

DSM-basierte Lehrsoftware für Schnittkräfte und Einflusslinien in MATLAB

Autor: Thurga Kirupakaran

Bachelor-Arbeit

Januar 2023

DSM-basierte Lehrsoftware für Schnittkräfte und Einflusslinien in MATLAB

Autor: Thurga Kirupakaran
kthurga@student.ethz.ch

Supervisors

Prof. Dr. Eleni Chatzi
Dr. A. Egger
P. Sieber

Januar 2023

Zusammenfassung

Dieser Bericht befasst sich mit den Überlegungen, der Theorie und den Implementierungen der Lehrsoftware, die entwickelt wird, um Schnittkräfte, Auflagerreaktionen sowie Einflusslinien auszugeben. Die Software soll ermöglichen eigene Übungsaufgaben in diesen Gebieten zu generieren, wobei die Systeme eigenhändig konstruiert werden können. Des Weiteren bietet es einen Einblick in die Implementierung der direkten Steifigkeitsmethode.

Schlagworte

DSM, Schnittkräfte, Auflagerreaktionen, Einflusslinie, MATLAB

Zitierungsvorschlag

<https://github.com/ETH-IBK-SMECH/DSM-und-Einflusslinien>

Inhalt

Abbildungen	3
1 Einleitung	4
2 Methodik	6
2.1 Plattform	6
2.2 MatrizenStatik	6
2.3 Input	8
2.3.1 Input einlesen	8
2.3.2 Input überprüfen	10
2.3.3 Input umwandeln	10
2.4 Rechnen	10
2.4.1 DSM	10
2.4.2 Einbindung Einflusslinien	13
2.5 Output	15
3 Implementierung	17
3.1 Model.Input	17
3.2 Model.Analyse	20
3.2.1 Funktion ModelFuerEinflusslinie	20
3.2.2 Funktion DirectStiffnessMethod	21
3.2.3 activeStabDOF	23
3.2.4 Implementierung der Teilsysteme	25
3.3 Model.Output	26
3.3.1 Schnittkräfte	26
3.3.2 Verformungslinie	27
3.3.3 Zeichnen	28
4 Numerische Beispiele	29
4.1 Schnittkräfte	29
4.2 Auflagerreaktionen	31
4.3 Einflusslinien	33
5 Schlusswort und Ausblick	36
6 Literatur	37
7 Quellenverzeichnis	37

A Idee der GUI-Vorlage	38
B Herleitung der Verdrehungen am Momentengelenk	39
C Ausschnitte aus InputFile	42
D Eigenständigkeitserklärung	45

Abbildungen

1 Hauptfunktion MatrizenStatik	7
2 Idee der GUI-Vorlage für den Input	9
3 MPC für die neuen Enden	14
4 Funktion ModelVonInputFile	17
5 Umwandlung der Lager in SPC	18
6 Eingabe der gesuchten Einflusslinie	19
7 Input umwandeln für Einflusslinie einer Schnittgrösse direkt neben Knoten . .	19
8 Ausschnitt aus Funktion ModelFuerEinflusslinie	20
9 Ausschnitt aus ModelFuerEinflusslinie	21
10 Einbindung von MPC in DSM mit Lagrange Multipliers	21
11 Ausschnitt Funktion getF	22
12 Funktion VerdrehungMomentengelenk	23
13 Mit Software erzeugte Struktur	24
14 Implementierung der activeStabDOF	25
15 Funktion getKnotesTS	25
16 Funktion kondensiereFTS	26
17 Berechnung der Querkraft für die kritischen Punkte	27
18 links: statisch bestimmt; rechts: statisch unbestimmt	28
19 Tragstruktur	29
20 Schnittkraftdiagramme des oberen Systems	29
21 links: Ausschnitt aus Kolloquiumsunterlagen; rechts: von MATLAB erzeugt . .	30
22 Schnittkraftdiagramme des oberen Systems	30
23 Ausschnitt aus Prüfung FS22 des Kurses Baustatik II	31
24 Auflagerreaktionen des oberen Systems	31
25 Auflagerreaktionen des Systems aus Kolloquium 4	32
26 Beispiel 1	33
27 Beispiel 2	34
28 Beispiel 3 (Die Feder ist sehr steif)	35
29 Beispiel 4	35
30 Am Anfang wird die Bedienung der Software erklärt	42
31 Im ersten Teil wird das System konstruiert	42
32 Eingabe der Lager sowie dessen Möglichkeiten	43
33 Im zweiten Teil wird der Output gewählt	43
34 Im dritten Teil können die Lasten eingegeben werden	43
35 Im vierten Teil kann die Einflusslinie gewählt werden	44

1 Einleitung

Computer und Software sind heutzutage fast nicht mehr wegzudenken im Leben und der Lehre. Die Digitalisierung schreitet voran und dies auch an der ETH Zürich. Im Bauingenieurwesen steht die Statik deutlich im Vordergrund. Für komplexere Tragwerke ist es jedoch kaum möglich, das Tragverhalten von Hand auszurechnen. Hier kommen Statiksoftwares ins Spiel, wovon bereits einige professionelle existieren, wie zum Beispiel Cubus¹. Für die Verwendung solcher Softwares ist jedoch ein solides Wissen von Statik erforderlich. Das Verständnis, wie Systeme aufgebaut sind, modelliert werden können und wie der Kräftefluss ungefähr aussehen kann, muss vorhanden sein, um diese Programme sinngemäss verwenden zu können. Einer der Kurse, welcher dieses Verständnis vermittelt sowie auch die mathematischen Ansätze erklärt, ist der Kurs "Baustatik II".²

In der ersten Hälfte dieses Kurses wird die Verformungsmethode und die direkte Steifigkeitsmethode zum Bestimmen von Schnittkräften und Verschiebungen behandelt. In der zweiten Hälfte wird auf Einflusslinien und die Traglast von Systemen eingegangen. Mithilfe von Kolloquien und Hausübungen wird das Verständnis dieser Themen vertieft, überprüft und angewandt. Eine Möglichkeit weitere Hilfsmittel für den Unterricht zu generieren, bieten Softwares, in welchen die Studierenden ihre eigenen Systeme bilden und berechnen können. Am Lehrstuhl für 'Strukturmechanik und Monitoring' des Instituts für Baustatik und Konstruktion der ETH Zürich wurden bereits im Rahmen von Bachelor-Arbeiten mehrere Softwares entwickelt für den Unterricht³, jedoch bis jetzt nach Kenntnissen des Autors noch keine für Einflusslinien. Auch wird das Thema von Teilsystemen noch in keinem dieser Programme behandelt. Für die Studierenden ist es von Vorteil wenn sie die Softwareumgebung bereits kennen und bestenfalls schon verwenden, wie zum Beispiel MATLAB.

Daher wird in dieser Arbeit eine Lehrsoftware für den Kurs Baustatik II in MATLAB⁴ entwickelt, mit welcher Schnittkräfte, Auflagerreaktionen sowie auch Einflusslinien bestimmt werden können. Die Studierenden können die Tragstrukturen dabei selber kreieren. Nebst Punkt- und Linienverteilten Lasten können an den Strukturen zudem Federn und vorgeschrriebene Verschiebungen angebracht werden. Des Weiteren besteht an den Knoten die Möglichkeit verschiedene Lager zu setzen. Für die Stäbe können selber neue Querschnitte

¹Cubus AG: "Cubus: Software für Bauingenieure" unter https://www.cubus-software.com/Guests/Home/d_main.html [Stand: 29.1.2023]

²dies bezieht sich auf die Vorlesung "Baustatik II" der ETH Zürich von Prof. Dr. Eleni Chatzi

³diese Softwares sind auf der folgenden Webseite zu finden: <https://chatzi.ibk.ethz.ch/education/baustatik/vorlesungen.html> [Stand:29.1.2023]

⁴MATLAB Release R2022a

generiert werden mit den gewünschten Eigenschaften. Dabei wird an den Stabenden jedes Stabendgelenk ermöglicht, das heisst es sind Normalkraftgelenke und Querkraftgelenke möglich, nebst einem Momentengelenk und keinem Gelenk. Durch diese Optionen kann jedes System, welches im Bereich Verformungsmethode und DSM sowie Einflusslinien im Kurs “Baustatik II” behandelt wird, rekonstruiert werden. Zusätzlich bietet die Software die Möglichkeit einzelne Stäbe zu Teilsystemen zusammenzufügen, um das Verständnis dieses Konzeptes zu stärken. Durch diesen Schritt wird das System vereinfacht und es kann mit einer kleineren Steifigkeitsmatrix gerechnet werden.

Der Input wird über ein Inputfile eingelesen. Die Idee ist jedoch, dass dieses später von einem Graphical User Interface (GUI) ersetzt wird, da dies die Bedienung für die Studierenden wesentlich vereinfachen würde. Jedoch sprengt die Entwicklung eines GUI den Rahmen dieser Arbeit, in welcher der Fokus auf das korrekte Rechnen gelegt wird. Schlussendlich müssen die Resultate richtig verarbeitet und übersichtlich dargestellt werden. Die optische Darstellung der Resultate ist auch kein Schwerpunkt der Arbeit. Trotzdem wurde versucht, die Resultate so gut wie möglich darzustellen, um eine visuelle Überprüfung der Resultate zu ermöglichen. Damit interessierte Studierende sich mit dem informatischen sowie auch rechnerischen Aspekt dieser Software auseinander setzen können, wird das Ganze als Opensource zur Verfügung gestellt.

Das Programm bietet den Studierenden ein Tool eigene Aufgaben zu kreieren, welche sie selbstständig überprüfen können. Es soll auch helfen Einflusslinien besser zu verstehen, da dies doch ein Thema ist, mit welchem viele Studierende Schwierigkeiten haben. Am besten kann das Verständnis dafür verbessert werden, indem verschiedenste Beispiele betrachtet werden, denn dadurch sind immer wie mehr Zusammenhänge erkennbar. Die Berechnung der Schnittkräfte wird in der Software auch integriert um möglichst alle Themen des Kurses abzudecken und da Schnittkräfte sowie Einflusslinien mit der DSM implementiert werden können. Die Einbindung der Traglast ist eine Frage für weiterführende Arbeiten.

Diese Arbeit ist in drei Teile gegliedert. Zuerst wird auf die benötigten Methoden sowie der Theorie dieser eingegangen im Kapitel Methodik. Danach werden die Implementierungen im gleichnamigen Kapitel dargelegt und erklärt. Zum Schluss sind numerische Beispiele abgebildet, welche durch die entwickelte Software in MATLAB erzeugt wurden.

2 Methodik

2.1 Plattform

Um die passende Programmierumgebung zu finden, wurde der Hauptfokus auf zwei grundlegende Aspekte gelegt. Zum Einen der Entwicklung der Software und zum Anderen der Verwendung dessen. Für beide Aspekte ist die Zugänglichkeit ausschlaggebend. Das heisst, jeder der die Software nutzen will sowie auch jeder, der die Software weiterentwickeln möchte, muss problemlos auf das Programm Zugriff haben. Da beide Aspekte vor allem die Studierenden der ETH Zürich betreffen, fiel die Entscheidung auf MATLAB. MATLAB weist eine überaus breite Dokumentation sowie auch eine gute Versionsverwaltung auf. Ausserdem ist es ideal für das Rechnen mit Matrizen. Damit problemlos Änderungen vorgenommen werden und zudem mehrere Personen gleichzeitig am Code arbeiten können, wird mit GitHub gearbeitet. GitHub bietet unter anderem eine gute Möglichkeit den Code den Studierenden zur Verfügung zu stellen, während dieser laufend verbessert werden kann.

2.2 MatrizenStatik

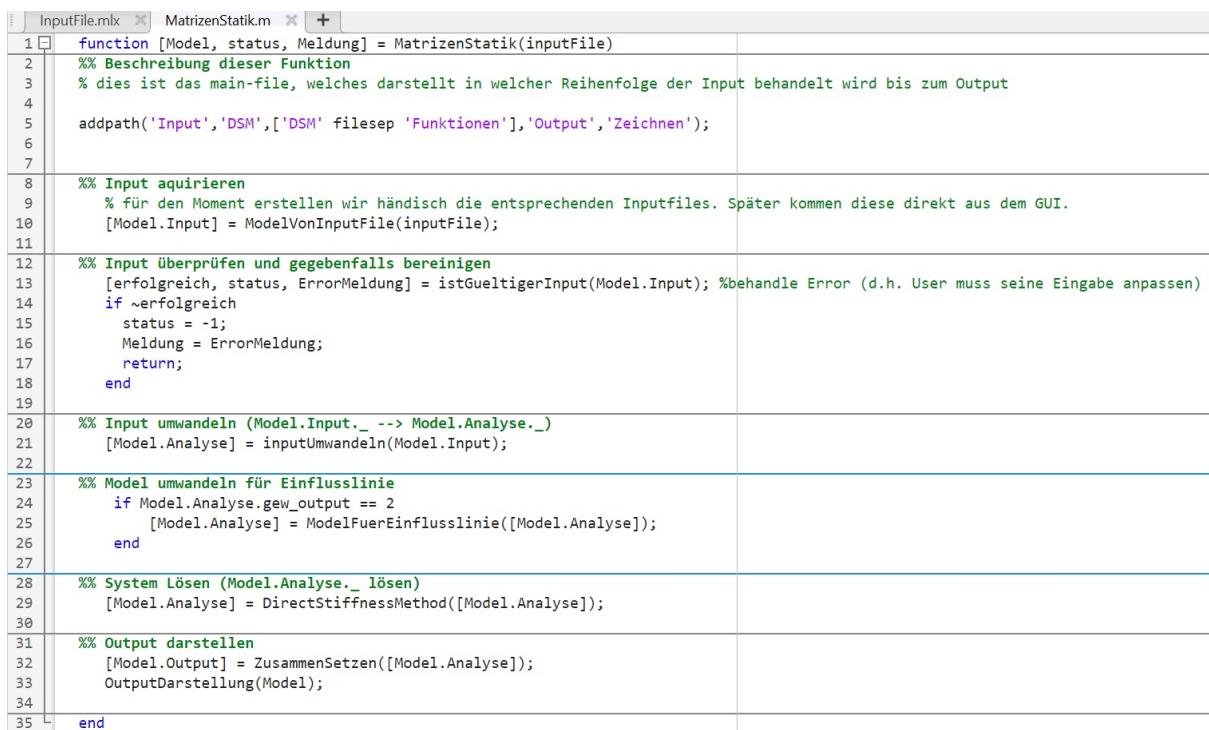
In dieser Arbeit geht es neben dem rechnerischen Aspekt auch darum das Thema der Softwareentwicklung für den Baustatik II Kurs zu vereinheitlichen, damit dies nicht wiederholt neu aufgegleist werden muss. Damit auch Personen, welche nicht von Beginn an an der Entwicklung beteiligt waren, aber diese jedoch verbessern und weiterentwickeln möchten, sich einfach einarbeiten können, muss es strukturiert aufgebaut sein und am besten klare Schnittstellen zu einzelnen Aspekten aufweisen. Daher ist das Programm in drei Teile gegliedert und zwar Input, Analyse und Output.

Um diesen klaren Aufbau zu ermöglichen, wird mithilfe von sogenannten “structs” gearbeitet. Dieser Datentyp ermöglicht einem verschiedene Werte respektive Daten in Unterkategorien zu speichern, den sogenannten “fields”.⁵ Die fields eines structs können verschiedene Datentypen aufweisen, so kann dies auch wiederum vom Typ struct sein. Auch ist es sehr simpel ein struct-Element zu indizieren. Dies erfolgt mit einer Punktnotation. Diese Eigenschaften bieten ein mächtiges Tool, um alle relevanten Daten, wie zum Beispiel den eingelesenen Input, Zwischenschritte der DSM oder auch Endresultate, wie

⁵struct/structure arrays in MATLAB: <https://ch.mathworks.com/help/matlab/ref/struct.html>
[Stand 28.1.2023]

Stabendkräfte, zu speichern und später darauf zuzugreifen. So können Zwischenresultate besser überprüft und dadurch auch Fehler schneller gefunden werden.

In der Software wird das main-struct “Model” gebildet. Zuerst wird der Input eingelesen und unter Model.Input gespeichert, wie in Zeile 10 in Abb. 1 zu erkennen ist. Danach wird dieser Input überprüft und ins Model.Analyse umgewandelt. Je nach gewünschtem Output wird das Model.Analyse entsprechend angepasst, mit welchem danach auch effektiv gerechnet wird. Sobald die Rechnungen durchgeführt wurden, das heisst die DirectStiffnessMethod (DSM) in Zeile 29 der Abb. 1 abgeschlossen ist, werden die Resultate dem gewünschten Output entsprechend im Model.Output verarbeitet und mithilfe der Funktion OutputDarstellung auch dargestellt.



```

1 function [Model, status, Meldung] = MatrizenStatik(inputFile)
2 % Beschreibung dieser Funktion
3 % dies ist das main-file, welches darstellt in welcher Reihenfolge der Input behandelt wird bis zum Output
4
5 addpath('Input','DSM',[ 'DSM' filesep 'Funktionen'], 'Output', 'Zeichnen');
6
7
8 %% Input aquirieren
9 % für den Moment erstellen wir händisch die entsprechenden Inputfiles. Später kommen diese direkt aus dem GUI.
10 [Model.Input] = ModelVonInputFile(inputFile);
11
12 %% Input überprüfen und gegebenfalls bereinigen
13 [erfolgreich, status, ErrorMeldung] = istGültigerInput(Model.Input); % behandle Error (d.h. User muss seine Eingabe anpassen)
14 if ~erfolgreich
15     status = -1;
16     Meldung = ErrorMeldung;
17     return;
18 end
19
20 %% Input umwandeln (Model.Input._ --> Model.Analyse._)
21 [Model.Analyse] = inputUmwandeln(Model.Input);
22
23 %% Model umwandeln für Einflusslinie
24 if Model.Analyse.gew_output == 2
25     [Model.Analyse] = ModelFuerEinflusslinie([Model.Analyse]);
26 end
27
28 %% System Lösen (Model.Analyse._ lösen)
29 [Model.Analyse] = DirectStiffnessMethod([Model.Analyse]);
30
31 %% Output darstellen
32 [Model.Output] = ZusammenSetzen([Model.Analyse]);
33 OutputDarstellung(Model);
34
35 end

```

Abbildung 1: Hauptfunktion MatrizenStatik

Die Abbildung 1 stellt das main-file des Programms dar. Der beschriebene Aufbau ermöglicht eine getrennte Entwicklung der einzelnen Aspekte ohne ein vertieftes Verständnis der anderen Aspekte aufweisen zu müssen. So kann ein GUI für das Einlesen des Inputs getrennt vom rechnerischen Teil entwickelt werden, solang die benötigten Daten geliefert werden. Auch kann sich mit der Darstellung des Outputs separat befasst werden.

2.3 Input

2.3.1 Input einlesen

Um rechnen zu können wird zuerst der Input benötigt. Das heisst, das System muss inklusive den angreifenden Lasten kreiert werden und der gewünschte Output definiert. Dabei sind in dieser Software folgende Outputs möglich:

- Schnittkräfte
- Auflagerreaktionen
- Einflusslinie

Für die Erstellung des Systems sind folgende Eingabemöglichkeiten gegeben, welche auch in Abb. 2 ersichtlich sind:

- Knoten (x- und y-Koordinate)
- Stäbe (Anfangs- und Endknoten, Querschnitt)
- Querschnitte (E, A, I, Stabanzfangs- und Stabendgelenk)⁶
- Lager (siehe Mitte rechts Abb. 2)
- Feder (Knoten, Richtung, Steifigkeit)
- Vorgeschriebene Verschiebungen (Knoten, Richtung, Wert)
- Teilsysteme
- Lasten
 - Knotenlasten (Knoten, Richtung, Wert)
 - konzentrierte Stablasten (Stab, Stelle, Wert)
 - verteilte Stablasten (Stab, Stelle von Anfang sowie Ende, Wert)
- Einflusslinie (Typ)
 - Lager (Knoten, Richtung)
 - Schnittgrösse (Stab, Stelle)

Für die Stabendgelenke, welche unter Querschnitte definiert werden kann, ist jedes Stabendgelenk möglich. Das heisst ein Stab kann ein Normalkraftgelenk, Querkraftgelenk, Momentengelenk oder kein Gelenk an den Stabenden aufweisen. Für die Lager sind die aufgeführten Optionen aus Abbildung 2 möglich. Schräg verschiebbliche Lager sind nicht implementiert. Jedoch kann für ein schräges Rolllager zum Beispiel ein sehr langer dehnstarrer Pendelstab erstellt werden. Dabei kommt das eine Ende an der Stelle des Rolllagers

⁶E steht für das E-Modul, A für die Fläche und I für das Trägheitsmoment

zu liegen und das andere in einer weiten Distanz senkrecht zur Ebene des Rollagers und dies unverschieblich aufgelagert. Für den Typ der Einflusslinie kann gewählt werden, ob die Einflusslinie für eine Normalkraft, einer Querkraft, einem Biegemoment oder einer Lagerreaktion gesucht wird. Für die Erstellung von Teilsystemen kann jeweils eingegeben werden, welche Stäbe zusammen ein Teilsystem bilden.

Wie in Abb. 2 zu erkennen ist, liegt eine klare Idee vor, welcher Input benötigt wird und wie das GUI aussehen könnte. Im Anhang A ist die Abbildung grösser dargestellt. In dieser Arbeit wird ein geführtes Inputfile als mlx-Datei erstellt⁷, in welchem genau beschrieben wird wie der Input eingegeben werden muss, damit die Studierenden keine Probleme damit haben bzgl. der Eingabe ihres Systems. Zusätzlich stehen fünf verschiedene Beispiele zur Verfügung, die dabei helfen das Programm respektive den Input besser zu verstehen.

Anzahl Knoten: <input type="button" value="▼"/>	<input type="button" value="▼"/>																												
Einheiten: <input type="button" value="▼"/> Ⓛ																													
<table border="1"> <thead> <tr> <th></th> <th>x</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>Knoten 1</td> <td></td> <td></td> </tr> <tr> <td>Knoten 2</td> <td></td> <td></td> </tr> </tbody> </table>		x	y	Knoten 1			Knoten 2			<input type="button" value="④"/> → symbolisch [-]																			
	x	y																											
Knoten 1																													
Knoten 2																													
	<input type="button" value="→ numerisch [m]"/>																												
	<input type="button" value="→ numerisch [mm]"/>																												
	<input type="button" value="→ numerisch [Eenm]"/>																												
Anzahl Stäbe: <input type="button" value="▼"/>																													
<table border="1"> <thead> <tr> <th></th> <th>Startknoten</th> <th>Endknoten</th> <th>Querschnitt</th> </tr> </thead> <tbody> <tr> <td>Stab 1</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Stab 2</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>				Startknoten	Endknoten	Querschnitt	Stab 1				Stab 2																		
	Startknoten	Endknoten	Querschnitt																										
Stab 1																													
Stab 2																													
Anzahl Teilsysteme: <input type="button" value="▼"/>																													
<table border="1"> <thead> <tr> <th></th> <th>Beteiligte Stäbe</th> </tr> </thead> <tbody> <tr> <td>Teilsystem 1</td> <td>[1, 2, 5]</td> </tr> <tr> <td>Teilsystem 2</td> <td>[3, 4]</td> </tr> </tbody> </table>				Beteiligte Stäbe	Teilsystem 1	[1, 2, 5]	Teilsystem 2	[3, 4]																					
	Beteiligte Stäbe																												
Teilsystem 1	[1, 2, 5]																												
Teilsystem 2	[3, 4]																												
<input type="button" value="Einheiten: ▼"/> → symbolisch → numerisch																													
<input type="button" value="④"/> → ■ ohne																													
→ [— N]																													
→ — ✓																													
→ ○— M																													
Anzahl Lager: <input type="button" value="▼"/>																													
<table border="1"> <thead> <tr> <th>Lager</th> <th>Knoten</th> <th>↑</th> <th>↗</th> <th>↖</th> <th>↙</th> <th>↘</th> <th>↙</th> <th>↗</th> </tr> </thead> <tbody> <tr> <td>Lager 1</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> <tr> <td>Lager 2</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> </tbody> </table>			Lager	Knoten	↑	↗	↖	↙	↘	↙	↗	Lager 1		○	○	○	○	○	○	○	Lager 2		○	○	○	○	○	○	○
Lager	Knoten	↑	↗	↖	↙	↘	↙	↗																					
Lager 1		○	○	○	○	○	○	○																					
Lager 2		○	○	○	○	○	○	○																					
Anzahl Federn: <input type="button" value="▼"/>																													
<table border="1"> <thead> <tr> <th>Feder</th> <th>Knoten</th> <th>■■■</th> <th>■■</th> <th>■○</th> <th>○○</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Feder 1</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>Feder 2</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>			Feder	Knoten	■■■	■■	■○	○○	Wert	Feder 1		○	○	○			Feder 2		○	○	○								
Feder	Knoten	■■■	■■	■○	○○	Wert																							
Feder 1		○	○	○																									
Feder 2		○	○	○																									
Anzahl vorgeschriebene Verschiebungen: <input type="button" value="▼"/>																													
<input type="button" value="Einheiten: ▼"/> Ⓛ																													
<table border="1"> <thead> <tr> <th>Aufgezwungene Verschiebung</th> <th>Knoten</th> <th>Richtung</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Aufgezwungene Verschiebung 1</td> <td></td> <td>④</td> <td></td> </tr> <tr> <td>Aufgezwungene Verschiebung 2</td> <td></td> <td>④</td> <td></td> </tr> </tbody> </table>			Aufgezwungene Verschiebung	Knoten	Richtung	Wert	Aufgezwungene Verschiebung 1		④		Aufgezwungene Verschiebung 2		④																
Aufgezwungene Verschiebung	Knoten	Richtung	Wert																										
Aufgezwungene Verschiebung 1		④																											
Aufgezwungene Verschiebung 2		④																											
<input type="button" value="④"/> → in x $\frac{dx}{dt}$																													
→ in y $\frac{dy}{dt}$																													
→ in z $\frac{dz}{dt}$																													
Anzahl Knotenlasten: <input type="button" value="▼"/>																													
<input type="button" value="Einheiten: ▼"/> → symbolisch [kg] ④ → numerisch [N/mm] → numerisch [N/mm²]																													
<table border="1"> <thead> <tr> <th>Knotenlast</th> <th>Stab</th> <th>Richtung</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Knotenlast 1</td> <td></td> <td>④</td> <td></td> </tr> <tr> <td>Knotenlast 2</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			Knotenlast	Stab	Richtung	Wert	Knotenlast 1		④		Knotenlast 2																		
Knotenlast	Stab	Richtung	Wert																										
Knotenlast 1		④																											
Knotenlast 2																													
Anzahl konzentrierte Elementlasten: <input type="button" value="▼"/>																													
<input type="button" value="Einheiten: ▼"/> Ⓛ																													
<table border="1"> <thead> <tr> <th>Stablast</th> <th>Stab</th> <th>Richtung</th> <th>Wert</th> <th>Start</th> <th>Ende</th> </tr> </thead> <tbody> <tr> <td>Stablast_k 1</td> <td></td> <td>④</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Stablast_k 2</td> <td></td> <td>④</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			Stablast	Stab	Richtung	Wert	Start	Ende	Stablast_k 1		④				Stablast_k 2		④												
Stablast	Stab	Richtung	Wert	Start	Ende																								
Stablast_k 1		④																											
Stablast_k 2		④																											
Anzahl verteilte Elementlasten: <input type="button" value="▼"/>																													
<input type="button" value="Einheiten: ▼"/> Ⓛ																													
<table border="1"> <thead> <tr> <th>Stablast_v 1</th> <th>Stab</th> <th>Richtung</th> <th>Wert</th> <th>Start</th> <th>Ende</th> </tr> </thead> <tbody> <tr> <td>Stablast_v 2</td> <td></td> <td>④</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			Stablast_v 1	Stab	Richtung	Wert	Start	Ende	Stablast_v 2		④																		
Stablast_v 1	Stab	Richtung	Wert	Start	Ende																								
Stablast_v 2		④																											

Abbildung 2: Idee der GUI-Vorlage für den Input

⁷dies ist ein MATLAB Live-Script, welches einem ermöglicht Text sowie auch Code in einem Dokument zu kombinieren

2.3.2 Input überprüfen

Sobald der Input eingelesen wurde, wird dieser zuerst überprüft bevor gerechnet wird, um Fehler so früh wie möglich abzufangen. Dies geschieht mit der Funktion istGueltigerInput(), welches das Programm beendet und eine Fehlermeldung ausgibt, falls ein Fehler in der Eingabe entdeckt wurde. Es existieren einfach abzufangende Fehler, welche meistens Flüchtigkeitsfehler sind, wie wenn Stäbe oder Knoten gewählt werden, die gar nicht existieren, oder Stäbe der Länge null kreiert werden. Auch kann es passieren das an einem Knoten zwei verschiedene Lager angebracht werden oder die Einflusslinie für eine Lagerreaktion gesucht wird, welche gar nicht existiert. Jedoch gibt es auch schwer zu erfassende Fehler, wie wenn ein instabiles System kreiert wird. Einfachere Fälle von instabilen Systemen, wie zum Beispiel ein Stab mit einem Querkraftgelenk an beiden Enden, können zwar abgefangen werden, Mechanismen hingegen sind schwieriger zu erfassen. Diese Funktion kann laufend erweitert werden.

2.3.3 Input umwandeln

Nachdem nun der Input sauber ist, wird dies in das Model.Analyse umgewandelt. Dieser Schritt soll das Rechnen vereinfachen. So werden die einzelnen Querschnitte den jeweiligen Stäben zugeordnet, damit alle Eigenschaften eines Stabes in einem struct gespeichert sind. In der Eingabe wird die Erstellung des Stabes sowie eines Querschnittes separat behandelt, da dies ermöglicht mehreren Stäben die gleichen Querschnittseigenschaften zuzuweisen ohne diese immer neu definieren zu müssen. Auch werden die Lager in Single Point Constraints (SPC) umgewandelt, also in vorgeschriebene Verschiebungen. Konzentrierte und verteilte Stablasten werden zu einem struct ‘StabLasten’ zusammengefügt.

2.4 Rechnen

2.4.1 DSM

Mit dem Model.Analyse wird nun gerechnet. Falls der gewünschte Output Einflusslinien sind, wird dieses Modell zuerst angepasst. Auf dies wird im Abschnitt 2.4.2 näher eingegangen. Die direkte Steifigkeitsmethode (DSM) wird nach dem klassischen Verfahren implementiert, welches grob wie folgt aussieht:

1. die Steifigkeitsmatrizen für die einzelnen Stäbe aufstellen im lokalen sowie auch im globalen Koordinatensystem
2. die aktiven Freiheitsgrade, die sogenannten “activeStabDOF” im Programm, bestimmen und den Stäben entsprechend zuweisen
3. den internen Kraftvektor P_{int} , welche durch die angreifenden Stablasten erzeugt wird, für jeden Stab bestimmen und ins globale Koordinatensystem rotieren zu $F_{sys,Stab}$
4. die globale Systemsteifigkeitsmatrix K_{sys} aufstellen sowie auch den globalen Systemlastvektor F_{sys} wobei für diesen gilt $F_{sys} = F_{sys,Knoten} - F_{sys,Stab}$
5. den globalen Verschiebungsvektor u_{sys} ausrechnen mit $u_{sys} = K_{sys}^{-1} \cdot F_{sys}$
6. den einzelnen Stäben die entsprechenden globalen Verschiebungen zuweisen und diese ins lokale Koordinatensystem umwandeln zu u_{loc}
7. die Stabendkräfte q_{loc} berechnen mit $q_{loc} = K_{loc} \cdot u_{loc} + P_{int}$

Um die Stablasten in interne Stabendkräfte umzuwandeln wurden Tabellen, in welcher die Stabendkräfte verschiedener Lastfälle für einen beidseitig eingespannten Balken dargestellt sind, zu Hilfe genommen aus dem Buch “Matrix Analysis of Structures” von A. Kassimali (Kassimali (2021)). Diese Lasten werden entsprechend den Stabendgelenken des Stabes kondensiert und weitergegeben.

Wie am Verfahren zu erkennen ist, liegt am Schluss der DirectStiffnessMethod-Funktion die Stabendverschiebungen, sowie die Stabendkräfte vor. Um die Stabendkräfte in die Schnittkraftkonvention zu wechseln, werden diese jeweils mit den folgenden Werten multipliziert: $[-1; 1; -1; 1; -1; 1]$. Um nun noch die Auflagerreaktionen zu berechnen, wird ein Vektor für die Summe der Kräfte für jeden Freiheitsgrad erstellt. Die globalen Stabendkräfte werden addiert und die Knotenlasten subtrahiert. Nun kann für die Lagerkräfte sowie für die Federn die entsprechende Summe des Freiheitsgrades abgelesen werden, welches auch der Lagerreaktion entspricht.

Um Teilsysteme in die Software einzubinden, wurde das konventionelle Verfahren ergänzt. Zuerst jedoch muss geklärt werden, was genau als Teilsystem gesehen wird. In dieser Arbeit wird die Möglichkeit aneinandergereihte Stäbe zu einem Teilsystem zusammenzufassen implementiert. Dabei werden die Stäbe in der Reihenfolge, wie sie aneinander liegen eingegeben. Die Freiheitsgrade der beiden Enden des so gebildeten Teilsystems stellen jeweils die externen Freiheitsgrade dar und die Freiheitsgrade an den Knoten zwischen diesen beiden Enden die Internen. Für die konkrete Einbindung von Teilsystemen wird in Schritt 2 des vorher beschriebenen Verfahrens mithilfe von Kondensation Steifigkeitsmatrizen für die Teilsysteme erstellt. In Schritt 3 werden die angreifenden Lasten im Teilsystem auf dessen Endknoten kondensiert. Um die Systemsteifigkeitsmatrix

zu erstellen werden jeweils die Steifigkeitsmatrizen der Stäbe, welche nicht zu einem Teilsystem gehören, die Steifigkeitsmatrizen der Teilsysteme sowie die Steifigkeiten der Federn addiert. Nach diesen Schritten wird normal weitergerechnet. Sobald \mathbf{u}_{sys} für die aktiven Freiheitsgrade ausgerechnet wurde, müssen die Verschiebungen der kondensierten Freiheitsgrade zurückgerechnet werden. Dies wurde noch nicht implementiert.

Was auch spezieller ist an der Implementation der DirectStiffnessMethod-Funktion ist, dass mit sehr viel Kondensation auf verschiedensten Stufen gearbeitet wird, um automatisch selbststabile Systeme (im Sinn von Matrizen ohne 0-er Zeilen) zu kreieren. Dabei wird auf folgenden Stufen kondensiert:

- Stabstufe: Grundsätzlich wird für jeden Stab zuerst die Steifigkeitsmatrix für den Fall eines beidseitig eingespannten Balkens aufgestellt. Je nach Stabendgelenk ist der entsprechende Freiheitsgrad nicht vorhanden am Stab. Solche werden als interne Freiheitsgrade (i) angesehen und die vorhandenen als externe (e) betrachtet. Um die Steifigkeitsmatrix lediglich für die aktiven Freiheitsgrade aufzustellen wird eine Kondensation wie folgt durchgeführt:⁸

$$\begin{bmatrix} K_{ee} & K_{ei} \\ K_{ie} & K_{ii} \end{bmatrix} \times \begin{bmatrix} u_e \\ u_i \end{bmatrix} = \begin{bmatrix} f_e \\ f_i \end{bmatrix} \quad (1)$$

$$K_{ee}u_e + K_{ei}u_i = f_e \quad (2)$$

$$K_{ie}u_e + K_{ii}u_i = f_i \quad (3)$$

$$u_i = K_{ii}^{-1}(f_i - K_{ie}u_e) \quad (4)$$

$$K_{ee}u_e + K_{ei} \cdot (K_{ii}^{-1}f_i - K_{ii}^{-1}K_{ie}u_e) = f_e \quad (5)$$

$$(K_{ee} - K_{ei}K_{ii}^{-1}K_{ie}) \cdot u_e = f_e - K_{ei}K_{ii}^{-1}f_i \quad (6)$$

$$\tilde{K}_{ee}u_e = \tilde{f}_e \quad (7)$$

Das heisst die Steifigkeitsmatrix sowie der Lastvektor für die externen (die aktiven) Freiheitsgrade werden folgendermassen ausgerechnet:

$$\tilde{K}_{ee} = K_{ee} - K_{ei}K_{ii}^{-1}K_{ie} \quad (8)$$

$$\tilde{f}_e = f_e - K_{ei}K_{ii}^{-1}f_i \quad (9)$$

In Schritt 1 werden die Steifigkeitsmatrizen für die Stäbe kondensiert mit Formel 8 und in Schritt 3 die Stablasten mit Formel 9.

⁸Das Vorgehen einer Kondensation wurde in der Vorlesung Baustatik II der ETHZ erläutert

- Teilsystemstufe: Das gleiche Prinzip der Kondensation auf Stabstufe wird auf die internen und externen Freiheitsgrade des Teilsystems angewendet.
- Systemstufe: Zwischen Schritt 4 und 5 wird die Systemsteifigkeitsmatrix kondensiert aufgrund der vorgeschriebenen Verschiebungen, welche in interne Freiheitsgrade umgewandelt werden. Da hierbei die internen Verschiebungen bekannt sind, kann eine einfachere Version der Kondensation durchgeführt werden. Dafür wird lediglich die Gleichung 2 folgendermassen umgestellt:

$$K_{ee}u_e = f_e - K_{ei}u_i \quad (10)$$

Damit kann die Systemsteifigkeitsmatrix sowie der Systemlastvektor für die freien (externen) Freiheitsgrade wie folgt aufgestellt werden:

$$\tilde{K}_{ee} = K_{ee} \quad (11)$$

$$\tilde{f}_e = f_e - K_{ei}u_i \quad (12)$$

2.4.2 Einbindung Einflusslinien

Für die Implementierung von Einflusslinien in die Software wird das Model.Analyse zuerst umgewandelt, bevor aktiv gerechnet wird. Zum Bestimmen der Einflusslinie einer Lagerreaktion, wird an der Stelle der Reaktion eine Verschiebung von -1 eingeführt und danach die Verformungslinie infolge dieser Verschiebung gezeichnet. Daher kann dieser Fall als eine Zwängung behandelt werden. Für die Einflusslinie einer Schnittgrösse wird die Bindung dieser Kraft an der entsprechenden Stelle gelöst und an dieser eine Verschiebung von -1 eingeführt. Diese Vorgehensweise kann durch Multiple Point Constraints (MPC) ausgedrückt werden, wie in Abb. 3 ersichtlich ist.⁹

⁹Dies ist ein Ausschnitt aus den für die Arbeit zur Hilfe erstellten Unterlagen von Dr. Adrian Egger

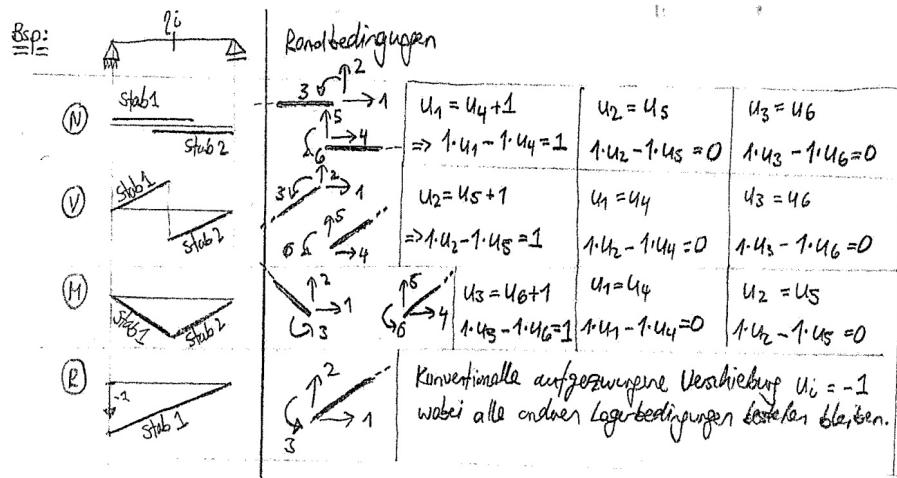


Abbildung 3: MPC für die neuen Enden

Die MPC können mit verschiedenen Methoden behandelt werden, wie der Master-Slave Methode¹⁰, der Penalty-Methode¹¹ oder durch die Lagrange Multiplier Adjunction¹². Die Implementierung der Lagrange Multipliers erweist sich als am einfachsten, da dafür die Steifigkeitsmatrix lediglich um drei Reihen und Spalten ergänzt und der Lastvektor auch nur um drei Einträge ergänzt werden muss, da drei MPC vorliegen für den neuen Doppelknoten. Bei der Lagrange Multiplier Adjunction wird für jeden MPC eine neue Unbekannte λ eingeführt, der sogenannte Lagrange Multiplier. In einer Matrix A wird die linke Handseite der MPC erfasst in Abhängigkeit der Verschiebungen u. Der Lastvektor wird mit den Bindungen, welche auf der rechten Seite der MPC zu finden sind, gebildet. Das neue System sieht folgendermassen aus:¹²

$$\begin{bmatrix} K & A^T \\ A & 0 \end{bmatrix} \times \begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ b \end{bmatrix} \quad (13)$$

Diese Bedingungen führen zu einem neuen Lastvektor, mit welchem die DSM-Funktion die dadurch verursachten Stabendverschiebungen ausrechnet. Falls die Einflusslinie einer Schnittgrösse direkt neben einem Knoten zu liegen kommt, wird ein sehr kleiner Stab erstellt und damit wie mit dem beschriebenden Verfahren gerechnet.

¹⁰Kapitel 8.3, Felippa (2004)

¹¹Kapitel 9.2, Felippa (2004)

¹²Kapitel 9.3, Felippa (2004)

2.5 Output

Nun werden die Endresultate der DSM dem gewünschtem Output nach verarbeitet. Für die Bestimmung der Schnittgrößen werden für jeden Stab die Stabendkräfte und die angreifenden Stablasten betrachtet. Jede Stelle, an welcher eine Last angreift, wird als kritischer Punkt in einem Vektor festgehalten. Die kritischen Punkte repräsentieren die Stellen, an welchen die Schnittgrößen jeweils mithilfe von Integration ausgerechnet und geplottet werden. Dies ist ausreichend für die vorkommenden Lastfälle, da diese, mit Ausnahme des Momentenverlaufs durch eine verteilte Last, immer einen linearen Verlauf der Schnittgrößen erzeugen. Um auch den quadratischen Verlauf des Momentes für die verteilte Last darzustellen, werden über die Spannweite des Lastfalls mehrere kritische Punkte erstellt.

Um die Verformungslinie zu zeichnen, werden die Stabendverschiebungen und sogenannte shape-functions zur Hand genommen. Letzteres wurde dem Buch ‘Matrix Analysis of Structures’ von Kassimali entnommen.¹³ Diese sehen wie folgt aus:

$$N1 = 1 - x/L; \quad (14)$$

$$N2 = 1 - 3x^2/L^2 + 2x^3/L^3; \quad (15)$$

$$N3 = x - 2x^2/L + x^3/L^2; \quad (16)$$

$$N4 = x/L; \quad (17)$$

$$N5 = 3x^2/L^2 - 2x^3/L^3; \quad (18)$$

$$N6 = -x^2/L + x^3/L^2; \quad (19)$$

Um die Verschiebung an einer Stelle x des Stabes zu erhalten, müssen die Werte der shape-functions mit den Stabendverschiebungen folgendermassen multipliziert werden: $\mathbf{N} \cdot \mathbf{u}_{loc}$ wobei $\mathbf{N} = [N1; N2; N3; N4; N5; N6]$ entspricht. Dafür wird zuerst ein Vektor mit verschiedenen Werten von x , für die $0 \leq x \leq L$ gilt, erzeugt. Für jeden x -Wert wird die Verschiebung mit der aufgeführten Gleichung berechnet und in einem neuen Vektor gespeichert. So kann die Verformungslinie für jeden Stab geplottet werden. Damit die shape-functions problemlos angewendet werden können, müssen alle Stabendverschiebungen vorhanden sein, das heisst auch die Verschiebungen der nicht aktiven Freiheitsgrade müssen bestimmt werden. Diese werden mithilfe der Balkengleichung hergeleitet. Die Herleitung ist in Anhang B zu sehen.

¹³Kapitel 5.3, Kassimali (2021)

Sobald die einzelnen Plots für die Stäbe vorhanden sind, müssen diese schlicht zusammengesetzt werden. Dabei ist von Relevanz wie der Stab und der zugehörige Plot rotiert und verschoben werden muss. Damit die Nutzer der Software auch wissen, welches System sie gezeichnet haben, werden Funktionen erstellt, welche folgende Elemente zeichnen:

- gerader Pfeil
- runder Pfeil
- Lager
- Feder
- Gelenk

Für diese Elemente wurden einzeln Funktionen erstellt, welche einen Koordinatenpunkt und einen Skalierungswert als Parameter aufnehmen können. Danach wird hauptsächlich mit der durchschnittlichen Länge der Stäbe skaliert. Für die Lasten wird jeweils noch der durchschnittliche Wert der vorkommenden Lasten mitberücksichtigt. Die Darstellungen dienen der visuellen Überprüfung der Werte und sind auch nutzerfreundlicher als die einzelnen Werte im entsprechenden struct-field nachlesen zu müssen.

3 Implementierung

3.1 Model.Input

Die geführte Inputvorlage ist in vier Teile gegliedert. Zuerst wird das System aufgebaut und danach wird der gewünschte Output bestimmt. Dabei stehen Schnittkräfte, Auflagerreaktionen und Einflusslinien zur Auswahl. Im dritten Teil können die Lasten eingegeben werden, falls die Schnittkräfte oder die Auflagerreaktionen gesucht werden. Wird sich für die Ausgabe von Einflusslinien entschieden, wird der vierte Teil bearbeitet. Ausschnitte des Inputfiles sind in Anhang C zu sehen. Es können mehrere InputFiles generiert werden und mit der in Abb. 4 dargestellten Funktion dem case entsprechend aufgerufen werden. Das heisst, sobald der Input eingegeben wurde, kann im Command Window von MATLAB der Befehl MatrizenStatik(case) aufgerufen werden und das zugehörige System wird berechnet. Auch sind bereits fünf Beispiele hinterlegt.

```

1  function [model] = ModelVonInputFile(inputFile)
2
3     switch inputFile
4         case 0
5             inputFile;
6         case 1
7             inputFileTest1; % Bsp. Schnittkräfte: Matlab-Übung 2
8         case 2
9             inputFileTest2; % Bsp. Einflusslinie: Prüfung FS22 Aufgabe Einflusslinie
10        case 3
11            inputFileTest3; % Bsp. Auflagerreaktionen: System Prüfung FS22 Aufgabe 2 Teil II
12        case 4
13            inputFileTest4; % Bsp. Schnittkräfte: Hausübung 6 Augabe 1
14        case 5
15            inputFileTest5; % Bsp. Einflusslinie: Kolloquium 7.2 Aufgabe 1
16        otherwise
17            disp('InputFile kann nicht gefunden werden.')
18        end
19
20        model = in;
21
22    end

```

Abbildung 4: Funktion ModelVonInputFile

Ist die Inputeingabe beendet und der Befehl MatrizenStatik aufgerufen, so wird der Input wie im Abschnitt 2.3.2 beschrieben überprüft. Wird kein Fehler gefunden, so wird der Input in ein geeigneteres Format zum Rechnen umgewandelt, dem Model.Analyse. Dies geschieht mit der Funktion InputUmwandeln. Die Lager werden folgenderweise in SPC umgewandelt:

```

77     for i=1:nLager
78         switch find(in.Lager.Lagerung(i,:))
79             case 1 %voll eingespannt
80                 nSPC = nSPC + 3;
81             case {2,5,6} %gelenkig gelagert, verschieblich eingespannt
82                 nSPC = nSPC + 2;
83             case {3,4} %Rolllager
84                 nSPC = nSPC + 1;
85         end
86     end
87     out.SPC(nSPC).node = [];
88     out.SPC(nSPC).dir = [];
89     out.SPC(nSPC).val = [];
90
91     Idx = 1 ;
92     for i=1:nLager
93         switch find(in.Lager.Lagerung(i,:))
94             case 1 %voll eingespannt
95                 out.SPC(Idx ).node = in.Lager.Knoten(i);
96                 out.SPC(Idx+1).node = in.Lager.Knoten(i);
97                 out.SPC(Idx+2).node = in.Lager.Knoten(i);
98                 out.SPC(Idx ).dir = 1;
99                 out.SPC(Idx+1).dir = 2;
100                out.SPC(Idx+2).dir = 3;
101                out.SPC(Idx ).val = 0;
102                out.SPC(Idx+1).val = 0;
103                out.SPC(Idx+2).val = 0;
104                Idx = Idx + 3;
105            case 2 %gelenkig gelagert
106                out.SPC(Idx ).node = in.Lager.Knoten(i);
107                out.SPC(Idx+1).node = in.Lager.Knoten(i);
108                out.SPC(Idx ).dir = 1;
109                out.SPC(Idx+1).dir = 2;
110                out.SPC(Idx ).val = 0;
111                out.SPC(Idx+1).val = 0;
112                Idx = Idx + 2;

```

Abbildung 5: Umwandlung der Lager in SPC

Für ein Auflager werden beispielsweise zwei neue SPC erzeugt. Die eine vorgeschriebene Verschiebung ist für die horizontale Verschiebung und die andere für die vertikale Verschiebung mit den Werten gleich null, da sich diese nicht verschieben werden. Für einen SPC wird jeweils der entsprechende Knoten, die Richtung und der Wert gespeichert. Die SPC, welche durch Lager erzeugt werden, werden den bereits beim Systemaufbau definierten vorgeschriebenen Verschiebungen angehängt.

Für die Eingabe der Einflusslinie muss unterschieden werden, ob diese für eine Schnittgrösse oder eine Lagerreaktion gesucht wird, denn je nach dem muss entweder der entsprechende Stab oder der entsprechenen Knoten angegeben werden. Daher werden diese Eingabemöglichkeiten getrennt behandelt wie in Abb. 6 ersichtlich ist. Für die Lagerreaktion wird die Richtung der Reaktion und der Knoten angegeben, hingegen für eine Schnittgrösse der Stab und die Stelle, an welcher diese gesucht ist. Die Stelle wird dabei in Abhängigkeit von L angegeben, das heisst sie kann Werte zwischen 0 und 1 annehmen.

Einflusslinie

12.Art der Einflusslinie

Hier kannst du die Art der Einflusslinie wählen. Wähle die entsprechende Nummer der gewünschten Art:

- 1: Einflusslinie für die **Normalkraft**. Bearbeite **Schritt 14**.
- 2: Einflusslinie für die **Querkraft**. Bearbeite **Schritt 14**.
- 3: Einflusslinie für das **Biegemoment**. Bearbeite **Schritt 14**.
- 4: Einflusslinie für eine **Lagerreaktion**. Bearbeite **Schritt 13**.

```
%AUSFÜLLEN, wenn Einflusslinie gesucht
TypEL = [3]; % 1:N, 2:V, 3:M, 4:Lager
```

13.Einflusslinie für eine Lagerreaktion

Hier wählst du an welchem Knoten sich das Lager befindet. Danach auf welche Richtung sich die Lagerreaktion bezieht. Also 1,2 oder 3.

```
if TypEL == 4
    %AUSFÜLLEN, wenn Einflusslinie für Lagerreaktion gesucht
    Knoten = [2];
    Richtung = [2];

    in.Einflusslinie = table(TypEL, Knoten, Richtung);
```

14.Einflusslinie für eine Schnittgrösse

Hier wählst du an welchem Stab sich deine gewünschte Schnittgrösse befindet. Danach wählst du die **Stelle in Abhängigkeit von L**. Wenn du dich für eine Schnittgrösse direkt neben einem Knoten interessierst, wählst du einfach den Stab, an welcher sich diese Stelle befindet und danach 0 oder 1, je nach dem welchem Ende es entspricht.

```
else
    %AUSFÜLLEN, wenn Einflusslinie für Schnittgrösse gesucht
    Stab = [1];
    Stelle = [0.5]; %zw. 0 <= x <= 1

    in.Einflusslinie = table(TypEL, Stab, Stelle);
end
```

Abbildung 6: Eingabe der gesuchten Einflusslinie

Kommt die Einflusslinie einer Schnittgrösse direkt am Knoten (bzw rechts oder links davon) zu liegen, das heisst wird für die Stelle eine 0 oder 1 gewählt, so wird dieser Wert in der Funktion InputUmwandeln zu 0.001 resp. 0.999 geändert. Dies ist in Abb. 7 dargestellt. Dadurch wird ein sehr kurzer Stab generiert um weiterhin mit MPC arbeiten zu können.

```
185     %Einflusslinie
186     out.gew_output = in.gew_output(1);
187     out.Einflusslinie = struct();
188     out.Einflusslinie = table2struct(in.Einflusslinie);
189
190     if out.gew_output == 2 && out.Einflusslinie.TypEL ~= 4
191         if out.Einflusslinie.Stelle == 0
192             out.Einflusslinie.Stelle = 0.001;
193         elseif out.Einflusslinie.Stelle == 1
194             out.Einflusslinie.Stelle = 0.999;
195         end
196     end
```

Abbildung 7: Input umwandeln für Einflusslinie einer Schnittgrösse direkt neben Knoten

3.2 Model.Analyse

3.2.1 Funktion ModelFuerEinflusslinie

Für das Bestimmen der Schnittkräfte und der Auflagerreaktionen wird das Model.Analyse direkt in die DirectStiffnessMethod-Funktion gegeben. Für den Fall der Einflusslinie wird dieses Modell zuerst angepasst, bevor damit gerechnet wird. Um die Einflusslinie einer Lagerreaktion zu bestimmen, wird eine Verschiebung von -1 in die Richtung der Lagerkraft eingeführt. Daher wird eine vorgeschriebene Verschiebung am Knoten des Lagers in die entsprechende Richtung angebracht und als Lastfall zum Rechnen weitergegeben. Diese Umwandlung des Modells passiert in der Funktion ModelFuerEinflusslinie und ist in Abb. 8 gezeigt.

```

11 %% Änderung des Modells für EL Lager
12
13     if Einflusslinie.TypEL == 4
14
15         SPC(end+1).node = Einflusslinie.Knoten;
16         SPC(end).dir = Einflusslinie.Richtung;
17         SPC(end).val = -1;
18
19         out.SPC = SPC;
20         return
21     end

```

Abbildung 8: Ausschnitt aus Funktion ModelFuerEinflusslinie

Für die Einflusslinie einer Schnittgrösse hingegen wird der Stab an der Stelle der gesuchten Schnittgrösse getrennt und in zwei Stäbe geteilt. An der Schnittstelle werden zwei neue Knoten mit den gleichen Koordinaten erzeugt, damit mit MPC gearbeitet werden kann. Die Implementierung der Trennung eines Stabes in zwei neue Stäbe erfolgt folgendermassen: Es werden zwei neue Stäbe generiert, welche die gleichen Eigenschaften übernehmen wie der ursprüngliche ungetrennte Stab. Für den ersten Stab wird jedoch der Endknoten zum ersten neu erzeugten Knoten geändert und für den zweiten Stab dessen Anfangsknoten zum zweiten neu erzeugten Knoten. Ausserdem wird an beiden Stellen das Stabendgelenk gelöscht, da dies bedeutet, dass kein Gelenk an der jeweiligen Stelle vorhanden ist. In Abb. 9 ist die Implementierung dargestellt. Zum Schluss wird der alte Stab gelöscht und das neue Modell an die DSM-Funktion weitergegeben.

```

24 %% Änderung des Modells für EL
25
26     Idx = Einflusslinie.Stab;
27
28     xS = Knoten(Stab(Idx).sNode).x;
29     yS = Knoten(Stab(Idx).sNode).y;
30     xE = Knoten(Stab(Idx).eNode).x;
31     yE = Knoten(Stab(Idx).eNode).y;
32
33     x_neu = xS + Einflusslinie.Stelle*(xE-xS);
34     y_neu = yS + Einflusslinie.Stelle*(yE-yS);
35
36     Kneu_1 = size(Knoten,2) + 1;
37     Kneu_2 = size(Knoten,2) + 2;
38     Knoten(Kneu_1).x = x_neu;
39     Knoten(Kneu_1).y = y_neu;
40     Knoten(Kneu_2).x = x_neu;
41     Knoten(Kneu_2).y = y_neu;

42     Stab(end+1) = Stab(Idx); %erster neue Teilstab
43     Stab(end).eNode = size(Knoten,2)-1;
44     Stab(end).eRelease = [];
45
46     Stab(end+1) = Stab(Idx);
47     Stab(end).sNode = size(Knoten,2);
48     Stab(end).sRelease = [];
49
50
51
52
53 %% alten Stab löschen
54
55     Stab(Idx) = [];

```

Abbildung 9: Ausschnitt aus ModelFuerEinflusslinie

3.2.2 Funktion DirectStiffnessMethod

Für die Berechnungen der Stabendverschiebungen, welche im Falle der Einflusslinie relevant sind, wird für eine Lagerreaktion nach dem konventionellem Verfahren der DSM vorgegangen. Für Schnittgrößen werden Multiple Point Constraints erzeugt und mithilfe von Lagrange Multipliern eingebunden. Dieses Vorgehen wurde in Abschnitt 2.4.2 beschrieben. Die Steifigkeitsmatrix wird nach Formel 13 erweitert. Der Lastvektor wird auch erweitert dabei wird die ‘rechte’ Seite der MPC betrachtet. Je nach gesuchter Schnittgröße wird an der entsprechenden Stelle eine Bindung von 1 oder -1 im erweiterten Lastvektor angebracht. Die Variation des Vorzeichens hängt mit der Konvention der Richtung von Schnittkräften zusammen. Beide Ergänzungen sind in Abb. 10 zu sehen.

```

317 %%Lagrange Multiplier Adjunction für Einflusslinie
318 if gew_output == 2 && Einflusslinie.TypEL ~=4
319     U_sys(end+3) = 0;
320
321     sizeK = size(K_sys,2);
322     A1 = zeros(3,sizeK-6);
323     A2 = [1,0,0,-1, 0, 0;
324            0,1,0, 0,-1, 0;
325            0,0,1, 0, 0,-1] ;
326     A = [A1,A2];
327
328     F_sys2 = zeros(3,1);
329     F_sys2(Einflusslinie.TypEL) = -1;
330     if Einflusslinie.TypEL == 2; F_sys2(Einflusslinie.TypEL) = 1; end
331
332     F_sys = [F_sys; F_sys2];
333
334     K_sys = [K_sys, A';
335             A, zeros(3)];
336
337     kond.s(end+1:end+3) = false;
338     kond.f(end+1:end+3) = true;
339
340
341
342 %%K_ff \ f_f_kond
343 %f_f_kond = f_f - K_fs*u_s
344 kond.F_sys_f_kond = F_sys(kond.f) - K_sys(kond.f,kond.s)*U_sys(kond.s);
345 kond.K_sys_ff = K_sys(kond.f,kond.f);

```

Abbildung 10: Einbindung von MPC in DSM mit Lagrange Multipliers

Die DSM-Funktion ist nach dem in Abschnitt 2.4.1 beschriebenem Verfahren aufgebaut. Um diese Hauptrechnungsfunktion übersichtlicher zu gestalten und dadurch eine schnellere Nachvollziehbarkeit der Implementierung zu gewährleisten, werden viele kleinere Funktionen für einzelne Teilschritte entwickelt, sowie zum Beispiel der Kondensation eines Teilsystems und dessen Lasten. Oder auch für die Umwandlung der Stablasten in interne Knotenlasten. Dies passiert in der Funktion getF, welche in Abb. 11 abgebildet ist.

```

14     switch StabLast.typ
15
16     case 1
17         l_2 = L - l_1;
18         f_loc = -[W*l_2/L; 0; 0; W*l_1/L; 0; 0];
19
20     case 2
21         l_2 = L - l_1;
22         f_loc = - [0;
23             W*(l_2^2)*(3*l_1 + l_2)/(L^3);
24             W*l_1*(l_2^2)/(L^2);
25             0;
26             W*(l_1^2)*(l_1 + 3*l_2)/(L^3);
27             -W*(l_1^2)*l_2/(L^2)];
28
29     case 3
30         l_2 = L - l_1;
31         f_loc = - [0;
32             -6*W*l_1*l_2/(L^3);
33             W*l_2*(l_2 - 2*l_1)/(L^2);
34             0;
35             6*W*l_1*l_2/(L^3);
36             W*l_1*(l_1 - 2*l_2)/(L^2)];
37
38     case 4
39         l_2 = (1-StabLast.eDist) * L;
40         f_loc = - [W*(L - l_1 - l_2)*(L - l_1 + l_2)/(2*L);
41             0;
42             0;
43             W*(L - l_1 - l_2)*(L + l_1 - l_2)/(2*L);
44             0;
45             0];
46
47     case 5
48         l_2 = (1-StabLast.eDist) * L;
49         f_loc = - [0;
50             (W*L/2)*(1 - l_1*(2*L^3 - 2*(l_1^2)*L + (l_1^3))/(L^4) - (l_2^3)*(2*L - l_2)/(L^4));
51             (W*(L^2)/12)*(1 - (l_1^2)*(6*L^2 - 8*l_1*L + 3*(l_1^2))/(L^4) - (l_2^3)*(4*L - 3*l_2)/(L^4));
52             0;
53             (W*L/2)*(1 - l_2*(2*L^3 - 2*(l_2^2)*L + (l_2^3))/(L^4) - (l_1^3)*(2*L - l_1)/(L^4));
54             - (W*(L^2)/12)*(1 - (l_2^2)*(6*L^2 - 8*l_2*L + 3*(l_2^2))/(L^4) - (l_1^3)*(4*L - 3*l_1)/(L^4))];
```

Abbildung 11: Ausschnitt Funktion getF

Die Werte wurden dem Buch von A. Kassimali (Kassimali (2021)) entnommen. Der Aufbau dieser Funktion ermöglicht es simpel weitere Lastfälle zu inkludieren, wie zum Beispiel dreiecksförmig verteilte Lasten.

Da für das Berechnen der Verformungslinie auch die Stabendverschiebungen der nicht vorhandenen Freiheitsgrade nötig sind, wird am Schluss der DSM-Funktion noch die Funktion VerdrehungMomentengelenk eingeführt, welche die Verdrehungen an einem Momentengelenk ausrechnet. Die Herleitung der gebrauchten Formeln erfolgt mit der Balkengleichung und ist im Anhang B gezeigt. Dabei werden die Fälle eines Pendelstabes,

sowie eines Stabes mit einem Momentengelenk behandelt. Die Funktion ist in Abb. 12 dargestellt.

```

1 function u_loc = VerdrehungMomentengelenk(u_loc, L, vorhandeneDOF)
2
3     if sum(vorhandeneDOF) == 5
4         if ~vorhandeneDOF(3)
5             u2 = -u_loc(2)*3/(2*L);
6             u5 = u_loc(5)*3/(2*L);
7             u6 = -u_loc(6)*1/2;
8             u_loc(3) = u2 + u5 + u6;
9         elseif ~vorhandeneDOF(6)
10            u2 = -u_loc(2)*3/(2*L);
11            u3 = -u_loc(3)*1/2;
12            u5 = u_loc(5)*3/(2*L);
13            u_loc(6) = u2 + u3 + u5;
14        end
15    elseif sum(vorhandeneDOF) == 2
16        if ~vorhandeneDOF(3) && ~vorhandeneDOF(6)
17            u2 = -u_loc(2)*1/L;
18            u5 = u_loc(5)*1/L;
19            u_loc(3) = u2 + u5;
20            u_loc(6) = u2 + u5;
21        end
22    end
23
24 end

```

Abbildung 12: Funktion VerdrehungMomentengelenk

3.2.3 activeStabDOF

In der entwickelten Software wird nur mit den aktiven Freiheitsgraden gerechnet, da dadurch vermieden wird, dass MATLAB mit Matrizen, welche 0-er Zeilen beinhalten, rechnen muss. Diese Freiheitsgrade werden avtiveStabDOF genannt und sind jeweils für jeden Stab als boolean-Vektor gespeichert. Dafür werden zuerst die lokal am Stab vorhandenen Freiheitsgrade bestimmt und in einem Vektor mit sechs boolean Werten unter vorhandeneStabDOF gespeichert. Sechs weil pro Knoten drei Freiheitsgrade existieren und ein Stab zwei Knoten aufweist. Die lokal aktiven Freiheitsgrade können aber in der globalen Betrachtung anders aussehen, vor allem wenn ein Stab rotiert ist. Daher muss auf folgende Punkte geachtet werden:

- Es kann sein, dass lokal an einem Stab das Momentengelenk und somit der dritte Freiheitsgrad an diesem Knoten nicht vorkommt, aber im globalen System trotzdem vorhanden ist, da am selben Knoten weitere Stäbe angreifen, welche ein Moment aufnehmen können. Dies ist zum Beispiel im System in Abb. 13 der Fall. Daher wird zuerst ein isActiveDOF Vektor mit der Grösse der Anzahl Knoten mit drei multipliziert erzeugt, welcher mit dem boolean-Wert false initiiert wird. Diese werden nur aktiviert, wenn dieser Freiheitsgrad auch wirklich vorkommt. Das heisst zum Bei-

spiel, dass wenn am Knoten kein Stab angreift, welches den Momentenfreiheitsgrad aktiviert, dann bleibt dieser Wert auch auf false.

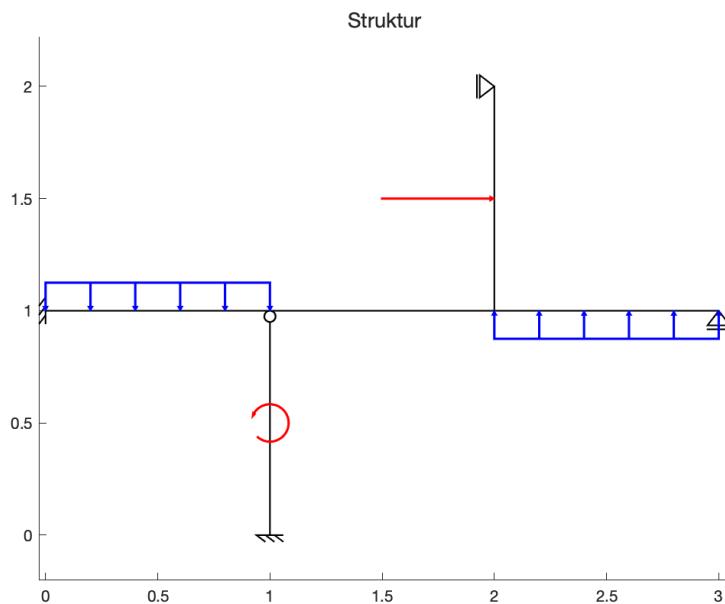


Abbildung 13: Mit Software erzeugte Struktur

- Wenn ein Stab zum Beispiel um 90° rotiert ist und im lokalen System der vertikale Freiheitsgrad am Knoten 2 nicht existiert, wird in der globalen Betrachtung der horizontale Freiheitsgrad an diesem Knoten nicht existieren, aber der Vertikale schon. Ein weiteres Problem stellen Stäbe dar, welche rotiert sind, aber nicht senkrecht stehen. Falls bei diesen zum Beispiel der vertikale Freiheitsgrad im lokalen System nicht existiert, werden global betrachtet trotzdem beide existieren. Daher wird für schräge Stäbe, welche weder horizontal noch vertikal im globalen Koordinatensystem stehen, jeweils immer der horizontale und vertikale Freiheitsgrad aktiviert. Dabei wird zunutze gemacht, dass der Betrag der Differenz des Cosinus und Sinus-Wertes des Rotationswinkels hierbei nie 1 ist. Bei horizontalen Stäben ändert sich nichts. Bei den vertikalen Stäben müssen jeweils die boolean-Werte der horizontalen Freiheitsgrade mit denen der vertikalen Freiheitsgrade vertauscht werden, um die lokalen vorhandeneStabDOF in die globalen activeStabDOF umzuwandeln. Um die vertikalen Stäbe herauszufiltern, werden die Bedingungen $\cos(\phi) = 0$ und $|\sin(\phi)| = 1$ zu Hilfe genommen
- Für Teilsysteme wird betrachtet, welche Freiheitsgrade der Enden effektiv existieren und nur diese werden aktiviert.
- Zum Schluss wird noch überprüft, ob eine Feder einen Freiheitsgrad aktiviert, da diese auch Kräfte aufnehmen können.

Die Implementierung der activeStabDOF ist in Abb. 14 zu sehen. Sobald alle vorhandenen Freiheitsgrade im Vektor isActiveDOF aktiviert wurden, werden die Anzahl true-Werte zusammengezählt, welche die Anzahl der aktiven Freiheitsgraden wiedergibt. Danach werden diese Freiheitsgrade nummeriert und den Stäben, den Teilsystemen sowie den Federn entsprechend zugeordnet. Nun kann mit den aktiven Freiheitsgraden gerechnet werden.

```

89 %activeStabDOF für jeden Stab -> wird auch für Kondensation von TS benötigt
90 for i = 1:Info.nStaebe
91     Stab(i).activeStabDOF = Stab(i).vorhandeneDOF;
92     diff = Stab(i).c - Stab(i).s; %um dof 1,2 zu aktivieren für rotiertes (ausser specialcases)
93     if ~isequal(abs(diff),1)
94         Stab(i).activeStabDOF([1,2,4,5]) = true;
95     end
96
97     if Stab(i).c==0 && abs(Stab(i).s)==1
98         Stab(i).activeStabDOF([1,2,4,5]) = Stab(i).activeStabDOF([2,1,5,4]);
99     end
100
101 end

```

Abbildung 14: Implementierung der activeStabDOF

3.2.4 Implementierung der Teilsysteme

Da zurzeit ein Teilsystem aus aneinander gereihten Stäben besteht und dies auch in der entsprechenden Reihenfolge eingegeben wird, müssen lediglich die Anfangs- und Endknoten herausgefiltert werden vom ersten und letzten Stab. Dabei spielt die Orientierung des Stabes eine wichtige Rolle. Mithilfe der Funktion getKnotenTS (Abb. 15) wird die Reihenfolge der Knoten im Teilsystem bestimmt. Dadurch kann der Anfangs- sowie Endknoten des System einfache herausgelesen werden.

```

1 function KnotenTS = getKnotenTS(KdS,anzahlStaebe)
2
3     KnotenTS = zeros(anzahlStaebe+1,1);
4
5     %[3 2 5 3 4 5]
6     %zu [2 3 5 4]
7
8     if KdS(1) == KdS(3) || KdS(1) == KdS(4)
9         KnotenTS(1) = KdS(2);
10        KdS([1,2]) = KdS([2,1]);
11    else
12        KnotenTS(1) = KdS(1);
13    end
14
15    for i = 1:anzahlStaebe-1
16        j = i*2;
17        KnotenTS(i+1) = KdS(j);
18        if KdS(j) == KdS(j+2)
19            KdS([j+1,j+2]) = KdS([j+2,j+1]);
20        end
21    end
22
23    KnotenTS(anzahlStaebe+1) = KdS(length(KdS));

```

Abbildung 15: Funktion getKnotenTS

Die Freiheitsgrade der beiden Enden im globalen Koordinatensystem repräsentieren die aktiven Freiheitsgrade. Mit diesen wird weitergerechnet. Mit Kondensation wird die Steifigkeitsmatrix und der Lastvektor für die Teilsysteme aufgestellt. In Abb. 16 ist beispielweise die Funktion kondensiereFTS gezeigt. Es wird eine separate Kondensierfunktion für Teilsysteme erstellt, um die Hauptfunktion DSM kompakt zu halten.

```

1 function F_TS_kond = kondensiereFTS(TS)
2
3     F_TS = TS.F_TS;
4     F_TS_kond = zeros(6,1);
5     K = TS.K_sys_TS;
6
7     isActive = TS.isActiveTSDOF;
8     nKnoten = length(TS.KnotenTSgeordnet);
9
10    e = [1:3,nKnoten*3-2:nKnoten*3];
11    eActive = isActive(e);
12    e = e(eActive);
13
14    i = [4:(nKnoten-1)*3];
15    iActive = isActive(i);
16    i = i(iActive);
17
18    F_TS_kond(eActive) = F_TS(e) - K(e,i)*K(i,i)^(-1)*F_TS(i);
19
20 end

```

Abbildung 16: Funktion kondensiereFTS

Für alle externen Freiheitsgrade des ganzen Systems können die Stabendkräfte und Stabendverschiebungen ausgerechnet werden. Die für interne Freiheitsgrade jedoch noch nicht, da das Zurückrechnen der kondensierten Freiheitsgrade der Teilsysteme nicht implementiert wurde. Das Gerüst dafür ist aber vorhanden.

3.3 Model.Output

3.3.1 Schnittkräfte

Damit die Schnittkräfte im Stab berechnet werden können, werden die Stabendkräfte sowie die Lastfälle, welche auf den Stab wirken benötigt. Daher wird zuerst jedem Stab, die Stablasten, die darauf angreifen zugeordnet. Danach wird ein Vektor für kritische Punkte initialisiert, welche am Anfang die Stellen 0 und 1 beinhaltet. Immer wenn eine Last am Stab angreift, werden für den Anfangs- sowie Endpunkt der Last zwei neue kritische Punkte erzeugt. Für die kritischen Punkte des Biegemomentes werden für den Fall einer verteilten Last mehrere neue Punkte auf dessen Spannweite erzeugt.

Parallel zur Erstellung der kritischen Punkte wird ein Vektor Lastfall erzeugt, welches die entsprechenden Stablasten an den jeweiligen Stellen speichert. Sobald diese beiden Vektoren fertiggestellt sind, werden diese nach den kritischen Stellen in aufsteigender Reihenfolge sortiert. Mit diesen beiden Vektoren kann nun mithilfe von Integration die Schnittgrösse an den kritischen Punkten ausgerechnet werden. In Abb. 17 ist beispielsweise die Berechnung der Querkräfte gezeigt.

```

121 %
122 for j = 2:length(SKStab(i).KP_V)-2
123     if SKStab(i).Lastfall_V(j+1) == SKStab(i).Lastfall_V(j)
124         switch StabLast(SKStab(i).Lastfall_V(j)).typ
125             case 2
126                 SKStab(i).SK_V(j+1:end) = SKStab(i).SK_V(j+1:end) + StabLast(SKStab(i).Lastfall_V(j)).val;
127             case 5
128                 SKStab(i).SK_V(j+1:end) = SKStab(i).SK_V(j+1:end) + StabLast(SKStab(i).Lastfall_V(j)).val*(SKStab(i).KP_V(j+1)-SKStab(i).KP_V(j))*SKStab(i).L;
129             end
130     elseif find(SKStab(i).Lastfall_V == SKStab(i).Lastfall_V(j),1) == j
131         z = find(SKStab(i).Lastfall_V == SKStab(i).Lastfall_V(j));
132         for y = z(1):(end)-1
133             SKStab(i).SK_V(y+1:end) = SKStab(i).SK_V(y+1:end) + StabLast(SKStab(i).Lastfall_V(j)).val*(SKStab(i).KP_V(y+1)-SKStab(i).KP_V(y))*SKStab(i).L;
134         end
135     end
136 end

```

Abbildung 17: Berechnung der Querkraft für die kritischen Punkte

Sobald die Schnittgrößen berechnet wurden, kann jeweils der Vektor mit den kritischen Punkten mit dem Vektor der Schnittgröße geplottet respektive gepatcht werden. Zum Zusammensetzen des ganzen Systems müssen die Plots richtig rotiert und zu den Koordinaten des Anfangsknoten des Stabes verschoben werden.

3.3.2 Verformungslinie

Um die Verformungslinie zu plotten, werden nur die Stabendverschiebungen benötigt. Mithilfe der in Abschnitt 2.5 beschriebenen shape-functions kann die Verformungslinie für statisch bestimmte sowie auch statisch unbestimmte Systeme gezeichnet werden. In Abbildung 18 sind zwei Systeme dargestellt für welche die Einflusslinie für die Querkraft in Stabmitte gesucht wird. Für das bestimmte System ist der lineare Verlauf zu erkennen, wohingegen die Verformungslinie des statisch unbestimmten Systems einen nichtlinearen Verlauf aufweist. Die Figuren wurden durch die Software erzeugt.

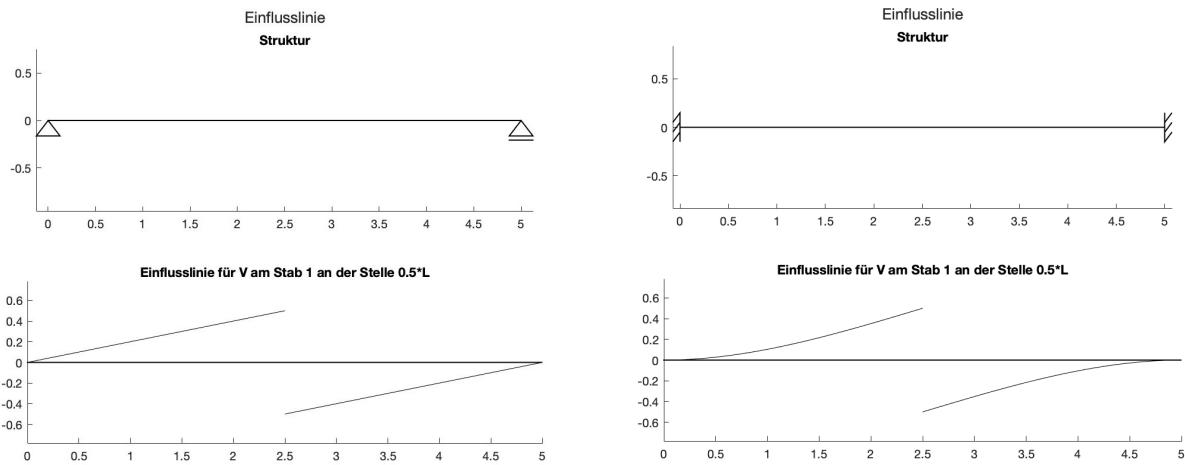


Abbildung 18: links: statisch bestimmt; rechts: statisch unbestimmt

3.3.3 Zeichnen

Das Zeichnen wurde auf freiwilliger Basis eingebaut und ist nicht Teil der Arbeit, da diese sich mit der richtigen Implementierung des rechnerischen Aspekts sowie einem klaren Aufbau der Software auseinandersetzt. Daher wird im Bericht nicht näher darauf eingegangen.

4 Numerische Beispiele

4.1 Schnittkräfte

Das Tragsystem in Abb. 19 wird durch eine verteilte Last mit Betrag 1, einer konzentrierten Last mit Betrag 7 und einer vorgeschriebenen Verdrehung am unteren Knoten mit dem Wert 2 belastet. Die horizontalen Stäbe weisen ein E-Modul mit dem Wert 2 und der vertikale ein E-Modul mit Wert 3 auf. Die Flächen und Trägheitsmomente aller Stäbe wird mit dem Wert 1 initiiert.

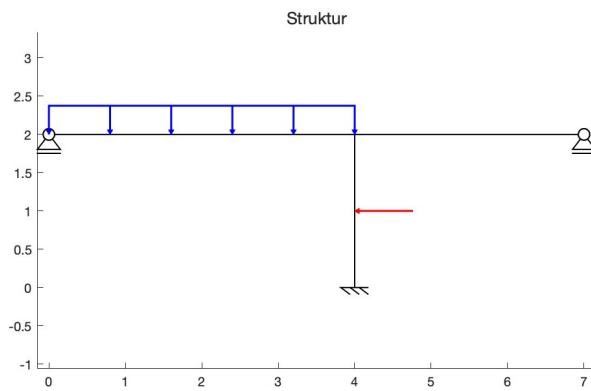


Abbildung 19: Tragstruktur

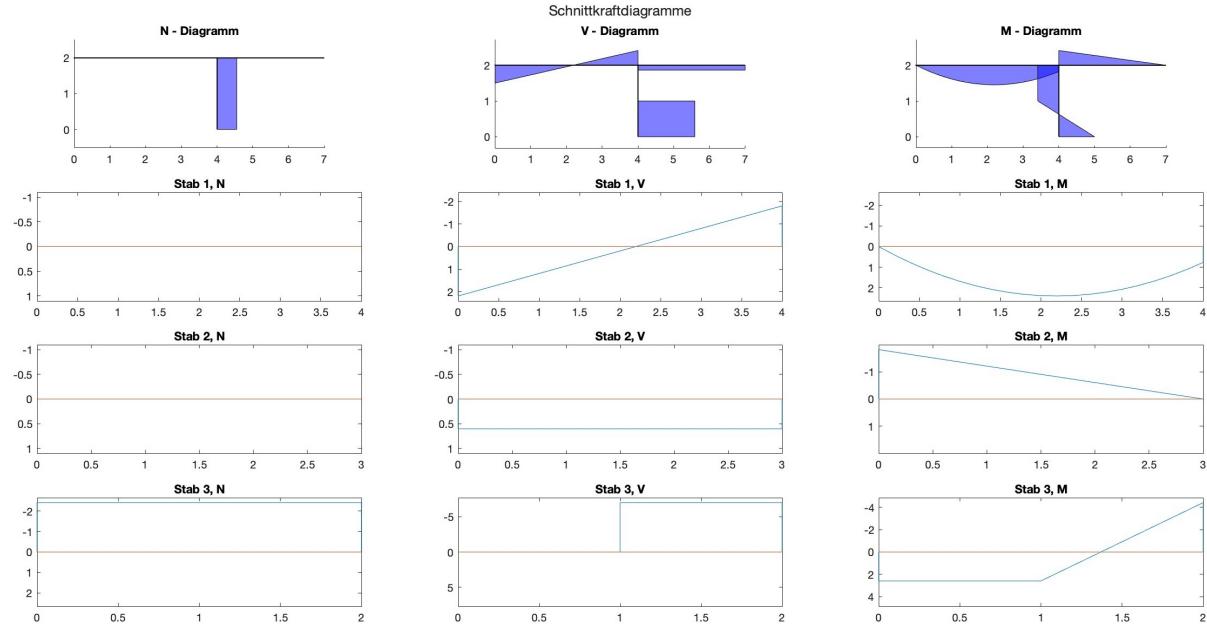


Abbildung 20: Schnittkraftdiagramme des oberen Systems

Das nachfolgende System (Abb. 21) stammt aus dem Kolloquium 4 des Kurses Baustatik II, welche im Frühjahrssemester 2022 gehalten wurde.¹⁴

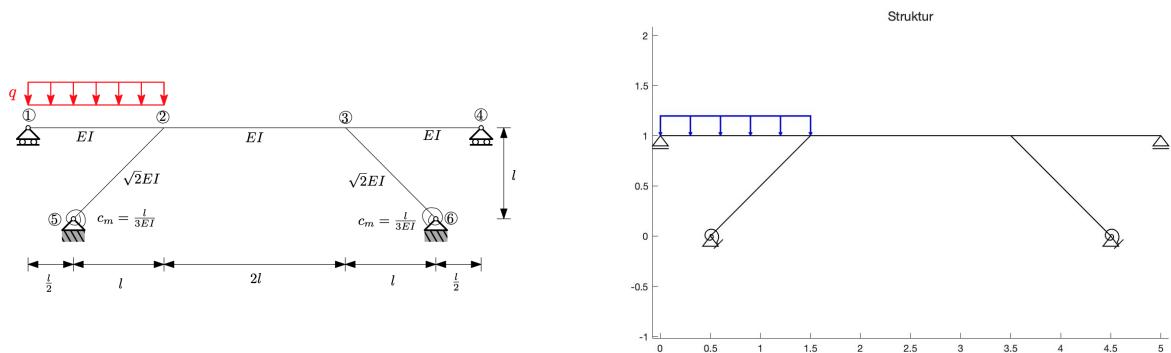


Abbildung 21: links: Ausschnitt aus Kolloquiumsunterlagen; rechts: von MATLAB erzeugt

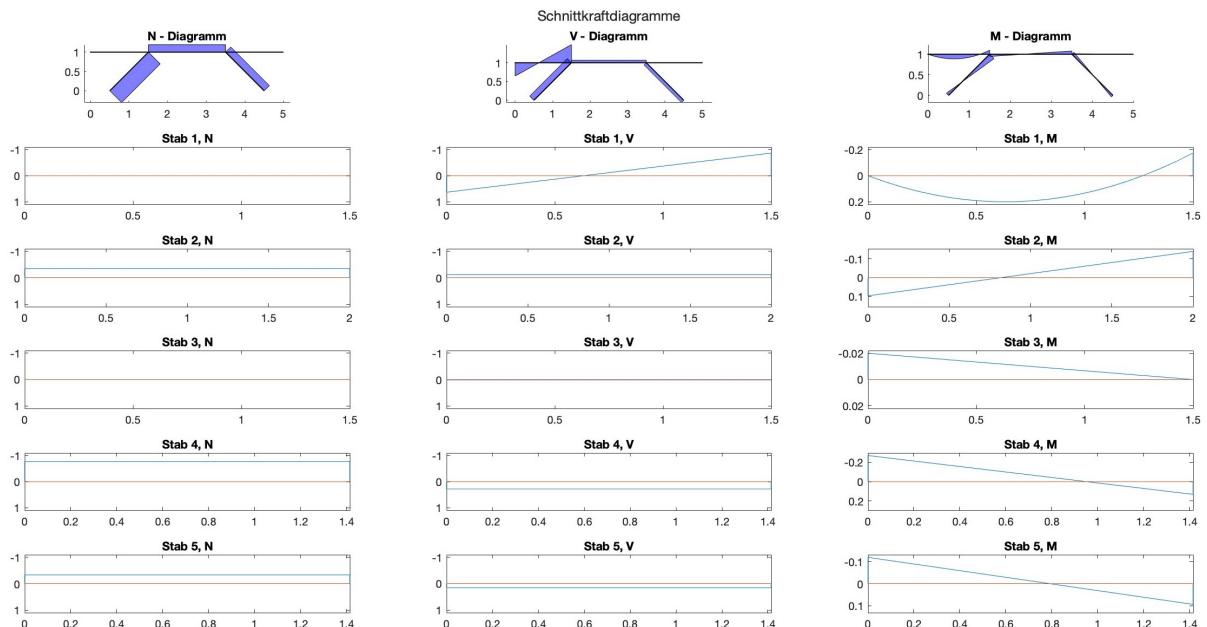


Abbildung 22: Schnittkraftdiagramme des oberen Systems

¹⁴Bild stammt aus den Unterlagen vom Kurs Baustatik II, FS22 der ETHZ und ist auf der folgenden Webseite zu finden: <https://chatzi.ibk.ethz.ch/education/baustatik/kolloquien.html> [Stand: 28.01.2023]

4.2 Auflagerreaktionen

Das System in Abb. 23 entstammt der Baustatik II Prüfung im Sommer 2022 und ist in der Fußnote verlinkt.¹⁵ In Abb. 24 sind die Auflagerreaktionen des Systems zu erkennen. Diese Figur wurde in MATLAB erzeugt.

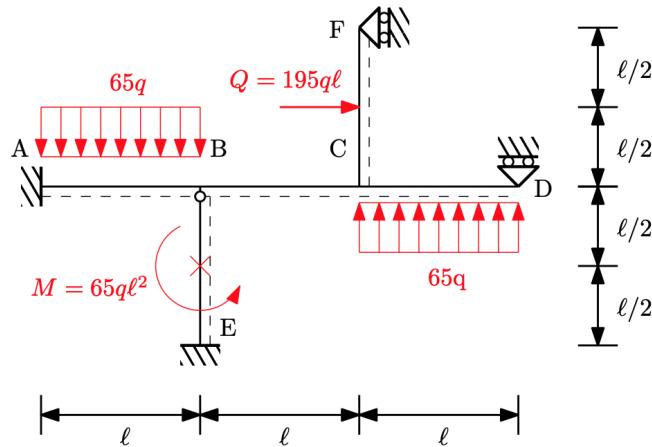


Abbildung 23: Ausschnitt aus Prüfung FS22 des Kurses Baustatik II

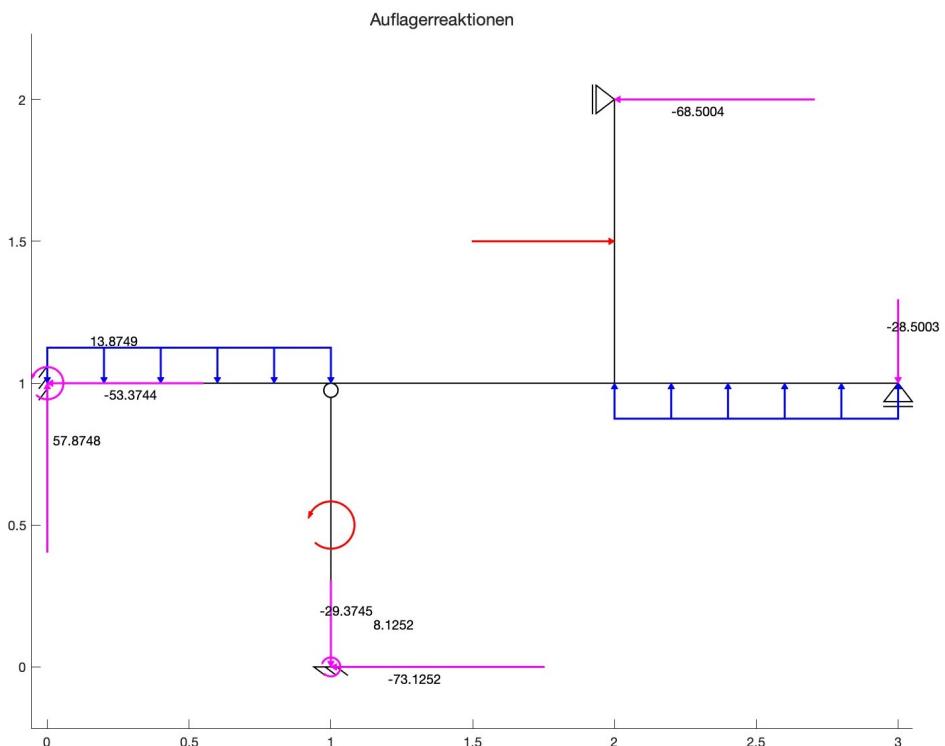


Abbildung 24: Auflagerreaktionen des oberen Systems

¹⁵<https://chatzi.ibk.ethz.ch/education/baustatik/pruefungen/musterloesungen-zu-sessionspruefungen.html> [Stand: 28.01.2023]

Im nachfolgendem Bild (Abb. 25) sind die Auflagerreaktionen des Systems aus Abbilung 21 dargestellt, welche von der Software berechnet werden.

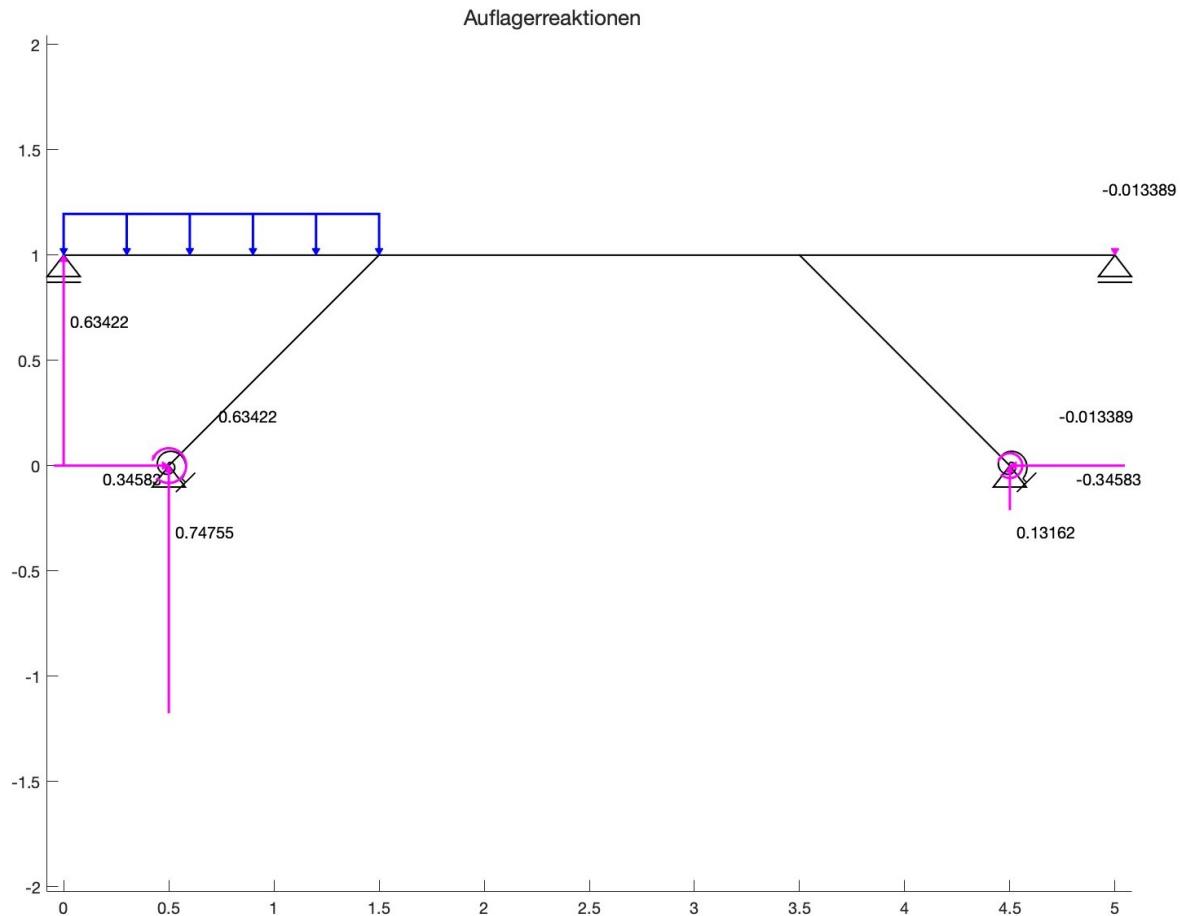


Abbildung 25: Auflagerreaktionen des Systems aus Kolloquium 4

4.3 Einflusslinien

Die nachfolgenden Abbildungen stellen alles Figuren dar, die durch die Software in MATLAB erzeugt wurden. Da diese selbsterklärend sind, werden sie nicht näher beschrieben.

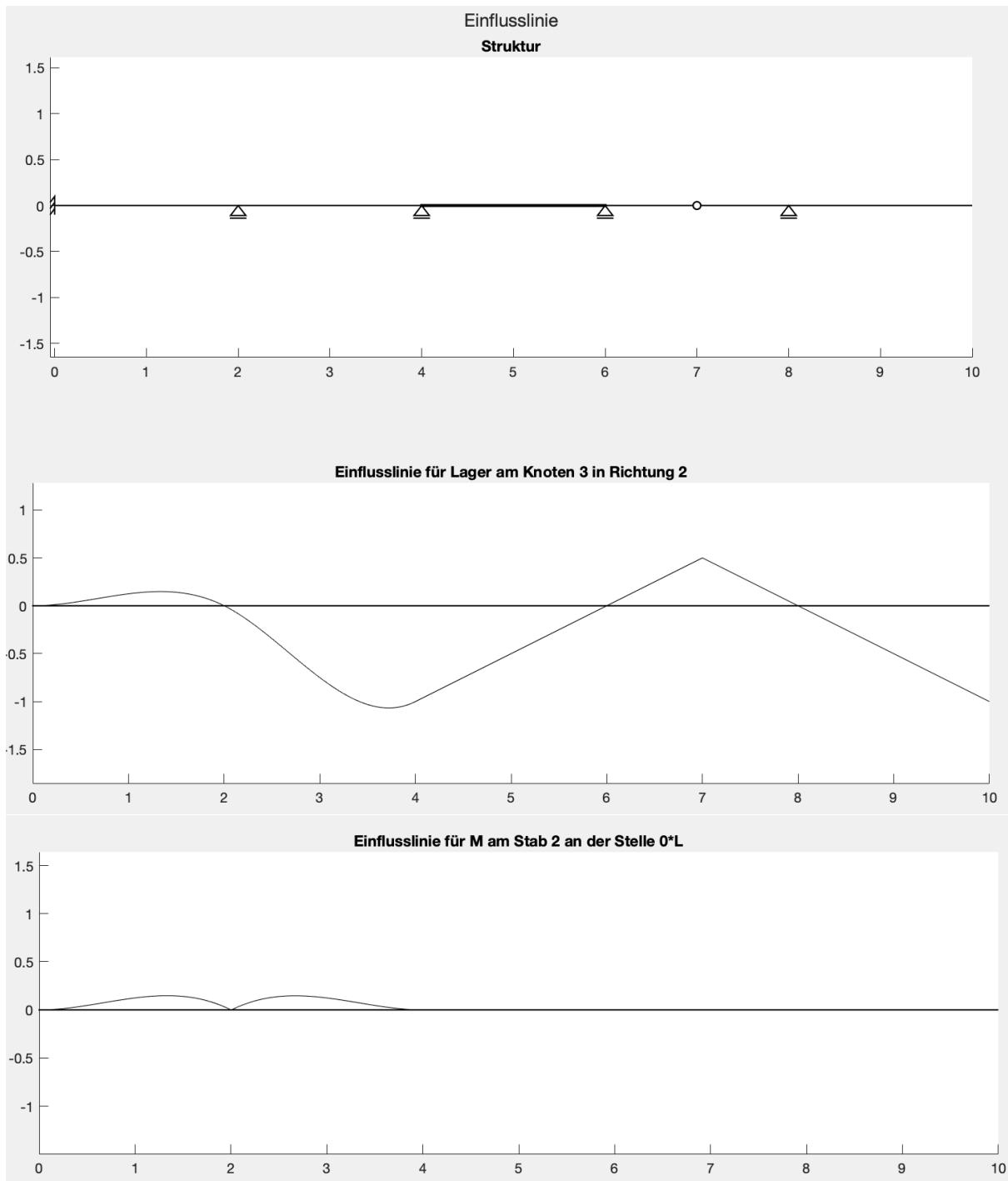


Abbildung 26: Beispiel 1

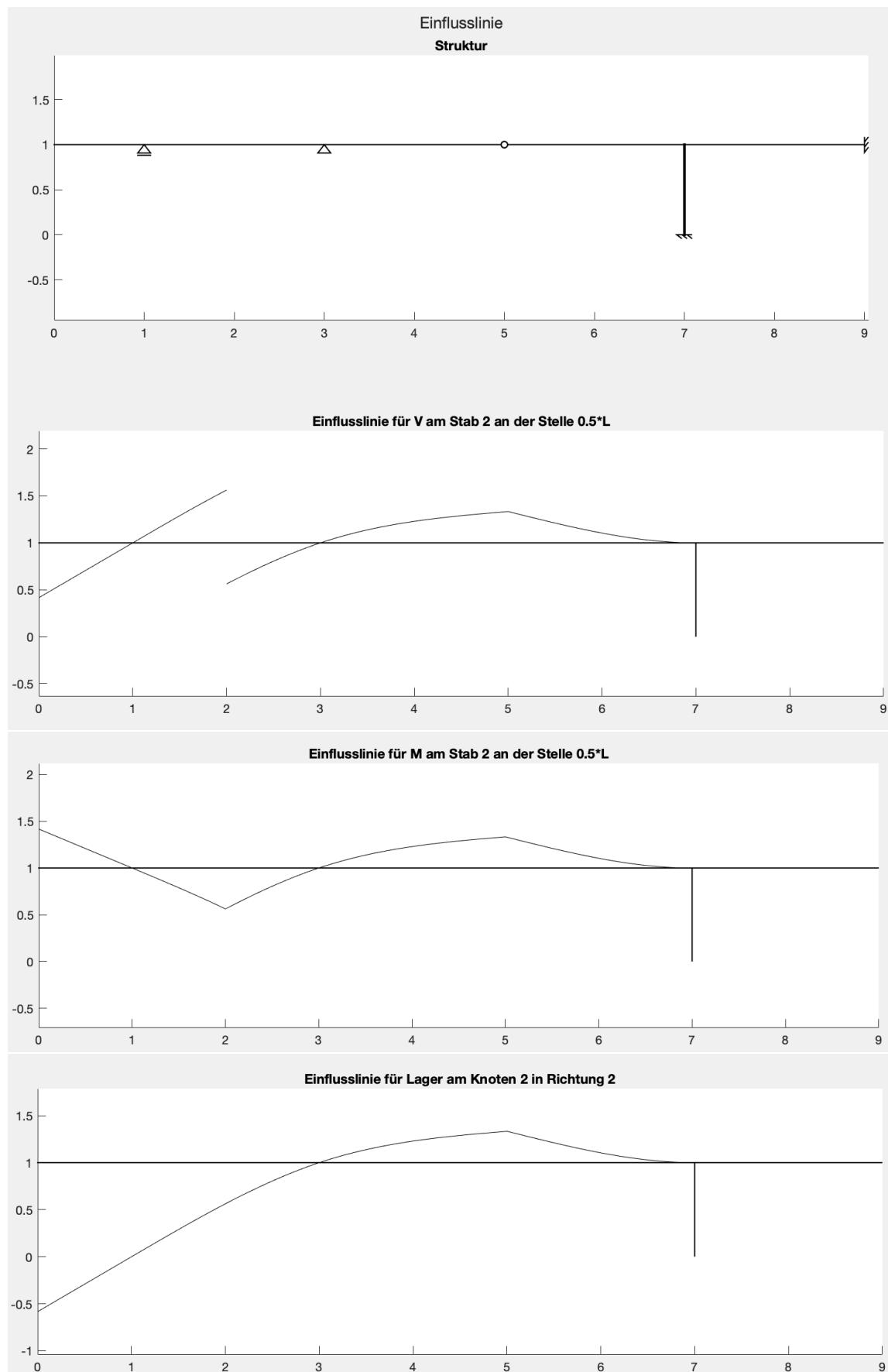


Abbildung 27: Beispiel 2

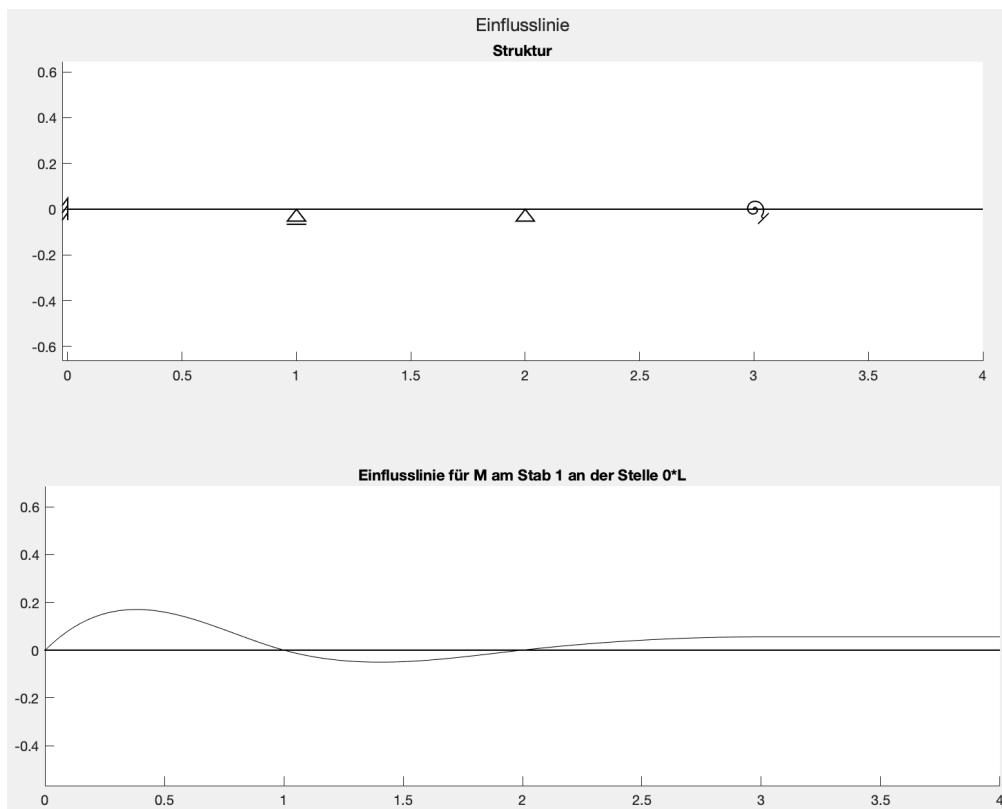


Abbildung 28: Beispiel 3 (Die Feder ist sehr steif)

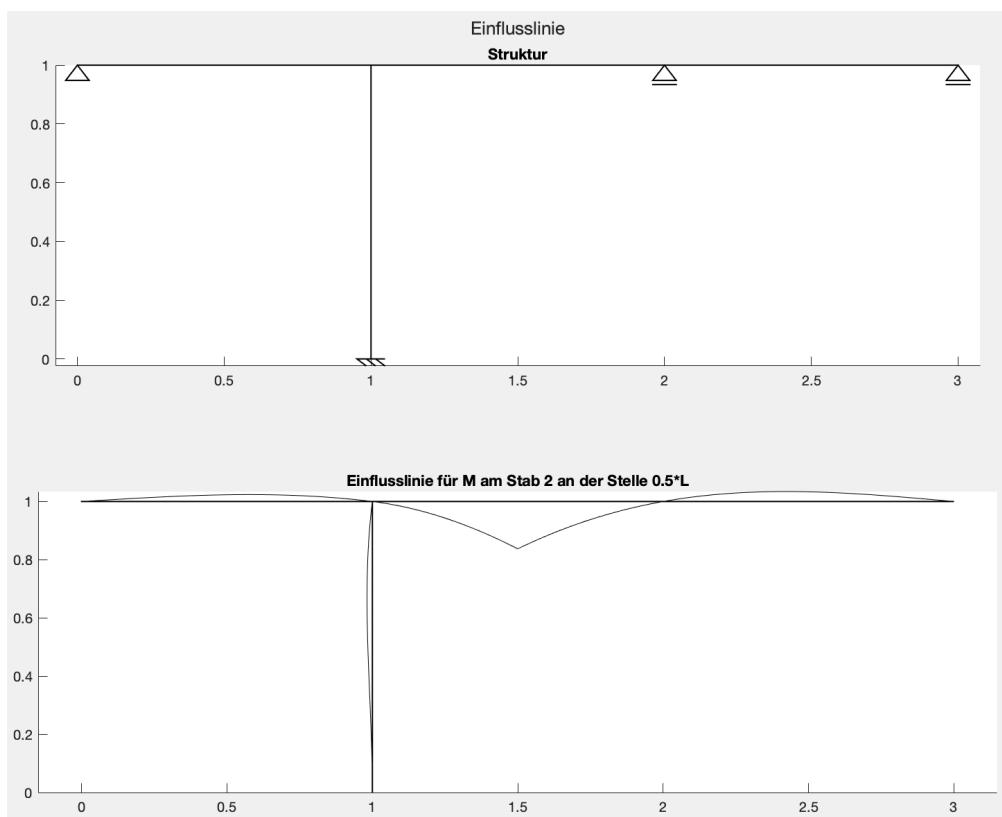


Abbildung 29: Beispiel 4

5 Schlusswort und Ausblick

In der vorliegenden Arbeit wurde eine Lehrsoftware entwickelt, um Schnittkräfte und Einflusslinien verschiedenster Tragstrukturen zu bestimmen, wobei das Berechnungsverfahren auf der direkten Steifigkeitsmethode basiert. Der Aufbau der Software weist klare Schnittstellen zwischen der Inputeingabe, den Berechnungen sowie der Ausgabe der Resultate auf. Somit können diese Teile eigenständig für sich weiterentwickelt und verbessert werden.

Die Arbeit ist gut gelungen und die gesetzten Ziele wurden erreicht. Es ermöglicht der Hauptzielgruppe, den Studierenden des Bauingenieurwesens der ETH Zürich, eigene Übungen zu generieren um die Verformungsmethode sowie die DSM anzuwenden. Des Weiteren hilft es, das Verständnis von Einflusslinien zu verbessern, da dies vor allem durch Betrachtungen mehrerer Systeme entwickelt wird. Dieser Bericht dient auch als Grundlage, sich in den rechnerischen Teil der Software einzuarbeiten. Daher wurde auf schwierigere Überlegungen detaillierter eingegangen. Im Code selbst sind weitere Anmerkungen in Form von Kommentaren angebracht für eine bessere Nachvollziehbarkeit.

Durch die Entwicklung dieser Software wurde mein mathematisches Verständnis von Matrizenrechnen sowie der Steifigkeitsmethode gestärkt und ich habe einen Einblick in Software Engineering erhalten. Weiter konnte ich meine Programmierfähigkeiten verbessern und habe viele Funktionen von MATLAB entdeckt. Ich bin zufrieden mit der Software, die ich mit der Betreuung von Dr. A. Egger entwickelt habe. Es kann aber stets erweitert und verbessert werden. Daher sind nachfolgend ein paar Ideen dazu aufgelistet:

- Die Entwicklung eines GUI. Eine Idee dafür ist vorhanden.
- Das Zurückrechnen der internen Freiheitsgrade der Teilsysteme
- Die Einführung von wählbaren Einheiten
- Die optische Darstellung der Resultate
- Ein Berichtsgenerator
- Implementierung von nichtlinearen Systemen. Das Grundgerüst dafür ist vorhanden.

An dieser Stelle möchte ich mich noch ganz herzlich bei Dr. A. Egger bedanken. Er hat mich sehr viel über Software Engineering gelehrt und sich viel Zeit genommen, mir die nötigen Grundlagen für die Entwicklung dieses Programms zu vermitteln. Der Aufbau der MatrizenStatik-Funktion, welches die klaren Schnittstellen formt, stammt von ihm. Weiter möchte ich mich auch herzlich bei Paul Sieber bedanken. Er hat sich um die organisatorischen Sachen gekümmert und war für meine Anliegen stets offen und hat sich sofort Zeit genommen für diese.

6 Literatur

Felippa, C. A. (2004) Introduction to finite element methods, *University of Colorado*, 885.

Kassimali, A. (2021) *Matrix analysis of structures*, Cengage Learning.

7 Quellenverzeichnis

Cubus AG: "Cubus: Software für Bauingenieure"

https://www.cubus-software.com/Guests/Home/d_main.html
[Stand: 29.1.2023]

Am IBK ETHZ bereits entwickelte Softwares:

<https://chatzi.ibk.ethz.ch/education/baustatik/vorlesungen.html>
[Stand: 29.1.2023]

Struct/structure arrays in MATLAB:

<https://ch.mathworks.com/help/matlab/ref/struct.html> [Stand 28.1.2023]

Kolloquium 4, Baustatik II ETHZ, FS22:

<https://chatzi.ibk.ethz.ch/education/baustatik/kolloquien.html>
[Stand: 28.01.2023]

Prüfung Baustatik II ETHZ, FS22: <https://chatzi.ibk.ethz.ch/education/baustatik/pruefungen/musterloesungen-zu-sessonspruefungen.html>
[Stand: 28.01.2023]

A Idee der GUI-Vorlage

Anzahl Knoten: ④

Einheiten: ♂ ♀

	X	Y	
Knoten 1			
Knoten 2			

④ → symbolisch [-]
→ numerisch [m]
+ numerisch [mm]

Anzahl Straßen: ④

Einheiten: ♂ ♀

	Startknoten	Endknoten	Querschnitt
Straße 1			
Straße 2			

Anzahl Teilsysteme: ④

	Beteiligte Straße
Teilsystem 1	[1, 2, 5]
Teilsystem 2	[3, 4]

Anzahl Lager: ④

	Knoten	Wert	Richtung	Start	Ende
Lager 1		○	↗		
Lager 2		○	↗		

Anzahl vorgeordnete Verbindungen: ④

Einheiten: ♂ ♀

	Knoten	Wert	Richtung	Start	Ende
Aufzehrung 1		○	↗		
Aufzehrung 2		○	↗		

Anzahl Feldern: ④

Einheiten: ♂ ♀

	Knoten	Wert	Richtung	Start	Ende
Feld 1	○	○	↗		
Feld 2	○	○	↗		

Anzahl Knotenlasten: ④

Einheiten: ♂ ♀

	Strasse	Richtung	Wert	Start	Ende
Knotenlast 1		↗	○		
Knotenlast 2		↗	○		

Anzahl verteilte Elementlasten: ④

Einheiten: ♂ ♀

	Start	Richtung	Wert	Start	Ende
Stablast_1		↗	○		
Stablast_2		↗	○		

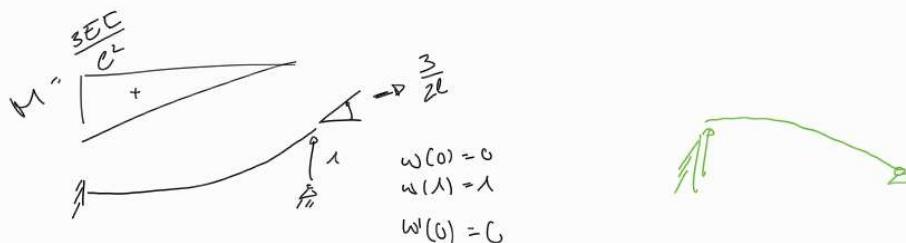
B Herleitung der Verdrehungen am Momentengelenk

Verschiebung nicht vorhandener DGF

$$x \cdot \delta_M = 1$$

$$\delta_{xx} = \frac{1}{EI} \cdot \frac{1}{3} \cdot l^3 = \frac{l^3}{3EI}$$

$$x = \frac{3EI}{l^3}$$



$$w'' = -\frac{M(x)}{EIy} = \frac{1}{EI} \cdot \left(1 - \frac{x}{l}\right) M$$

$$w' = \int -\frac{M(x)}{EIy} dx + C_1 = \frac{1}{EI} \cdot \left(Mx - M \frac{x^2}{2l}\right) + C_1$$

$$w = \left[-\frac{M(x)}{EIy} \cdot dx \right] + C_1 x + C_2 = \frac{1}{EI} \cdot \left(M \frac{x^2}{2} - M \frac{x^3}{6l}\right) + C_2$$

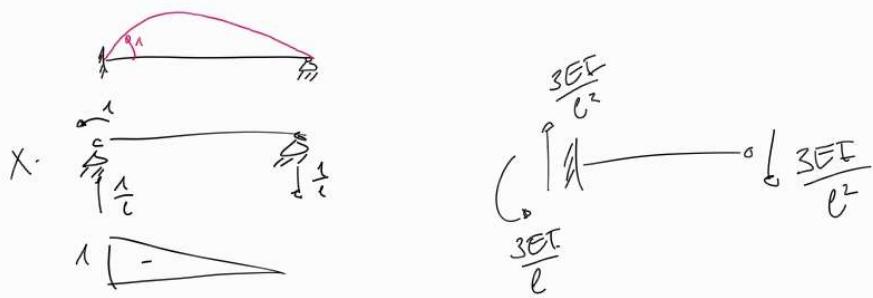
$$\Rightarrow w = \frac{1}{EI} \cdot \frac{3EI}{l^2} \cdot \left(\frac{x^2}{2} - \frac{x^3}{6l}\right)$$

$$w = \frac{3}{l^2} \cdot \left(\frac{x^2}{2} - \frac{x^3}{6l}\right)$$

$$w| = \frac{3}{l^2} \cdot \left(x - \frac{1}{2l} x^2\right)$$

$$w'(l) = \frac{3}{l^2} \cdot \left(l - \frac{l^2}{2l}\right) = \frac{3}{l^2} \cdot \frac{l}{2} = \underline{\underline{\frac{3}{2l}}} \quad \Rightarrow \text{infinitesimal klein}$$

\Rightarrow Steigung \approx Winkel



$$X \cdot \delta_{11} = 1$$

$$\delta_{11} = \frac{1}{3} \cdot 1 \cdot 1 \cdot \frac{\ell}{EI} = \frac{\ell}{3EI}$$

$$X = \frac{1}{\delta_{11}} = \frac{3EI}{\ell}$$

$$\frac{3EI}{\ell} = \frac{-}{c} + \frac{\ell-x}{c}$$

$$w'' = - \frac{M(x)}{EIy} = \frac{1}{EI} \cdot (1 - \frac{x}{\ell}) M$$

$$w' = \int - \frac{M(x)}{EIy} dx + C_1 = \frac{1}{EI} \cdot (Mx - M \frac{x^2}{2\ell}) + C_1$$

$$w = \left[- \frac{M(x)}{EIy} \cdot dx \right] + C_1 \cdot x + C_2 = \frac{1}{EI} \cdot (M \frac{x^2}{2} - M \frac{x^3}{6\ell}) + C_1 x + C_2$$

$$w(0) = 0 \Rightarrow C_2 = 0$$

$$w(\ell) = 0 : \frac{1}{EI} \cdot \frac{3EI}{\ell} \cdot \left(\frac{\ell^2}{2} - \frac{\ell^3}{6\ell} \right) + C_1 \cdot \ell = 0$$

$$\frac{3}{\ell} \cdot \frac{\ell^2}{3} + C_1 \cdot \ell = 0 \Rightarrow C_1 = -\lambda$$

$$w = \frac{3}{\ell} \cdot \left(\frac{x^2}{2} - \frac{x^3}{6\ell} \right) - \lambda$$

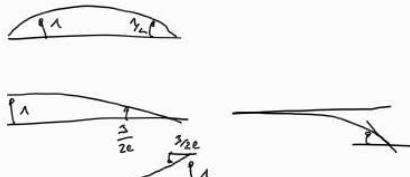
$$w' = \frac{3}{2} \cdot \left(x - \frac{1}{2\ell} x^2 \right) - \lambda$$

$$w'(0) = -\lambda$$

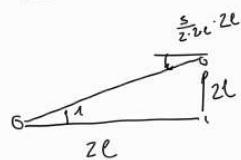
$$w'(\ell) = \frac{3}{2} \cdot \left(\ell - \frac{\ell^2}{2\ell} \right) - \lambda = \frac{3}{2} \cdot \frac{\ell}{2} - \lambda = \frac{3}{2} - \lambda = \underline{\underline{\frac{1}{2}}}$$

Glück usfintere

↳ efach wenn Flomantegänk:
 → auf eine Sittc!
 → Sittc picke & addiere



↳ wenn beide Sittc:
 Verdrehtig durch Verschiebung usfinge



jede Stab drüregooch

↳ vorhandene DDF checke
 ↳ wenn sum (vork ...) == 5

if ~ vorhandene DDF(5)

$$u_2 = -u_{loc}(2) \cdot \frac{3}{2L}$$

$$u_5 = u_{loc}(5) \cdot \frac{3}{2L}$$

$$u_6 = -u_{loc}(6) \cdot \frac{1}{2}$$

$$u_{loc}(3) = u_2 + u_5 + u_6$$

if ~ vorhandene DDF(6)

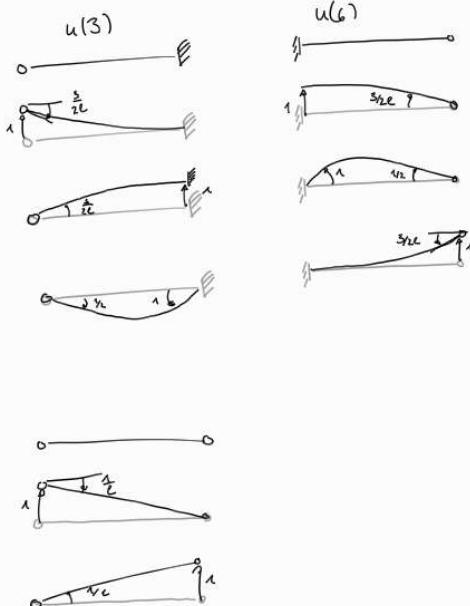
$$u_2 = -u_{loc}(2) \cdot \frac{3}{2L}$$

$$u_3 = -u_{loc}(3) \cdot \frac{1}{2}$$

$$u_5 = u_{loc}(5) \cdot \frac{3}{2L}$$

$$u_{loc}(6) = u_2 + u_3 + u_5$$

end



$$u_2 = -\frac{1}{2} \cdot u_{loc}(2)$$

$$u_5 = \frac{1}{2} \cdot u_{loc}(5)$$

$$u_{loc}(5) = u_2 + u_5$$

$$u_{loc}(6) = u_{loc}(3)$$

C Ausschnitte aus InputFile

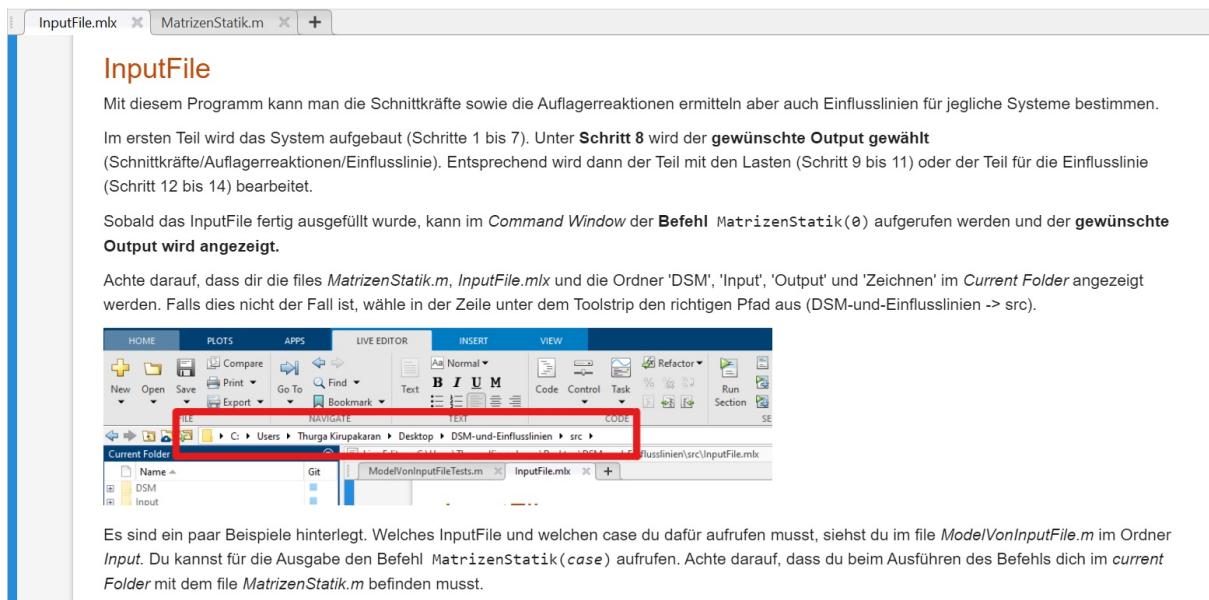


Abbildung 30: Am Anfang wird die Bedienung der Software erklärt

Systemaufbau

1.Knoten

Gebe die x und y Koordinaten der Knoten entsprechend ein. Die Werte an der ersten Stelle entsprechen jeweils der x- resp. y-Koordinate des ersten Knotens etc. Zurzeit sind die Knoten 1 an Stelle (0|0), Knoten 2 an Stelle (1|0) und Knoten 3 an Stelle (2|0) eingegeben. Ändere diese und erstelle die weiteren Knoten deines Systems. Die Werte werden dabei **mit einem Komma getrennt**. Es ist dir überlassen, welche Einheiten du wählst, solang du konstant bleibst. Falls L keine Einheiten aufweist, kannst du beispielsweise L = 1 wählen.

```
%AUSFÜLLEN
xPos = [0,1,2]';
yPos = [0,0,0]';

Knoten = table(xPos,yPos)
```

	xPos	yPos
1	0	0
2	1	0
3	2	0

```
in.Knoten = Knoten;
```

2.Stäbe

Die Knoten hast du nun erstellt. Wenn du auf *Run* drückst, siehst du diese in einer Tabelle. Nun erstellst du deine Stäbe. Dabei wählst du für einen Stab an welchem Knoten dieser anfängt und erfasst diesen Knoten unter StartKnoten. Dann wählst du an welchem Knoten der Stab aufhört und erfasst dies unter EndKnoten. Danach wählst du welchen Querschnitt dieser Stab aufweist. Die Querschnitte können in Schritt 3 erstellt werden.

Die Werte an der ersten Stelle gehören zum ersten Stab, die an der zweiten Stelle zum zweiten etc. Ändere die vorhandenen Wert und erstelle weitere Stäbe entsprechend deinem System. Wenn du auf *Run* drückst werden deine erstellten Stäbe in der Tabelle ersichtlich sein. **Trenne die Werte mit einem Komma.**

```
%AUSFÜLLEN
StartKnoten = [1,2]';
EndKnoten = [2,3]';
Querschnitt = [1,1]';
```

Abbildung 31: Im ersten Teil wird das System konstruiert

4.Lager

Nun werden die Lager erstellt. Dabei stehen folgende Lager zur Auswahl:

	Knoten						
Lager 1		○	○	○	○	○	○
Lager 2		○	○	○	○	○	○

Zuerst bestimmst du an welchen Knoten ein Lager erstellt wird. Danach kannst du in den logical-Ausdrücken den Typ deines Lagers festlegen. Setze die 1 an der entsprechenden Stelle. Achte darauf, dass du soviele logical Ausrücke hast wie Anzahl Lager. Die logical Ausdrücke müssen wie folgt geschrieben werden: `logical([0,0,0,0,1,0])`; die 1 wird dabei an der Stelle des entsprechenden Lagers gesetzt. Die Lagerung ist in einem Vektor gespeichert, also falls du das löscht, vergiss nicht, es wieder als Vektor zu schreiben.

```
%AUSFÜLLEN
Knoten = [1,2]';
Lagerung = [logical([1,0,0,0,0,0]);
            %logical([1,0,0,0,0,0]);
            logical([0,0,1,0,0,0]);
```

Abbildung 32: Eingabe der Lager sowie dessen Möglichkeiten

Definiere Output

8.Schnittkräfte oder Einflusslinie

Hier definierst du welchen Output du möchtest. Wähle die entsprechende Nummer!

1: gibt **Schnittkräfte** aus. Falls du die Schnittkräfte ausrechnen möchtest **bearbeite die Schritte 9 bis 11**

2: gibt die **Einflusslinie** aus. **Überspringe** dafür die **Schritte 9 bis 11** und **bearbeite die restlichen Schritte**

3: zeichnet die **Auflagerreaktionen**. **Bearbeite** dafür die **Schritte 9 bis 11**

```
%AUSFÜLLEN
in.gew_output = 2; % 1: Schnittkräfte | 2: Einflusslinie | 3: Auflagerreaktionen
```

Abbildung 33: Im zweiten Teil wird der Output gewählt

Lasten

9.Knotenlasten

Hier werden Knotenlasten erfasst, falls welche vorhanden sind. Die Lasten werden **im globalen Koordinatensystem** erfasst. Trenne die Werte mit einem Komma.

```
%AUSFÜLLEN, wenn vorhanden
Knoten = []';
Richtung = []';
Wert = []';

KnotenLasten = table(Knoten, Richtung, Wert)

KnotenLasten =
0x3 empty table

in.KnotenLasten = KnotenLasten;
```

10.konzentrierte Stablasten

Abbildung 34: Im dritten Teil können die Lasten eingegeben werden

Einflusslinie

12.Art der Einflusslinie

Hier kannst du die Art der Einflusslinie wählen. Wähle die entsprechende Nummer der gewünschten Art:

- 1: Einflusslinie für die **Normalkraft**. Bearbeite **Schritt 14**.
- 2: Einflusslinie für die **Querkraft**. Bearbeite **Schritt 14**.
- 3: Einflusslinie für das **Biegemoment**. Bearbeite **Schritt 14**.
- 4: Einflusslinie für eine **Lagerreaktion**. Bearbeite **Schritt 13**.

```
%AUSFÜLLEN, wenn Einflusslinie gesucht
TypEL = [3]; % 1:N, 2:V, 3:M, 4:Lager
```

13.Einflusslinie für eine Lagerreaktion

Hier wählst du an welchem Knoten sich das Lager befindet. Danach auf welche Richtung sich die Lagerreaktion bezieht. Also 1,2 oder 3.

```
if TypEL == 4
    %AUSFÜLLEN, wenn Einflusslinie für Lagerreaktion gesucht
    Knoten = [2];
    Richtung = [2];

    in.Einflusslinie = table(TypEL, Knoten, Richtung);
```

14.Einflusslinie für eine Schnittgrösse

Hier wählst du an welchem Stab sich deine gewünschte Schnittgrösse befindet. Danach wählst du die **Stelle in Abhängigkeit von L**. Wenn du dich für eine Schnittgrösse direkt neben einem Knoten interessierst, wählst du einfach den Stab, an welcher sich diese Stelle befindet und danach 0 oder 1, je nach dem welchem Ende es entspricht.

```
else
    %AUSFÜLLEN, wenn Einflusslinie für Schnittgrösse gesucht
    Stab = [1];
    Stelle = [0.5]; %zw. 0 <= x <= 1

    in.Einflusslinie = table(TypEL, Stab, Stelle);
end
```

Abbildung 35: Im vierten Teil kann die Einflusslinie gewählt werden

D Eigenständigkeitserklärung



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei Ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

DSM-basierte Lehrsoftware für Schnittkräfte und Einflusslinien in MATLAB

Verfasst von (in Druckschrift):

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):

Kirupakaran

Vorname(n):

Thurga

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum

Zürich, 28.01.2023

Unterschrift(en)

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.