10**(-6) * rho_k**2  # N/mm2
        Kopfdurchziehparameter
        f_tensk = L_f_tensk[index]
        M_yrk = 0.15 * 600 * d**(2.6)
        f_hk = (0.082 * rho_k * d**(-0.3))

        # Umrechnung in Bogenmaß
        alph = alpha * pi / 180

        # Effektive Länge der Schraube
        if t_Blech != 0:
            l_ef = L - t_Blech
        elif t_Blech == 0:
            l_ef = min(L - t_1, t_1)

        #__________Axialtragfähigkeit__________

        # Ausziehwiderstand
        if alpha <= 45:
            k_axk = 0.3 + (0.7 * alpha) / 45
        else:
            k_axk = 1

        F_axrk1 = (k_axk * f_axk * d * l_ef) * (rho_k / 350)**0.8  # N

        # Umrechnung in Kilonewton
        F_axRk1 = F_axrk1 / 1000  # kN
```

```python

        # Kopfdurchziehwiderstand
        F_headrk = f_head * (d_h)**2 * (rho_k / 350)**0.8  # N

        # Umrechnung in Kilonewton
        F_headRk = F_headrk / 1000  # kN

        # Zugfestigkeit
        F_tRk = f_tensk  # kN

        # resultierende Axialtragfähigkeit
        # if-Abfrage, ob Kopfdurchziehwiderstand berücksichtigt werden muss
        if t_Blech != 0 or t_1 > 4 * d:
            F_axRk = min(F_axRk1, F_tRk)
            nw.append('A: o. Kopfdurchzieh')

            # Umrechnung in Kilonewton
            F_axrk = min(F_axrk1, F_tRk * 1000)

        elif t_Blech == 0 or t_1 < 4 * d:
            F_axRk = min(F_axRk1, F_headRk, F_tRk)
            nw.append('A: m. Kopfdurchzieh')

            # Umrechnung in Kilonewton
            F_axrk = min(F_axrk1, F_headrk, F_tRk * 1000)

        #__________Schertragfähigkeit__________

        # einschnittige Holz-Holz-Verbindung
        if t_Blech == 0:
            nw.append('S: H-H')

            F_vk1 = round(f_hk * t_1 * d, 2)
            F_vk2 = round(f_hk * l_ef * d, 2)
            F_vk3 = round((f_hk * t_1 * d) / 2 * (sqrt(1 + 2 * (1 + l_ef / t_1 +
                            (l_ef / t_1)**2) + (l_ef / t_1)**2) - (1 + l_ef / t_1)) + F_axrk / 4, 2)
            F_vk4 = 1.05 * (f_hk * t_1 * d) / (3) * (sqrt(4 +
                                                        (12 * M_yrk) / (f_hk * t_1**2 * d)) - 1) + F_axrk / 4
            F_vk5 = 1.05 * (f_hk * l_ef * d) / (3) * (sqrt(4 +
                                                        (12 * M_yrk) / (f_hk * l_ef**2 * d)) - 1) + F_axrk / 4
            F_vk6 = 1.15 * sqrt(2 * M_yrk * f_hk * d) + F_axrk / 4
            F_vrk = min(F_vk1, F_vk2, F_vk3, F_vk4, F_vk5, F_vk6)

            # Umrechnung in Kilonewton
            F_vRk1 = round(F_vk1 / 1000, 2)
            F_vRk2 = round(F_vk2 / 1000, 2)
            F_vRk3 = round(F_vk3 / 1000, 2)
            F_vRk4 = round(F_vk4 / 1000, 2)
            F_vRk5 = round(F_vk5 / 1000, 2)
            F_vRk6 = round(F_vk6 / 1000, 2)
            F_vRk = round(F_vrk / 1000, 2)

        # Einschnittige Holz-Stahl-Verbindung (dickes Blech)¶
        elif t_Blech >= d:
            nw.append('S: H-S (dick)')

            F_vk1 = f_hk * t_1 * d
```

```
            F_vk2 = f_hk * t_1 * d * \
                (sqrt(2 + (4 * M_yrk) / (f_hk * d * t_1**2)) - 1) + F_axrk / 4
            F_vk3 = 2.3 * sqrt(M_yrk * f_hk * d) + F_axrk / 4
            F_vrk = min(F_vk1, F_vk2, F_vk3)

            # Umrechnung in Kilonewton
            F_vRk1 = round(F_vk1 / 1000, 2)
            F_vRk2 = round(F_vk2 / 1000, 2)
            F_vRk3 = round(F_vk3 / 1000, 2)
            F_vRk4 = '-'
            F_vRk5 = '-'
            F_vRk6 = '-'
            F_vRk = round(F_vrk / 1000, 2)

        # Einschnittige Holz-Stahl-Verbindung (dünnes Blech)¶
        elif t_Blech < d and t_Blech != 0:
            nw.append('S: H-S (dünn)')

            F_vk1 = 0.4 * f_hk * t_1 * d
            F_vk2 = 1.15 * sqrt(2 * M_yrk * f_hk * d) + F_axrk / 4
            F_vrk = min(F_vk1, F_vk2)

            # Umrechnung in Kilonewton
            F_vRk1 = round(F_vk1 / 1000, 2)
            F_vRk2 = round(F_vk2 / 1000, 2)
            F_vRk3 = '-'
            F_vRk4 = '-'
            F_vRk5 = '-'
            F_vRk6 = '-'
            F_vRk = round(F_vrk / 1000, 2)

        return round(F_axRk, 2), F_vRk, nw

# Funktion: verfügbaren Längen der Schrauben
def get_length(hersteller, d):

    L_d = [6, 8, 10, 12]
    index = L_d.index(d)
    if hersteller == 'Würth':
        # Würth
        L_L = [[120,140,160,180,200,220,240,260],
               [120,140,160,180,200,220,240,260,280,300,330,380,430,480,530,580],
               [120,140,160,180,200,220,240,260,280,300,320,340,360,380,400,430,480,530,580,600,650,700,750,800],
               [120,140,160,180,200,220,240,260,280,300,380,480,600]]

    elif hersteller == 'Spax':
        # SPAX
        L_L = [[120,140,160,180,200,220,240,260,280,300,350,400,450,500,550,600],
               [120,140,160,180,200,220,240,260,280,300,350,400,450,500,550,600],
               [120,160,200,220,240,260,280,300,350,400,450,500,550,600,700,800],
               [200,240,280,300,350,400,450,500,550,600]]
```

```
228
229         return L_L, L_L[index]
230
```

A13. Jupyter Notebook PDF-Ausdruck: Einzelnachweis – Schraubenbemessung

A14. Python-Code: Web-App mit Streamlit

```python
# Schraubenbemessungsprogramm: Webapp mit Streamlit - Axial- und Schertragfähigkeit von Würth Vollgewindeschrauben
# Bibliotheken
from math import pi, sqrt
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import streamlit as st
from schraubenbemessung_ec5_87_functions import get_length, ec5_87_tragfähigkeit_vg

# HTML Einstellungen
st.set_page_config(layout="wide")
st.markdown("""<style>
[data-testid="stSidebar"][aria-expanded="false"] > div:first-child {width: 500px;}
[data-testid="stSidebar"][aria-expanded="false"] > div:first-child {width: 500px;margin-left: -500px;}
</style>""",unsafe_allow_html=True)

# Eingangsparameter
# Listen
L_kled = ['ständig', 'lang', 'mittel', 'kurz', 'sehr kurz']
L_kmod = [[0.6, 0.7, 0.8, 0.9, 1.1], [
    0.6, 0.7, 0.8, 0.9, 1.1], [0.5, 0.55, 0.65, 0.7, 0.9]]
L_rho_k = [380, 350, 410, 380, 430, 410, 450, 430]
L_holzart = ['GL24h', 'GL24c', 'GL28h',
             'GL28c', 'GL32h', 'GL32c', 'GL36h', 'GL36c']
L_di_axial = [[], [], [], []]
L_di_scher = [[], [], [], []]
L_no = [[], [], [], []]
L_d = [6, 8, 10, 12]


#__________Seitenleiste__________

with st.sidebar:

    # Eingabewerte
    # Überschrift
    st.title('Eingangsparameter')
    st.subheader('Beiwerte')

    # Nachweisführung
    nachweisführung = st.radio(
        'Nachweisführung', ('Charakteristische Werte', 'Bemessungswerte'), horizontal=True)
    if nachweisführung == 'Bemessungswerte':
        nkl = st.selectbox('Nutzungsklasse', [1, 2, 3])
        kled = st.selectbox('Lasteinwirkung', L_kled)
        index = L_kled.index(kled)
        k_mod = L_kmod[nkl][index]
        gamma = 1.3
        chi = k_mod/gamma
    else:
        chi = 1

    # Verbindungsmittel
    st.subheader('Verbindungsmittel')
    hersteller = 'Würth'
    holzart = st.selectbox('Holzgüte', L_holzart)
    t_1 = int(st.text_input('Seitenholz 1 [mm]', 50))
```

```
    t_2 = int(st.text_input('Seitenholz 2 [mm]', 100))
    t_Blech = int(st.text_input('Blechdicke [mm]', 0))
    alpha = int(st.text_input('Winkel', 0))

    # Einwirkung
    st.subheader('Einwirkung')
    F_t = int(st.text_input('Zugkraft [kN]', 4))
    F_v = int(st.text_input('Querkraft [kN]', 2))
    blank = st.write('')
    button_einlesen = st.button('Einlesen')
    blank = st.write('')

    # Bearbeitung der Eingabwerte
    if t_Blech == 0:
        L_tot = t_1+t_2  # Länge
    else:
        L_tot = t_1+t_Blech  # Länge
    L_L, L_Li = get_length(hersteller, 6)  # Liste der Längen
    index = L_holzart.index(holzart)  # Rohdichte des Holzes (Index)
    rho_k = L_rho_k[index]  # Rohdichte des Holzes

    #__________Berechnung der Tragfähigkeit__________

    if 'Einlesen':

        # Berechnung aller Tragfähigkeiten mit Funktion ec5_87_tragfähigkeit_vg()
        for n, d in enumerate(L_d):
            for m, L in enumerate(L_L[n]):
                F_axRk, F_vRk, nw = ec5_87_tragfähigkeit_vg(
                    'Würth', d, L, t_1, t_2, t_Blech, rho_k, alpha)
                L_di_axial[n].append(round(F_axRk*chi, 2))
                L_di_scher[n].append(round(F_vRk*chi, 2))
                L_no[n].append(m+1.5)

        # Dictionary für jeden Durchmesser
        # Durchmesser 6 mm
        dict_d6 = {'L': L_L[0], 'F_axRk': L_di_axial[0],
                   'F_vRk': L_di_scher[0]}
        df_d6 = pd.DataFrame(dict_d6)
        # Durchmesser 8 mm
        dict_d8 = {'L': L_L[1], 'F_axRk': L_di_axial[1],
                   'F_vRk': L_di_scher[1]}
        df_d8 = pd.DataFrame(dict_d8)
        # Durchmesser 10 mm
        dict_d10 = {'L': L_L[2], 'F_axRk': L_di_axial[2],
                    'F_vRk': L_di_scher[2]}
        df_d10 = pd.DataFrame(dict_d10)
        # Durchmesser 12 mm
        dict_d12 = {'L': L_L[3], 'F_axRk': L_di_axial[3],
                    'F_vRk': L_di_scher[3]}
        df_d12 = pd.DataFrame(dict_d12)
        # Dictionaries in Listen verpacken
        L_dict = [df_d6, df_d8, df_d10, df_d12]


#__________Hauptseite__________

# Text
st.image('image_header_vg.png')
st.title('Bemessung von Vollgewindeschrauben')
st.write('DIN EN 1995-1-1 Abs. 8 & ETA-11/0190')
```

```
st.write('Dieses Programm ermittelt die Tragfähigkeiten von Vollgewindeschrauben hinsichtlich einer Axial- und Scherbeanspruchung. \
        Die herstellerspezifischen Kennwerte beziehen sich auf eine **Würth ASSY plus VG** als Senkkopfausführung.')
st.image('image_vg.png')
st.subheader('Nachweisführung')
betrachtung = st.selectbox('', ('Bemessungstabellen', 'Einzelnachweis'))

# Widerstandsbeiwerte
if nachweisführung == 'Bemessungswerte':
    st.write('Widerstandsbeiwerte')
    col1, col2 = st.columns(2, gap="small")
    with col1:
        st.latex('k_{mod} = ' + str(k_mod))
    with col2:
        st.latex('\gamma_{m} = ' + str(gamma))

#__________Bemessungstabelle__________

if betrachtung == 'Bemessungstabellen':

    # Text
    st.title('Bemessungstabellen')
    st.subheader('Filter')

    # Filter
    options_filter = st.multiselect('', ['Längen', 'Tragfähigkeit'])

    # Eingangswerte
    # Listen und Dictionaries
    fig_6 = fig_8 = fig_10 = fig_12 = []
    table_6 = table_8 = table_10 = table_12 = []
    L_fig = [fig_6, fig_8, fig_10, fig_12]
    L_table = [table_6, table_8, table_10, table_12]
    L_keys = ['f6', 'f8', 'f10', 'f12']

    # Tabs
    st.write('Durchmesser')
    col_d6, col_d8, col_d10, col_d12 = st.tabs(
        ["6 mm", "8 mm", "10 mm", "12 mm"])

    # Schleife für Reiter (Tabs)
    for n, tab_i in enumerate([col_d6, col_d8, col_d10, col_d12]):

        # Tab
        with tab_i:

            # Listen
            # Allgemein
            red = '#ff4b4b'
            light_grey = '#f0f2f6'
            grey = '#bfced8'
            dark_grey = 'lightslategrey'
            L_Li = L_L[n]
            L_di_axiali = L_di_axial[n]
            L_di_scheri = L_di_scher[n]
            # Diagramme
            L_title = ["Axialtragfähigkeit", "Schertragfähigkeit"]
            L_text = ["Axialkraft", "Scherkraft"]
            L_hover = ["F_axRd", "F_vRd"]
```

```
            L_F = [F_t, F_v]
            L_colors = [dark_grey, red]
            L_colors_bar = [[], []]
            L_font = {'color': 'white'}
            L_font_prozent = [{'color': 'white'}, {'color': 'white'}]
            L_color_font = [[], []]
            L_color_fill = [[], []]
            # Tabelle
            L_t_fill1 = []
            L_t_fill2 = []
            L_t_fill3 = []
            dict_header = {'values': [('L', '[mm]'),  ('F_axRk', '[kN]'), ('F_vRk', '[kN]')],
                           'align': 'center', 'font': {'size': 15, 'color': 'black'}, 'fill_color': grey, 'line_color': 'white'}
            dict_cells = {'values': [L_Li,  L_di_axiali, L_di_scheri], 'height': 30, 'font': {'size': 15, 'color': 'black'},
                          'fill_color': [L_t_fill1, L_t_fill2, L_t_fill3]}

            # Fehlermeldung
            st.subheader('Würth ASSY Plus VG')
            if nw[0] != 'success':
                st.subheader(nw[0])
                st.write(nw[1])
            else:

                #__________Tabelle__________

                tabelle = L_table[n]

                # Tabellen
                # Dynamische Anpassung der Farben entsprechend der Filter
                # Filter (Tragfähigkeit, Längen, Tragfähigkeit & Längen)
                for i in range(len(L_Li)):

                    if (options_filter == ["Tragfähigkeit", "Längen"] or options_filter == ["Längen", "Tragfähigkeit"]):

                        dict_cells['fill_color'][0].append(light_grey)

                        if L_Li[i] > L_tot or L_di_axiali[i] == 0 or L_di_scheri[i] == 0:

                            dict_cells['fill_color'][1].append(light_grey)

                        elif L_di_axiali[i] > F_t and L_Li[i] < L_tot:

                            dict_cells['fill_color'][1].append(dark_grey)
                        else:

                            dict_cells['fill_color'][1].append(light_grey)

                        if L_Li[i] > L_tot or L_di_axiali[i] == 0 or
```

```
                            L_di_scheri[i] == 0:
                                dict_cells['fill_color'][2].append(light_grey)

                            elif L_Li[i] < L_tot and L_di_scheri[i] > F_v:
                                dict_cells['fill_color'][2].append(red)
                            else:
                                dict_cells['fill_color'][2].append(light_grey)

                        elif (options_filter == ["Tragfähigkeit"] and options_filter != ["Längen"]):
                            dict_cells['fill_color'][0].append(light_grey)

                            if L_di_axiali[i] < F_t:
                                dict_cells['fill_color'][1].append(light_grey)
                            else:
                                dict_cells['fill_color'][1].append(dark_grey)

                            if L_di_scheri[i] < F_v:
                                dict_cells['fill_color'][2].append(light_grey)
                            else:
                                dict_cells['fill_color'][2].append(red)

                        elif (options_filter == ["Längen"] and options_filter != ["Tragfähigkeit"]):
                            dict_cells['fill_color'][0].append(light_grey)

                            if L_Li[i] < L_tot:
                                dict_cells['fill_color'][1].append(dark_grey)
                                dict_cells['fill_color'][2].append(red)
                            else:
                                dict_cells['fill_color'][1].append(light_grey)
                                dict_cells['fill_color'][2].append(light_grey)

                        else:
                            dict_cells['fill_color'][0].append(light_grey)
                            dict_cells['fill_color'][1].append(light_grey)
                            dict_cells['fill_color'][2].append(light_grey)

                    # Erstellen der Tabelle
                    tabelle = go.Figure(
                        data=[go.Table(header=dict_header, cells=dict_cells, columnwidth=[40])])
                    tabelle.update_layout(
                        width=1100, height=600, title_text='Tabelle')
                    st.write(tabelle)

```

```
                    #__________Diagramme__________

                    # Erzeugung von 2 Diagrammen für die Axial- und Schertragfähigkeit
                    for m, Frd_i in enumerate([L_di_axiali, L_di_scheri]):

                        # Plotly Balkendiagramm
                        y_max = max(Frd_i)
                        fig = L_fig[n]
                        annotations = go.Annotations()

                        # Plotly Balkendiagramm
                        fig = px.bar(L_dict[n], x=L_no[n], y=Frd_i, height=600, width=1100,
                                     title=L_title[m], text_auto=True, hover_name='L')

                        # Hinzufügen der x-Achse (Längen)
                        for j in range(len(L_Li)):
                            fig.add_annotation(
                                text=L_Li[j], x=L_no[n][j], y=-y_max/20, showarrow=False)

                        # Anwendung der Filter
                        # Filter Tragfähigkeit
                        for p, Frd in enumerate(Frd_i):

                            if options_filter == ["Tragfähigkeit"]:

                                # Ausnutzung als Beschriftung
                                for k in range(len(L_dict[n])):

                                    # if-Abfrage zur Vermeidung der Division durch 0
                                    if Frd_i[k] == 0:
                                        print("")
                                    else:

                                        annotations.append(go.Annotation(text=str(f"{round(L_F[m]/Frd_i[k],2):.0%}"),
                                                                         x=k+1.5,
                                                                         y=Frd_i[k] + y_max/20,
                                                                         xref='x', yref='y', showarrow=False, font={'color': 'lightslategrey'}))

                                # Dynamische Anpassung der Farben entsprechend der Filter
                                if Frd > L_F[m]:
                                    L_colors_bar[m].append(L_colors[m])
                                else:
                                    L_colors_bar[m].append('lightgray')

                            # Filter Längen
                            elif options_filter == ["Längen"]:

                                L_tot = t_1+t_2
```

```
                                # Dynamische Anpassung der Farben entsprechend der Filter
                                if L_Li[p] < L_tot:
                                    L_colors_bar[m].append(L_colors[m])
                                else:
                                    L_colors_bar[m].append('lightgray')

                            # Filter Längen & Tragfähigkeit
                            elif options_filter == ["Tragfähigkeit", "Längen"] or options_filter == ["Längen", "Tragfähigkeit"]:

                                # Ausnutzung als Beschriftung
                                for k in range(len(L_dict[n])):

                                    # if-Abfrage zur Vermeidung der Division durch 0
                                    if Frd_i[k] == 0:
                                        print("")
                                    else:

                                        annotations.append(go.Annotation(text=str(f"{round(L_F[m]/Frd_i[k],2):.0%}"),
x=k+1.5,
y=Frd_i[k] +
y_max/20,
xref='x', yref='y', showarrow=False, font={'color': 'lightslategrey'}))

                                # Dynamische Anpassung der Farben entsprechend der Filter
                                if Frd > L_F[m] and L_Li[p] < L_tot:
                                    L_colors_bar[m].append(L_colors[m])
                                else:
                                    L_colors_bar[m].append('lightgray')

                            elif options_filter == []:
                                L_colors_bar[m].append(L_colors[m])

                        # Erzeugen der Diagramme
                        fig.update_traces(
                            marker_color=L_colors_bar[m], textfont=L_font_prozent[m])
                        fig.update_layout(annotations=annotations)
                        fig.update_yaxes(title_text="Tragfähigkeit [kN]")
                        fig.update_xaxes(
                            showticklabels=False, tickangle=90, title_text="Längen [mm]", type='linear')
                        fig.update_layout(barmode='group', autosize=False)
                        fig.update_traces(hovertemplate="<br>".join(
                            ["%{L_hover[m]} = %{y} kN"]))
                        st.plotly_chart(fig)

#__________Einzelnachweise__________


```

```python
if betrachtung == 'Einzelnachweis':

    st.title('Einzelnachweis')
    st.subheader('Eingabewerte')
    d = int(st.selectbox('Durchmesser [mm]', (6, 8, 10, 12)))
    L = int(st.selectbox('Längen [mm]', get_length(hersteller, d)[1]))

    # Schalter
    button_axial = st.button('Berechnen')

    if button_axial:

        st.header('Ergebnisse')
        alph = alpha*pi/180
        t_2 = L-t_1
        if t_Blech != 0:
            l_ef = L-t_Blech
        elif t_Blech == 0:
            l_ef = min(L-t_1, t_1)

        # Eingangswerte
        if hersteller == 'Würth':
            # WÜrth
            L_d = [6, 8, 10, 12]
            L_f_axk = [11.5, 11, 10, 10]
            L_f_tensk = [11, 20, 32, 45]
            L_d_h = [14, 22, 25.2, 29.4]
            L_f_head = [13, 13, 10, 10]

        # Index für Kennwerte
        index = L_d.index(d)
        d = L_d[index]
        d_h = L_d_h[index]
        f_head = L_f_head[index]
        f_axk = L_f_axk[index]
        f_tensk = L_f_tensk[index]
        M_yrk = round(0.15*600*d**(2.6), 2)
        f_hk = round((0.082*rho_k*d**(-0.3)), 2)
        k_axk = 0.3+(0.7*alpha)/45 if alpha <= 45 else 1

        # Auflistung der Kennwerte und Eingangswerte
        st.subheader('Kennwerte')
        st.write('ETA-11/0190')
        col1, col2 = st.columns(2)
        with col1:
            st.latex('d_{h} = ' + str(d_h) + ' mm')
            st.latex('l_{eff} = ' + str(l_ef) + ' mm')
            st.latex(r'''\rho_{k} = ''' + str(rho_k) + ' kg/m^3')
            st.latex('k_{axk} = ' + str(k_axk))
        with col2:
            st.latex('f_{headk} = ' + str(f_head) + ' N/mm^2')
            st.latex('f_{axk} = ' + str(f_axk) + ' N/mm^2')
            st.latex('f_{tensk} = ' + str(f_tensk) + ' N/mm^2')
            st.latex('M_{yRk} = ' + str(M_yrk) + ' Nmm')

        #__________Ausziehwiderstand__________

        st.subheader('Ausziehwiderstand')
        # Ausziehwiderstand
        # Berechnung
        F_axrk1 = (k_axk*f_axk*d*l_ef) * (rho_k/350)**0.8  # N
        F_axRk1 = round(F_axrk1/1000, 2)
```

```
            # LATEX
            st.latex(
                r''' F_{axRk}  =  k_{axk} * f_{axk} * l_{ef} * \left(\frac{\rho_{k}}{350} \right)^{0.8} = ''' + str(F_axRk1) + ' kN')

            # Kopfdurchzieh
            # Berechnung
            F_headrk = f_head*(d_h)**2 * (rho_k/350)**0.8
            # F_headrk =  (k_axk*f_axk*d*4*d) * (rho_k/350)**0.8 #N
            F_headRk = round(F_headrk/1000, 2)

            # LATEX
            st.subheader('Kopfdurchziehwiderstand')
            st.latex(
                r'''F_{headRk}  = f_{head} * d_{h}^2 * \left(\frac{\rho_k}{350} \right)^{0.8} = ''' + str(F_headRk) + ' kN')

            # Zugfestigkeit
            # Berechnung
            F_tRk = f_tensk
            # LATEX
            st.subheader('Zugtragfähigkeit')
            st.latex('F_{tRk} = f_{tensk} = ' + str(F_tRk) + ' kN')

            st.subheader('Axialtragfähigkeit')
            st.write('DIN EN 1995-1-1 Abs. 8.7.2')
            if t_Blech != 0 or t_2 > 4*d:
                # if t_Blech != 0:
                F_axRk = min(F_axRk1, F_tRk)
                F_axrk = min(F_axrk1, F_tRk*1000)
                st.write('Kopfdurchziehtragfähigkeit nicht zu berücksichtigen.')
                st.latex(r'''  F_{axRk} = min( ''' + rf'''{F_axRk1}, ''' +
                         rf'''{F_tRk}) = ''' + str(F_axRk) + ' kN')

            elif t_Blech == 0 or t_2 < 4*d:
                # elif t_Blech == 0:
                F_axRk = min(F_axRk1, F_headRk, F_tRk)
                F_axrk = min(F_axrk1, F_headrk, F_tRk*1000)
                st.latex(r'''  F_{axRk} = min( ''' + rf'''{F_axRk1}, ''' +
                         rf'''{F_headRk}, ''' + rf'''{F_tRk})  = ''' + str(F_axRk) + ' kN')

            #__________Schertragfähigkeit__________

            # Holz-Holz
            st.subheader('Schertragfähigkeit')
            if t_Blech == 0:

                F_vk1 = round(f_hk*t_1*d, 2)
                F_vk2 = round(f_hk*t_2*d, 2)
                F_vk3 = round((f_hk*t_1*d)/2 * (sqrt(1+2*(1+t_2/t_1 +
                               (t_2/t_1)**2)+(t_2/t_1)**2)-(1+t_2/t_1))+F_axrk/4, 2)
                F_vk4 = 1.05*(f_hk*t_1*d)/(3) * \
                    (sqrt(4+(12*M_yrk)/(f_hk*t_1**2*d))-1)+F_axrk/4
                F_vk5 = 1.05*(f_hk*t_2*d)/(3) * \
                    (sqrt(4+(12*M_yrk)/(f_hk*t_2**2*d))-1)+F_axrk/4
                F_vk6 = 1.15*sqrt(2*M_yrk*f_hk*d)+F_axrk/4
```

```
                F_vrk = min(F_vk1, F_vk2, F_vk3, F_vk4, F_vk5, F_vk6)

                F_vRk1 = round(F_vk1/1000, 2)
                F_vRk2 = round(F_vk2/1000, 2)
                F_vRk3 = round(F_vk3/1000, 2)
                F_vRk4 = round(F_vk4/1000, 2)
                F_vRk5 = round(F_vk5/1000, 2)
                F_vRk6 = round(F_vk6/1000, 2)
                F_vRk = round(F_vrk/1000, 2)

                st.write('einschnittige Holz-Holz Verbindung.')
                st.write('DIN EN 1995-1-1 Abs. 8.2.2 (8.6)')
                st.latex(r''' F_{vk1} = f_{hk}*t_{1}*d =  ''' +
                         str(F_vRk1) + ' kN')
                st.latex(r''' F_{vk2} = f_{hk}*t_{2}*d =  ''' +
                         str(F_vRk2) + ' kN')
                st.latex(r''' F_{vk3} = \frac{f_{hk}*t_{1}*d}{1+\beta}*\left(\sqrt{1+2\beta^2*(1+\frac{t_2}{t_1}+\left(\frac{t_2}{t_1}\right)^2)+\beta^3\left(\frac{t_2}{t_1}\right)^2} - \beta\left(1+\frac{t_2}{t_1}\right)\right) + \frac{F_{axRk}}{4} = ''' + str(F_vRk3) + ' kN')
                st.latex(r''' F_{vk4} = 1.05 * \frac{f_{hk}*t_{1}*d}{2+\beta}*\left(\sqrt{2\beta*(1+\beta)+\frac{4*\beta*(2+\beta)*M_{yRk}}{f_{hk}*d*t_{1}^2}}-\beta\right) + \frac{F_{axRk}}{4} = ''' + str(F_vRk4) + ' kN')
                st.latex(r''' F_{vk5} = 1.05 * \frac{f_{hk}*t_{2}*d}{1+2\beta}*\left(\sqrt{2\beta^2*(1+\beta)+\frac{4*\beta*(1+2\beta)*M_{yRk}}{f_{hk}*d*t_{2}^2}}-\beta\right) + \frac{F_{axRk}}{4} = ''' + str(F_vRk5) + ' kN')
                st.latex(
                    r''' F_{vk6} = 1.15 * \sqrt{\frac{2\beta}{1+\beta}}*\sqrt{2*M_{yRk}*f_{hk}*d} + \frac{F_{axRk}}{4} = ''' + str(F_vRk6) + ' kN')

                st.latex(r'''  F_{vRk} = min( ''' + rf'''{F_vRk1}, ''' + rf'''{F_vRk2}, ''' + rf'''{F_vRk3}, ''' +
                         rf'''{F_vRk4}, ''' + rf'''{F_vRk5}, ''' +
                         rf'''{F_vRk6})  = ''' + str(F_vRk) + ' kN')

            # Holz-Stahl (einschnittig)
            # dickes Stahlblech
            elif t_Blech >= d:

                F_vk1 = f_hk*t_1*d
                F_vk2 = f_hk*t_1*d*(sqrt(2+(4*M_yrk)/(f_hk*d*t_1**2))-1)+F_axrk/4
                F_vk3 = 2.3*sqrt(M_yrk*f_hk*d)+F_axrk/4
                F_vrk = min(F_vk1, F_vk2, F_vk3)

                F_vRk1 = round(F_vk1/1000, 2)
                F_vRk2 = round(F_vk2/1000, 2)
                F_vRk3 = round(F_vk3/1000, 2)
                F_vRk = round(F_vrk/1000, 2)

                st.write(
                    'Einschnittige Stahl-Holz Verbindung mit einem dicken Stahlblech.')
                st.write('DIN EN 1995-1-1 Abs. 8.2.3 (8.10)')
                st.latex(r''' F_{vk1} = f_{hk}*t_{1}*d =  ''' +
                         str(F_vRk1) + ' kN')
```

```
519                 st.latex(
520                     r''' F_{vk2} = f_{hk}*t_{2}*d * \left( \sqrt{2+ r(
                        \frac{4*M_{yRk}}{f_{hk}*d*t_{1}^2}} ) \right) +
                        \frac{F_{axRk}}{4} = ''' + str(F_vRk2) + ' kN')
521                 st.latex(
522                     r''' F_{vk3} = 2.3 * \sqrt{2*M_{yRk}*f_{hk}*d} +
                        \frac{F_{axRk}}{4} = ''' + str(F_vRk3) + ' kN')
523                 st.latex(r'''  F_{vRk} = min( ''' + rf'''{F_vRk1}, ''' +
524                         rf'''{F_vRk2}, ''' + rf'''{F_vRk3}) = ''' +
                            str(F_vRk) + ' kN')
525
526             # dünnes Stahlblech
527             elif t_Blech < d:
528
529                 F_vk1 = 0.4*f_hk*t_1*d
530                 F_vk2 = 1.15*sqrt(2*M_yrk*f_hk*d)+F_axrk/4
531                 F_vrk = min(F_vk1, F_vk2)
532
533                 F_vRk1 = round(F_vk1/1000, 2)
534                 F_vRk2 = round(F_vk2/1000, 2)
535                 F_vRk = round(F_vrk/1000, 2)
536
537                 st.write(
538                     'Einschnittige Stahl-Holz Verbindung mit einem
                        dünnen Stahlblech.')
539                 st.write('DIN EN 1995-1-1 Abs. 8.2.3 (8.9)')
540                 st.latex(
541                     r''' F_{vk1} = 0.4 * f_{hk}*t_{1}*d =  ''' +
                        str(F_vRk1) + ' kN')
542                 st.latex(
543                     r''' F_{vk2} = 1.15 * \sqrt{2*M_{yRk}*f_{hk}*d} +
                        \frac{F_{axRk}}{4} = ''' + str(F_vRk2) + ' kN')
544                 st.latex(r'''  F_{vRk} = min( ''' + rf'''{F_vRk1}, ''' +
545                         rf'''{F_vRk2}) = ''' + str(F_vRk) + ' kN')
546
547             #__________Nachweise__________
548
549             st.subheader('Nachweise')
550             if nachweisführung == 'Bemessungswerte':
551
552                 # Nachweisführung: Bemessungswerte
553                 col1, col2 = st.columns(2, gap="small")
554                 with col1:
555                     st.latex('Herausziehen')
556                     st.latex(r'''F_{axEd} = ''' + str(round(F_t, 2)) + '
                        kN')
557                     st.latex(r'''F_{axRd} =
                        \frac{F_{axRk}*k_{mod}}{\gamma_{m}} = ''' +
558                             str(round(F_axRk*k_mod/(gamma), 2)) + ' kN')
559                     st.latex(r'''\eta_{axd} = \frac{F_{axEd}}{F_{axRd}}
                        = ''' +
560                             str(round(F_t*gamma/(F_axRk*k_mod), 2)))
561                 with col2:
562                     st.latex('Abscheren')
563                     st.latex(r'''F_{vEd} = ''' + str(round(F_v, 2)) + '
                        kN')
564                     st.latex(r'''F_{vRd} =
                        \frac{F_{vRk}*k_{mod}}{\gamma_{m}} = ''' +
565                             str(round(F_vRk*k_mod/(gamma), 2)) + ' kN')
566                     st.latex(r'''\eta_{vd} = \frac{F_{vEd}}{F_{vRd}} =
                        ''' +
567                             str(round(F_v*gamma/(F_vRk*k_mod), 2)))
```

```python

        # Nachweisführung: charakteristisch
        else:
            col1, col2 = st.columns(2, gap="small")
            with col1:
                st.latex('Herausziehen')
                st.latex(r'''F_{axEk} = ''' + str(round(F_t, 2)) + ' kN')
                st.latex(r'''F_{axRk} = ''' + str(F_axRk) + ' kN')
                st.latex(
                    r'''\eta_{axk} = \frac{F_{axEk}}{F_{axRk}} = '''
                    + str(round(F_t/(F_axRk), 2)))
            with col2:
                st.latex('Abscheren')
                st.latex(r'''F_{vEk} = ''' + str(round(F_v, 2)) + ' kN')
                st.latex(r'''F_{vRk} = ''' + str(F_vRk) + ' kN')
                st.latex(
                    r'''\eta_{vk} = \frac{F_{vEk}}{F_{vRk}} = ''' +
                    str(round(F_v/(F_vRk), 2)))
```