Bachelor-Thesis zur Erlangung des Grades

Bachelor of Science

im Studiengang Maschinenbau

an der

Technischen Universität Berlin,
Institut für Bionik und Evolutionstechnik

Fachwerkoptimierung mittels Evolutionsstrategie

Betreuer: Prof. Dr.-Ing. Ingo Rechenberg
Dipl.-Ing. Michael Stache

vorgelegt von

Jonas Doßmann

Matrikelnr. 318490
Berlin, 29.10.2012

Tel: +49 15777 809002
Mail: j.dossmann@freenet.de

Die selbständige und eigenhändige Anfertigung versichert an Eides statt

Berlin, den

.....
Unterschrift

Inhaltsverzeichnis

0 Einleitung und Motivation

Optimierung begegnet man heute, wenn man genau hinsieht, überall und in vielen Facetten. Es fängt schon an beim Zeitmanagement – man sollte seine Zeit sinnvoll einteilen und nutzen um erfolgreich zu arbeiten und seine Freizeit genießen zu können; Die Senkung des Benzinverbrauchs von Autos ist eines der wichtigsten Aufgaben der Automobilbranche; Das Steigern der Effizienz des allgemeinen Energieverbrauchs ist ein großer Bestandteil der Energiewende und Voraussetzung für ihr Gelingen. Man sieht: Die Optimierung lohnt nicht nur, sie ist schlichtweg erforderlich. Durch die weitere Zunahme von Wettbewerb, Ressourcenverknappung und –verteuerung sowie anhaltendem ausgeprägten gesellschaftlichen Umweltbewusstseins und vielerlei weiterer Veränderungen, stehen die Chancen gut, dass der Optimierung noch eine Schlüsselrolle zuteil werden wird sofern dies nicht schon längst passiert ist. Die zentrale Frage lautet aber, *wie* Optima gefunden werden können.

Eine Optimierungsmethode ist die von Prof. Dr. Rechenberg erdachte *Evolutionsstrategie*. Sie zeichnet sich im Vergleich zu anderen Optimierungsverfahren durch einen universellen Charakter[1] aus: Einerseits ist sie auf sehr vielfältige technische Problemstellungen anwendbar, andererseits erschließt sich ihr auch jenseits des technischen Einsatzgebietes ein breites Anwendungsfeld, wie z.B. beim *Traveling–Salesman Problem*.

Genauso universell wie die Probleme auf die sie anwendbar ist, ist auch schon ihr Ansatz. Dieser zeichnet sich dadurch aus, den Anpassungsprozess eines Organismus an seine Umwelt in Abstraktion zu Eigen zu machen und ihn im Sinne der Optimierung anzuwenden. Die Überlegung, dass die Natur der beste Lehrmeister hinsichtlich Weiterentwicklung, Verbesserung und Anpassung ist, erscheint insofern plausibel, als dass auch noch im 21. Jahrhundert Natur– und Ingenieurwissenschaftler sämtlicher Couleur die Leistungen der Natur nach wie vor ehrfürchtig staunen lassen.

1 Ingo Rechenberg: Evolutionsstrategie ′94, Stuttgart 1994, S.15

1 Aufgabe und Problembeschreibung

Gegenstand der Untersuchung ist ein üblicher Fenstersturz eines Hauses. Vor dem Hintergrund eines möglichen Einsatzes eines neuen Werkstoffs soll das Optimierungspotential hinsichtlich des Gewichts ausgelotet werden. In Abstraktion wird der Sturz durch ein Fachwerk abgebildet, das durch drei Stäbe und drei Knoten die Form eines Dreiecks annimmt und an den beiden unteren, auf gleicher Höhe befindlichen Knoten durch ein Fest– bzw. ein Loslager befestigt ist. Damit ist das Fachwerk statisch bestimmt, was die Möglichkeit einer analytischen Ermittlung der Stab– und Lagerkräfte aus den Gleichgewichtsbedingungen an den Knoten ermöglicht. Auf dem oberen Knoten wirkt eine senkrechte Einzellast, welche die Masse oberhalb des Sturzes darstellt und vom Sturz selber getragen werden muss.

Die Gesamtaufgabe der Gewichtsoptimierung teilt sich in zwei einzelne und aufeinander aufbauende Aufgaben. Zunächst wird das Fachwerk unter der Maxime der Kohärenz in Matlab[2] abgebildet. Kohärenz bedeutet in diesem Fall, dass das vorliegende System ausschließlich aus jenen Informationen beschrieben wird, die ein allgemeines Statik–Problem beschreiben. Sie gewährleistet einerseits dass damit ein stabiles und fehlerresistentes Programm realisiert wird, und andererseits dass das Fachwerk mit minimalem Aufwand beliebig verändert und erweitert werden kann. Anschliessend wird genau dies an zwei deutlich komplexeren Fachwerken getestet werden um den Algorithmus zu verifizieren. Im Folgenden wird dann eine einfache Evolutionsstrategie in den erstellten Quellcode eingepasst sowie ein Format zur grafischen Ausgabe implementiert. Schließlich wird die Optimierung, bei denen mit unterschiedlichen Startschrittweiten experimentiert werden wird, durchgeführt. Die Ergebnisse werden sodann bewertet werden.

2 Matlab ist ein Produkt der MathWorks Inc. Verwendete Version: R2011b

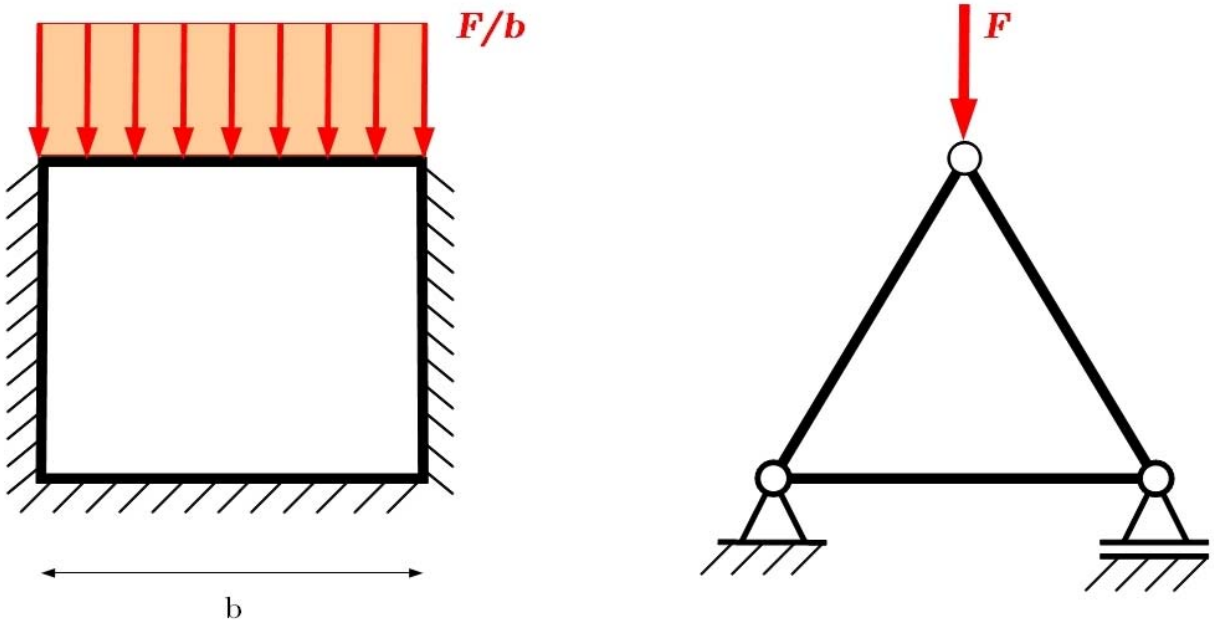


Abbildung 1.1: Modell eines Fenstersturzes und Abstraktion als Fachwerk

2 Grundlagen

2.1 Analyse der Stab– und Lagerkräfte

Der Fenstersturz wird durch ein statisch betimmes Fachwerk modelliert. Ein Fachwerk ist „statisch bestimmt“, wenn die Lager– und die Stabkräfte allein aus den Gleichgewichtsbedingungen (d.h. aus der Statik) bestimmbar sind.[3] Für ein statisch bestimmtes Fachwerk ergibt dich die notwendige Bedingung, dass die Anzahl der Stäbe, der Knoten und der Lagerreaktionen in einem definierten Verhältnis zueinander stehen müssen. Für ebene Fachwerke lautet dieses:

$$Anzahl\ Stäbe\ +\ Anzahl\ Lagerreaktionen\ =\ 2 \times Anzahl\ Knoten$$

Die hinreichende Bedingung für statische Bestimmtheit fordert, dass das Fachwerk *kinematisch bestimmt* sein muss, d.h. es darf nicht beweglich sein.[4] Dies lässt sich nicht im Vorfeld durch eine für alle Fachwerke gültige Gleichung überprüfen, sondern erst später, in einem fortgeschrittenen Stadium der Problembehandlung zu der an entsprechender Stelle eingegangen werden wird. Die einzige Methode, die Zulässigkeit von beliebigen Fachwerken im Sinne der kinematischen Bestimmtheit zu erkennen, ist mit bloßem Auge bewegliche, also nicht feste Teile zu identifizieren.

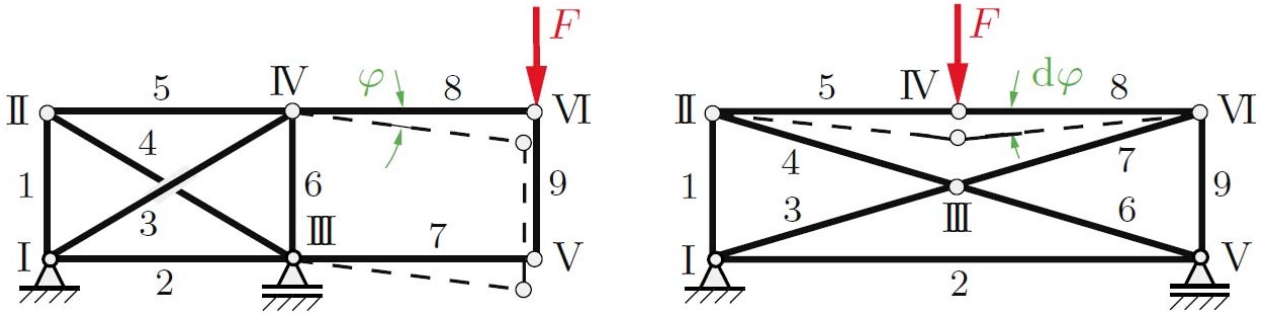


Abbildung 2.1: Kinematisch unbestimmte Fachwerke[5]

Unter diesen Voraussetzungen lassen sich nun die Stab– und Lagerkräfte analytisch ermitteln. Hierfür gibt es zahlreiche Verfahren, von denen allerdings das sog.

3 vgl. Dietmar Gross u.A.: Technische Mechanik Band 1 – Statik, Berlin Heidelberg 2009, S.148
4 vgl. ebd., S.148
5 ebd., S.149

Knotenpunktverfahren am praktikabelsten ist und daher auch hier zur Anwendung kommen soll.

Der erste Schritt ist hierbei, zunächst alle Knoten und Stäbe zu nummerieren. Die Knoten werden üblicherweise mit römischen, die Stäbe mit arabischen Ziffern versehen.[6] In diesem Fall wurde die Knotennummerierung entgegen dem Uhrzeigersinn vorgenommen, die Stabnummerierung erfolgt durch die den Stäben jeweils gegenüberliegenden Knoten. In der handschriftlichen Weise (die sich von der programmtechnischen Umsetzung unterscheidet) wird sodann jeder Knoten des Fachwerks ringsum freigeschnitten. Sämtliche Kräfte (Stabkräfte als Zugkräfte) werden anschließend unter Berücksichtigung ihrer geometrischen Lage im Koordinatensystem aufsummiert und jede dieser Gleichungen gleich 0 gesetzt. Dieses Gleichungssystem aus $2 \cdot$ *Knoten* Gleichungen (für jeden Knoten zwei Richtungen in der Ebene) werden dann systematisch miteinander verrechnet was zur allgemeinen Lösung für die unbekannten Kräfte führt. Die eindeutigen Lösungen für alle Kräfte wird durch die Bedingungen der statischen Bestimmtheit gewährleistet.

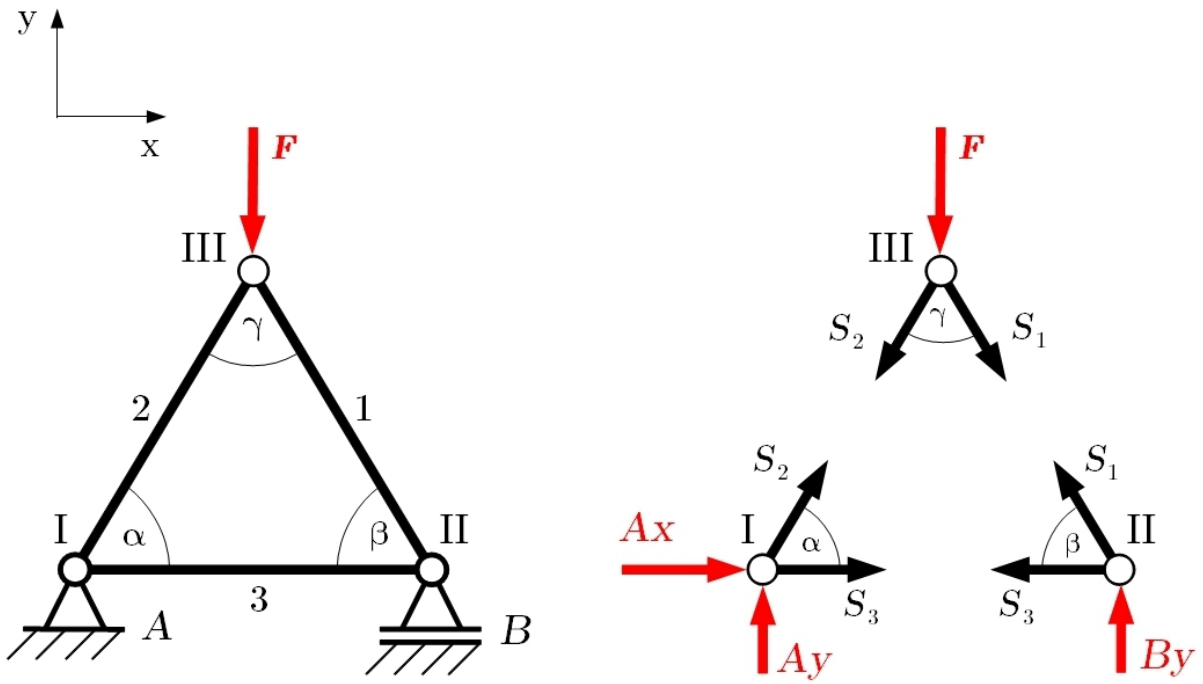


Abbildung 2.2: Links: Nummerierung der Knoten und Stäbe, Rechts: Knotenfreischnitte

6 Dietmar Gross u.A.: Technische Mechanik Band 1 – Statik, Berlin Heidelberg 2009, S.147f.

Die Gleichungen der Kräftegleichgewichte an den Knoten lauten wie folgt:

Knoten I:

$$\sum F_x = A_x + S_2 \cos(\alpha) + S_3 \stackrel{!}{=} 0$$
$$\sum F_y = S_2 \sin(\alpha) + A_y \stackrel{!}{=} 0$$

Knoten II:

$$\sum F_x = -S_1 \cos(\beta) - S_3 \stackrel{!}{=} 0$$
$$\sum F_y = B_y + S_1 \sin(\beta) \stackrel{!}{=} 0$$

Knoten III:

$$\sum F_x = S_1 \sin(\gamma) - S_2 \sin(\gamma) \stackrel{!}{=} 0$$
$$\sum F_y = -F - S_1 \cos(\gamma) - S_2 \cos(\gamma) \stackrel{!}{=} 0$$

Oder in ausführlicher Weise, um bereits auf das Matrizenverfahren hinzudeuten:

Knoten I:

$$\sum F_x = 0 \cdot S_1 + \cos(\alpha) \cdot S_2 + 1 \cdot S_3 + 1 \cdot A_x + 0 \cdot A_y + 0 \cdot B_y \stackrel{!}{=} 0$$
$$\sum F_y = 0 \cdot S_1 + \sin(\alpha) \cdot S_2 + 0 \cdot S_3 + 0 \cdot A_y + 1 \cdot A_y + 0 \cdot B_y \stackrel{!}{=} 0$$

Knoten II:

$$\sum F_x = -\cos(\beta) \cdot S_1 + 0 \cdot S_2 + (-1) \cdot S_3 + 0 \cdot A_x + 0 \cdot A_y + 0 \cdot B_y \stackrel{!}{=} 0$$
$$\sum F_y = \sin(\beta) \cdot S_1 + 0 \cdot S_2 + 0 \cdot S_2 + 0 \cdot S_3 + 0 \cdot A_x + 0 \cdot A_y + 1 \cdot B_y \stackrel{!}{=} 0$$

Knoten III:

$$\sum F_x = \sin(\gamma) \cdot S_1 - \sin(\gamma) \cdot S_2 + 0 \cdot S_3 + 0 \cdot A_x + 0 \cdot A_y + 0 \cdot B_y \stackrel{!}{=} 0$$
$$\sum F_y = -\cos(\gamma) \cdot S_1 - \cos(\gamma) \cdot S_2 + 0 \cdot S_3 + 0 \cdot A_x + 0 \cdot A_y + 0 \cdot B_y \stackrel{!}{=} F$$

Dieses Gleichungssystem aus sechs Gleichungen beinhaltet mit drei Stab– und drei Lagerkräften insgesamt sechs Unbekannte und ist damit lösbar. Ferner sind die Gleichungen, da das Fachwerk statisch bestimmt ist, linear unabhängig, was eindeutige Lösungen gewährleistet.

In Matrizenform überführt, entsteht das Matrixschema $\underline{\underline{A}} \cdot \vec{x} = \vec{b}$. Matrix $\underline{\underline{A}}$ ist die sogenannte *Koeffizientenmatrix*. Sie stellt die bijektive Abbildung der Knoten und vorhandener Zwangskräfte dar. Die Zeilen entsprechen dabei den Knoten in jeweils x– und y–Richtung, die Spalten den Stab– und Lagerkräften.

$$\text{Koeffizientenmatrix}: \quad \begin{array}{c} \\ K1x \\ K1y \\ K2x \\ K2y \\ K3x \\ K3y \end{array} \begin{array}{c} \begin{array}{cccccc} S_1 & S_2 & S_3 & A_x & A_y & B_y \end{array} \\ \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \end{array}$$

Abbildung 2.3: Aufbau und Abbildung der Koeffizientenmatrix

Vektor $\vec{x}$ enthält die zu ermittelnden Unbekannten, in diesem Fall die Kräfte, die entsprechend der Spalten der Koeffizientenmatrix angeordnet sein müssen. Vektor $\vec{b}$ enthält die äußeren Lasten, die auf der rechten Seite des Gleichheitszeichens stehen müssen. Das Gleichungssystem in Matrizenform lautet somit:

$$\underline{\underline{A}} \cdot \vec{x} = \vec{b} = \begin{bmatrix} 0 & \cos(\alpha) & 1 & 1 & 0 & 0 \\ 0 & \sin(\alpha) & 0 & 0 & 1 & 0 \\ -\cos(\beta) & 0 & -1 & 0 & 0 & 0 \\ \sin(\beta) & 0 & 0 & 0 & 0 & 1 \\ \sin(\gamma) & -\sin(\gamma) & 0 & 0 & 0 & 0 \\ -\cos(\gamma) & -\cos(\gamma) & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ A_x \\ A_y \\ B_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ +F \end{bmatrix}$$

An dieser Stelle lässt sich nun eindeutig untersuchen, ob das Fachwerk kinematisch bestimmt ist, und zwar indem man die Determinante der Koeffizientenmatrix prüft. Mit $\alpha = \beta = \gamma = 60\,^{\circ}$ (gleichseitiges Dreieck) ist in diesem Fall $det(A) = 0.866$. Sollte die Determinante einer Koeffizientenmatrix gleich 0 sein, ist das Fachwerk nicht kinematisch bestimmt.

Damit ist das Knotenpunktverfahren praktisch abgeschlossen. Die Lösung dieser Matrizengleichung ließe sich nun beispielsweise per Gauss–Algorithmus ermitteln.

2.2 Evolutionsstrategie

Der Name Evolutionsstrategie ist hier als Überbegriff für das Rechenberg'sche Optimierungsverfahren zu verstehen, das de facto verschiedenartige

Evolutionsstrategien umfasst. So kann der Anpassungsprozess von Lebewesen an ihre Umwelt auf unterschiedlichen Abstraktionsniveaus simuliert werden, das entsprechend dem vorhandenen Optimerungsproblem ausgewählt werden muss. Für komplexe Optimierungsprobleme sind in der Regel entsprechend aufwändige Evolutionsstrategien erforderlich, doch in diesem Fall wird die sogenannte (1+1)−Evolutionsstrategie ausreichen. Die (1+1)−Evolutionsstrategie „stellt die maximale Abstraktion des natürlichen Geschehens dar".[7]

Zu Beginn steht die Definition und Formulierung einer *Zielfunktion* die beschreibt, welche Grösse oder welches Merkmal optimiert werden soll. Unter Einbeziehung von ausserdem zu definierenden Randbedingungen heisst das, die Zielfunktion zu minimiern oder zu maximieren. Hier handelt es sich um die Gewichtsoptimierung eines Fachwerks, d.h. die Summe der Stabmassen ist zu minimieren.

$$Masse = \sum_{i=1}^{3} m_i \Rightarrow Minimum$$

Ferner müssen die Variablen definiert werden, durch deren Veränderung (Variierung) das vorliegende System optimiert werden soll. Im hier behandelten Fachwerk werden das Knotenpositionen, also bestimmte Knotenkoordinaten sein. Mithilfe des Zufalls werden diese Variablen sodann variiert. Für beide Variablenmengen (ursprüngliche und variierte) wird dann jeweils der Wert der Zielfunktion errechnet und dieser miteinander verglichen. Es gilt nun das Darwin′sche Prinzip *Survival of the fittest*. Das bedeutet, dass diejenige Variablenmenge für die der Zielfunktionswert schlechter ausfällt ausstirbt, und diejenige die den besseren Zielfunktionswert aufweist eine neue variierte Variablenmenge (von sich) erzeugt. Anschliessend werden erneut die Zielfunktionswerte ermittelt und miteinander verglichen usw. Über die mehrfache Wiederholung dieses Verfahrens (*Generationen*) wird sich so dem Optimum schrittweise genähert, wobei Stagnation (Ursprüngliche Variablenmenge ist besser als die variierte) oder Fortschritt (variierte Variablenmenge ist besser) auftreten können.

Dies ist das (streng genommen unbiologische) Abbild von Fortpflanzung und

7 Bernd Kost: Optimierung mit Evolutionsstrategien, Frankfurt am Main 2003, S. 109f.

natürlicher Selektion, das die (1+1)−Evolutionsstrategie kennzeichnet: Ein Elter erzeugt ein Kind, welches dadurch mutiertes Erbgut des Elters aufweist (variierte Variablen). Der eine oder der andere ist nun besser an seine Umwelt angepasst (bewirkt eine Verbesserung der Zielvorgabe). Ist es der Elter, pflanzt er sich erneut fort und erzeugt ein neues Kind mit anders (da zufällig) mutiertem Erbgut. Ist es das Kind, ist dieses der neue Elter der sich fortpflanzt. Was in der Natur die über Generationen hinweg dauernde Anpassung des Lebewesens an seine Umwelt ist, entspricht in der Evolutionsstrategie der iterativen, schrittweisen Annäherung der variablenbedingten Zielfunktion an das Optimum.

3 Umsetzung und Lösung

3.1 Bestimmung von Statik–Problemen

Um sich auf die Implementierung vorzubereiten und ein kohärentes, fehlerresistentes Programm erstellen zu können, muss bekannt sein, welche Informationen ein Statik–Problem beschreiben, aus denen schließlich die Kräfte ermittelt werden können.

Statik–Probleme werden beschrieben durch

– *Geometrie–Information* (vorzugeben als ein Satz von Punkten, bezogen auf ein beliebig festzulegendes Koordinatensystem)

– *Topologie–Information* (beschreibt die Lage der einzelnen Teile des Systems zueinander)

– *Information über die Belastung*

– *Information über die Lagerung.*[8]

Das Statik–Problem im vorliegenden Fall ist ein spezifisches, und zwar geht es hier um ebene, statisch bestimmte Fachwerke und das Ermitteln der Stab– und Lagerkräfte. Damit können die letzten beiden Punkte konkretisiert werden, und zwar handelt es sich hier um Informationen über Ort und Richtung der Lagerung, bei der Belastung zusätzlich um die Größe der Kraft bzw. der Kräfte.

3.2 Theoretischer Hintergrund

Diese Informationen über Geometrie, Topologie, Belastung und Lagerung werden nun in Matrizenform definiert, auf die sämtliche folgende Rechenoperationen zugreifen und aus denen allein schließlich die Lösungen ermittelt werden. Für die oben beschriebene Art von Statik–Problemen ergeben sich vier Matrizen (*Eingangsmatrizen*), die die Ausgangs– und Grundlage für jedes weitere Vorgehen bilden.

8 Jürgen Dankert u.A. Technische Mechanik – Computerunterstützt, Stuttgart 1994, S.77

Es sei angemerkt, dass der Aufbau und die Problemlösung eines statisch bestimmten Fachwerks in der Ebene allein aus den Statik–Informationen ein deutlich höheres Abstraktionsniveau darstellt als jenes, das sich in Kapitel 2.1 darstellte und einen entsprechenden Aufbau des Lösungsalgorithmus erfordert. Im Folgenden muss und wird also die Erläuterung des Lösungsalgorithmus einen deutlich größeren Rahmen einnehmen.

Um die Entwicklung des Lösungsalgorithmus möglichst nachvollziehbar zu halten, werden einfache und simple Werte verwendet, die aber selbstverständlich vom Anwender veränderbar sind. So hat die Kraft F zunächst einen Wert von 100, die Stablängen jeweils Werte von 10 woraus ein gleichseitiges Dreieck mit einheitlichen Innenwinkeln entsteht. Es sei an dieser Stelle darauf hingewiesen, dass sämtliche Werte einheitenlos sind, bzw. einheitenkonform eingegeben werden müssen. Eine Kraft in Newton muss Stablängen bzw. Knotenabständen in Meter entsprechen (und nicht beispielsweise Millimeter).

3.2.1 Eingangsmatrizen

Geometrie–Information
Zunächst werden die Knoten und ihre Lage entsprechend dem Koordinatensystem definiert. Für ein gleichseitiges Dreieck mit den Stablängen 10, und einem Koordinatensystem in Knoten 1, folgen für Knoten 1 die Koordinaten (0 , 0), für Knoten 2 (10 , 0) und für Knoten 3 näherungsweise (5 , 8.66).

$$Knoten = \begin{bmatrix} 0 & 10 & 5 \\ 0 & 0 & 8.66 \end{bmatrix}$$

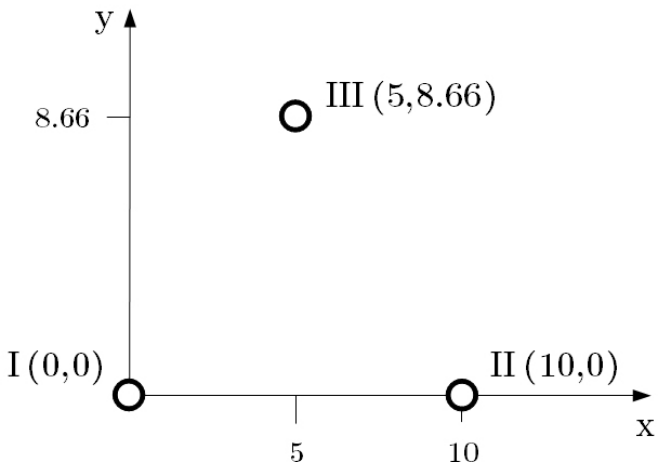


Abbildung 3.1: Knotenpositionen im Koordinatensystem

Information über die Belastung

Der Kraftvektor greift an Knoten 3 an und wirkt entgegen der y–Richtung. Damit wird in einer Matrix LASTEN in der dritten Spalte (Knoten) und zweiten Zeile (Kraft wirkt in y–Richtung) der Betrag der Kraft eingetragen und mit negativem Vorzeichen versehen (entgegengesetzte Wirkrichtung bzgl. Koordinatenrichtung).

$$Lasten = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -100 \end{bmatrix}$$

Information über die Lagerung

Die Lagerung wird realisiert, indem der Anteil der aufgenommenen Kraft des Lagers je Richtung von 0 bis 1 eingetragen wird. *0* bedeutet, keine Lagerung bzw. es wird keine Kraft aufgenommen, *1* bedeutet dass hier ein Lager existiert, das Kräfte in diese Richtung vollständig aufnimmt. Das Festlager in Knoten 1 bekommt also für die x– und y–Richtung jeweils eine *1*, da hier für beide Richtungen Kräfte aufgenommen werden. Das Loslager in Knoten 2 bekommt lediglich für den Eintrag in y–Richtung eine *1*, denn es nimmt nur in y–Richtung Kräfte auf.

$$Lager = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

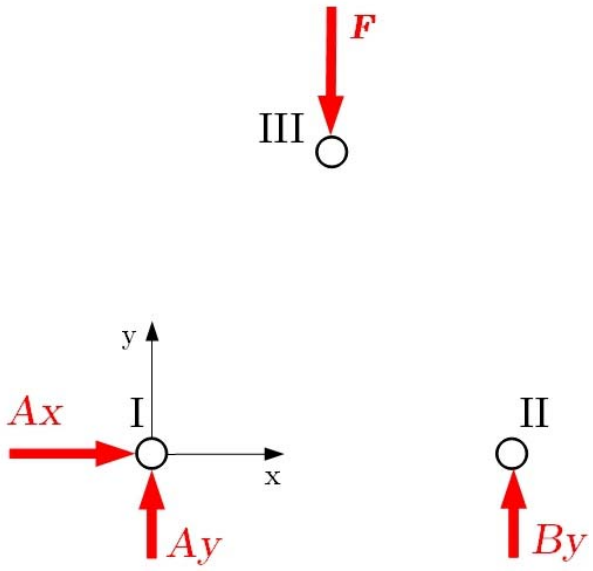


Abbildung 3.2: Aufbau von Knoten, Lager und Lasten

Topologie–Information

Schließlich muss die Topologie–Information erfasst werden, was derart geschieht, dass die Stäbe durch ihre angebundenen Knoten beschrieben werden. Stab 1 ist also angebunden an Knoten 2 und 3, Stab 2 an Knoten 1 und 3, und Stab 3 an Knoten 1 und 2. Welcher Eintrag oben bzw. unten steht, sollte hierbei irrelevant sein und wurde auch so in der Implementierung realisiert.

$$Verbindung = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 3 & 2 \end{bmatrix}$$

Zusammenfassung der Strukturmerkmale

Diese vier Matrizen kennzeichnen zwei verschiedene Strukturmerkmale: LAGER, LASTEN und KNOTEN bilden auf die Knoten im Koordinatensystem ab. Die Spalten dieser Matrizen entsprechen damit den einzelnen Knoten, ihre Zeilen den Koordinatenrichtungen. Die Matrix VERBINDUNG allerdings bildet – ganz unabhängig von einem Koordinatensystem – auf die Stäbe mit ihren angebundenen Knoten ab. Die Spalten entsprechen somit den einzelnen Stäben, die Zeilen ihren jeweils angebundenen Knoten (und nicht Koordinatenrichtungen).

3.2.2 Geometriebestimmung

Die Matrix VERBINDUNG beschreibt nicht nur die Knoten an denen die jeweiligen Stäbe angebunden sind, sondern dient im Folgenden auch der Einführung von *Stabvektoren*. Aus diesen Stabvektoren werden wiederum die *Stabkoeffizienten* generiert, die die aus der Matrix KNOTEN aufbereiteten und nutzbar gemachten Geometrie–Informationen darstellen.

Um zunächst die Stabvektoren herzustellen, werden im Programm die Spalteneinträge von VERBINDUNG als Vektoren mit Ursprung im Knoten entsprechend Eintrag der oberen Zeile und Orientierung auf den Knoten entsprechend Eintrag der unteren Zeile interpretiert (Abbildung 3.3). Durch die Abstände der Knoten wiederum, die unter Rückgriff auf die Matrix KNOTEN ermittelt wird, ist ihre Länge komponentenweise vorgegeben.

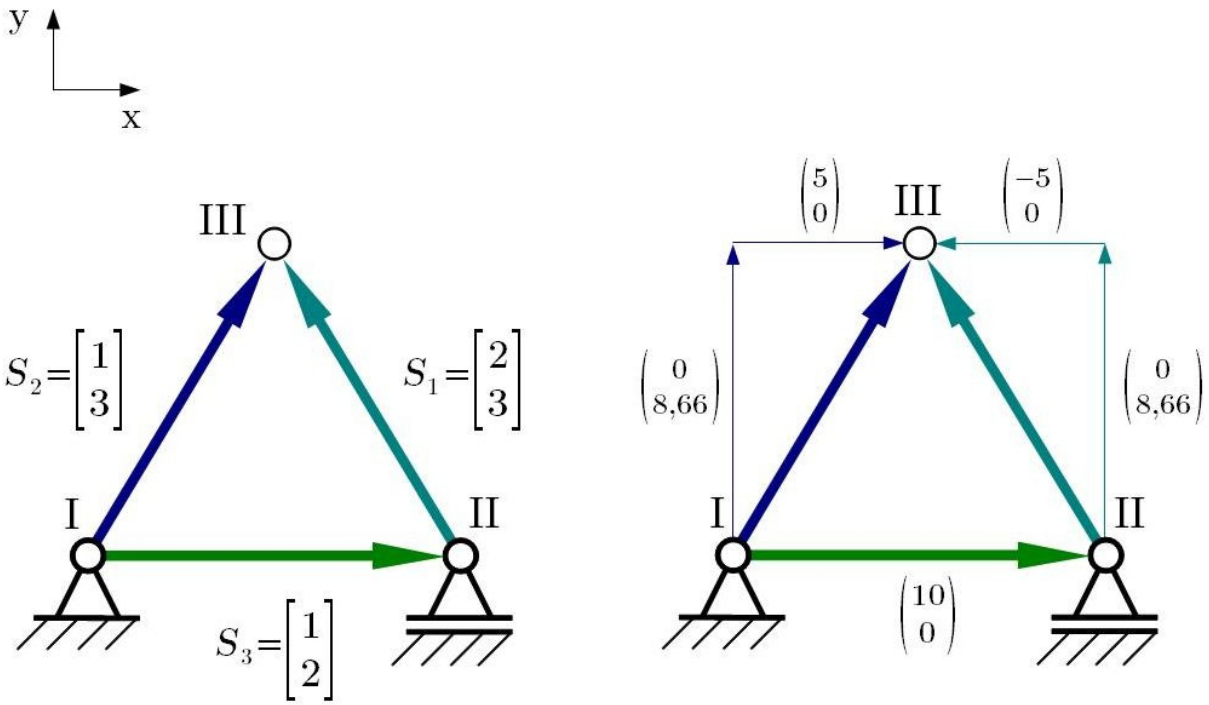


Abbildung 3.3: Konstruktion der Stabvektoren
Links: VERBINDUNG *–Einträge als Vektoren interpretiert,*
Rechts: Vektorlängen aus den Knotenpositionen

Der nächste Schritt ist, die Stabvektoren durch ihre Länge zu teilen, also zu normieren. In diesem konkreten Fall kommt das für jeden Vektor (oder präziser: für dessen

einzelne Komponenten) einer Division durch den Faktor 10 gleich. Diese normierten Stabvektoren nun sind die Stabkoeffizienten. Durch sie wird die vollständige geometrische Beziehung der drei Vektoren zum Koordinatensystem beschrieben. Abbildung 3.5 verdeutlicht die geometrische Beziehung exemplarisch an Stabvektor 1. Durch das rechtwinklige Dreieck und beiden bestimmten Ankatheten (die normierten Komponenten der Abstände von Knoten 3 zu Knoten 2) ist das Geometrieverhältnis vollständig bestimmt.

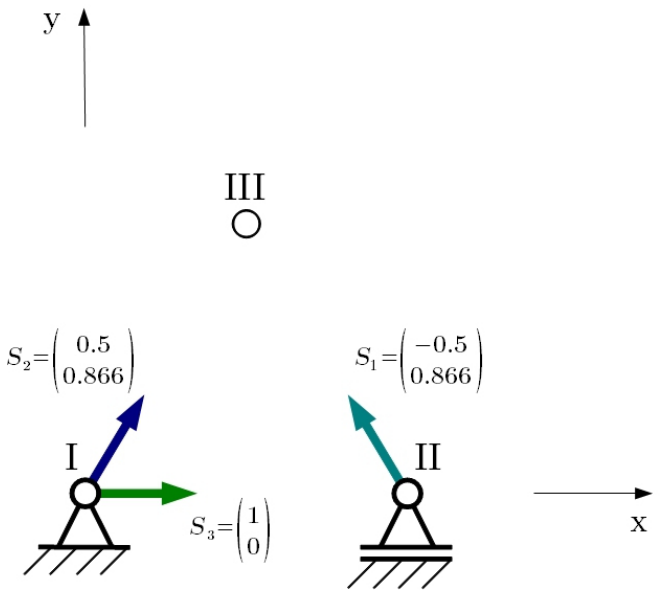


Abbildung 3.4: Normierte Stabvektoren (Stabkoeffizienten)

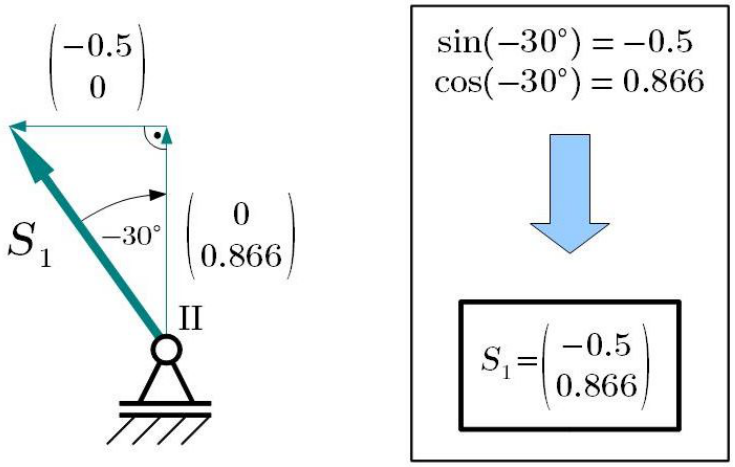


Abbildung 3.5: Geometriebestimmung von Vektor S_1 durch die Stabkoeffizienten

Es sind nun drei vektorielle Stabkoeffizienten vorhanden die im Programm unter der Matrix STABKOEFFIZIENTEN gespeichert werden.

$$Stabkoeffizienten = \begin{bmatrix} -0.5 & 0.5 & 1 \\ 0.866 & 0.866 & 0 \end{bmatrix}$$

Was nun noch fehlt, sind die entsprechenden Stabkräfte an den Knoten, die für jeden Stab in der unteren Zeile der VERBINDUNG–Matrix stehen. Anders ausgedrückt: Die jeweils gegenüberliegenden Vektoren von den bis hierher aufgestellten fehlen (vgl. Abbildung 2.2 und 3.4). Diese zu ermitteln ist ein Leichtes, da sie sich ja jeweils entlang der Vektorachsen befinden und identisch ausgerichtet sind. Die bereits vorhandenen Vektoren müssen also entlang ihrer Achse auf die entsprechenden Knoten verschoben werden. Einzig die Orientierung muss, um die Zugstabkonvention einzuhalten, an diesen Knoten umgekehrt werden, was einem Vorzeichenwechsel gleichkommt.

Diese drei weitere Vektoren werden im Programm an dieser Stelle nicht errechnet und auch nicht separat gespeichert, sondern später direkt mit negativem Vorzeichen den entsprechenden Stellen in der Koeffizientenmatrix zugewiesen. Der genaue Ablauf wird im nächsten Kapitel erläutert. Für das theoretische Verständnis genügt es an dieser Stelle zu erkennen, dass das Knotenpunktverfahren aus einem allgemeinen Statik–Problem heraus abgeschlossen ist. Sämtliche Stabkoeffizienten sind bestimmt, die Lagerkoeffizienten bereits vorhanden (die Einträge der Lagermatrix), und die äußeren Kräfte in der Matrix LASTEN gespeichert. Damit lässt sich nun das Gleichungssystem $\underline{\underline{A}} \cdot \vec{x} = \vec{b}$ aufstellen, was nur noch gelöst werden muss (s. S. 6f. bzw. S. 29f.).

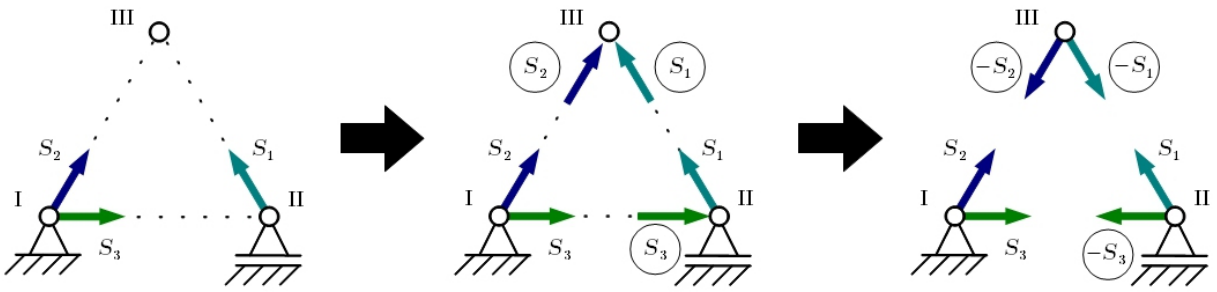


Abbildung 3.6: Herstellen der fehlenden Stabvektoren durch Projektion und Vorzeichenwechsel

3.3 Implementierung

3.3.1 Eingangsmatrizen

```
1  clear
2
3  % ------------------------------------------------------------------------
4  % Eingangsmatrizen, Konstruktion des Fachwerks
5  % ------------------------------------------------------------------------
6
7  % Äussere Kraft
8  F=100;
9
10 % Knotenkoordinaten, Koordinatensystem in Knoten 1
11 P1=[00,00];
12 P2=[10,00];
13 P3=[05,8.66];
14
15 % Knotenmatrix;
16 Knoten=[P1;P2;P3]';
17
18 % Knotenanfang und -ende der Stäbe
19 S1=[02,03];
20 S2=[01,03];
21 S3=[01,02];
22
23 % Stabmatrix
24 Verbindung=[S1;S2;S3]';
25
26 % Lagermatrix, Dimension enstpricht Knotenmatrix
27 Lager=zeros(size(Knoten));
28 Lager(:,1)=[1 1]';
29 Lager(:,2)=[0 1]';
30
31 % Lastmatrix, Dimension enstpricht Knotenmatrix
32 Lasten=zeros(size(Knoten));
33 Lasten(2,3)=-F;
```

Zuallererst löscht der Befehl *clear* in Zeile 1 zunächst einmal den Speicher. Damit sollen etwaige Reste von zuvor ausgeführten Rechnungen entfernt werden, die den Ablauf des Programms beeinträchtigen oder den Nutzer verwirren könnten. Damit ist auch sichergestellt, dass nach Ablauf des Programms sämtliche gespeicherte Werte eindeutig dem aktuellen Programmlauf zugeordnet werden können.

Anschließend wird der Betrag einer Kraft definiert, die später gemäß Angriffspunkt und Richtung in der Lastmatrix eingetragen wird:

$F = 100$

Im Folgenden werden die Knoten durch ihre Koordinaten definiert und zugeordnet. Das Koordinatensystem wurde in den Knoten *P1* gelegt, weshalb dieser auch die Koordinaten (0,0) hat. Diese Knotenkoordinaten werden dann spaltenweise in der Matrix KNOTEN zusammengefasst.

Für ein gleichseitiges Dreieck aus drei Stäben mit einer Stablänge von 10 und einem Koordinatensystem nach Abbildung 3.1, ergeben sich für die Knoten folgende Werte:

$$P_1=\begin{pmatrix}0\\0\end{pmatrix},\ P_2=\begin{pmatrix}10\\0\end{pmatrix},\ P_3=\begin{pmatrix}5\\8.66\end{pmatrix}\ \rightarrow\ Knoten=\begin{bmatrix}0 & 10 & 5\\0 & 0 & 8.66\end{bmatrix}$$

Es folgt die Zuweisung der Knoten für die die einzelnen Stäbe angebunden sind. Wie bereits erwähnt, wurde bei der Programmierung darauf geachtet, dass die Reihenfolge der Einträge beliebig erfolgen kann. In der Matrix VERBINDUNG werden diese Vektoren dann spaltenweise zusammengefasst. Entsprechend der Stäbe und ihrer angebundenen Knoten ergeben sich folgende Werte:

$$S_1=\begin{bmatrix}2\\3\end{bmatrix},\ S_2=\begin{bmatrix}1\\3\end{bmatrix},\ S_2=\begin{bmatrix}1\\2\end{bmatrix}\ \rightarrow\ Verbindung=\begin{bmatrix}2 & 1 & 1\\3 & 3 & 2\end{bmatrix}$$

Zum Erstellen der Matrix Lager, wird zunächst eine Nullmatrix entsprechend der Struktur der Matrix KNOTEN erstellt. Der Bezug auf die Knotenmatrix kommt daher, da ja die Lagerung sich nur an Knoten befinden kann; ergo muss die Lagermatrix die gleiche Struktur wie die Knotenmatrix haben. Der Aufbau über eine Nullmatrix ist hier sinnvoll, da bei etwaigen größeren Fachwerken sonst für jeden Knoten viele Nullen definiert werden müssten was viel Aufwand bedeuten würde. Auf diese hier angewandte Weise sind sämtliche Nullen durch die Nullmatrix schon da, und es bedarf nur noch den (wenigen) Einträgen für die Lager. Hier ist Knoten 1 in x−und y−Richtung fixiert, in der ersten Spalte von LAGER wird also in beiden Zeilen eine 1 eingetragen. An Knoten 2 befindet sich ein Loslager was in y−Richtung, nicht jedoch in x−Richtung Kräfte aufnimmt. Daher wird in der zweiten Spalte der oberen Zeile eine 0 und der unteren Zeile eine 1 zugewiesen:

$$Lager = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Die Lastenmatrix entspricht wieder der Struktur der Knotenmatrix, da äußere Kräfte laut Definition für ein Fachwerk nur auf Knoten wirken. Nachdem also eine Nullmatrix entsprechend der Struktur von KNOTEN erstellt wird, erfolgt die Zuordnung des zuvor eingeführten Kraftvektors. Unter Berücksichtigung von Ort, Koordinatenrichtung und Orientierung (also Spalte, Zeile und Vorzeichen) wird Kraft F eingetragen. Hier wirkt die Kraft auf Knoten 3 entgegen der y–Koordinatenrichtung, deshalb wird F mit negativem Vorzeichen in die zweite Zeile der dritten Spalte eingetragen:

$$Lasten = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -F \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -100 \end{bmatrix}$$

Es sei darauf hingewiesen, dass die Verwendung einer eigenen Variablen für einen Kraftbetrag optional ist. Eine direkte Zuweisung von Kraftbeträgen in die Lastenmatrix ist ebenso möglich und mit Vor– und Nachteilen verbunden. Generell gilt aber, dass unter Verwendung von Variablen die größte Sicherheit, Übersichtlichkeit und Flexibilität gewährleistet ist und daher auch empfohlen wird.

3.3.2 Statische Bestimmtheit

```
35  % Prüfen auf notwendige Bedingung für "statisch bestimmt" (Lagerreaktionen+Stäbe=2*Knoten)
36  if (sum(sum(Lager.^2))+size(Verbindung,2)) ~= 2*(size(Knoten,2))
37      error ('Das Tragwerk ist nicht statisch bestimmt, überprüfen Sie Ihre Eingaben')
38  end
```

Wie oben schon erwähnt, ist die notwendige Bedingung für ein statisch bestimmtes Fachwerk $Lagerreaktionen + Stäbe = 2 \cdot Knoten$. Der Wert der Summe der Lagerreaktionen ist die Summe aus Zeilen und Spalten der Matrix LAGER, also die Summe aller Einträge. Die Anzahl der Stäbe entspricht den Spalten der Matrix VERBINDUNG. Die Summe dieser beiden Werte also, wird mit dem Wert der doppelten Knotenanzahl verglichen. Sie entspricht der doppelten Anzahl der Spalten der Knotenmatrix. Sofern diese notwendige Bedingung verletzt wird, bricht das Programm mit einer Fehlermeldung ab.

Hier ist die Summe aller Einträge von LAGER gleich 3. Die Anzahl der Stäbe und damit der Spalten von VERBINDUNG beträgt ebenfalls 3. Die doppelte Anzahl der Spaltenanzahl von KNOTEN beträgt $2 \cdot 3 = 6$.

Da dieses Fachwerk also mit 3+3=6 statisch bestimmt ist, wird keine Fehlermeldung ausgegeben und das Programm wird fortgesetzt.

3.3.3 Stabkoeffizienten

```
40  % --------------------------------------------------------------------
41  % Geometriebestimmung
42  % --------------------------------------------------------------------
43
44  % Nullvektor und Nullmatrix; Dimension entsprechend der Stäbe, also von "Verbindung"
45  Stablaengen=zeros(1,size(Verbindung,2));
46  Stabkoeffizienten=zeros(size(Verbindung));
47
48  %Generieren der Stablängen und der Stabkoeffizienten
49  for j=1:size(Verbindung,2)
50      t=Verbindung(:,j);
51      Stablaengen(j)=norm(Knoten(:,t(2))-Knoten(:,t(1)));
52      Stabkoeffizienten(:,j)=(Knoten(:,t(2))-Knoten(:,t(1)))/Stablaengen(j);
53  end
```

Ein Nullvektor und eine Nullmatrix namens STABLAENGEN bzw. STABKOEFFIZIENTEN werden erstellt. Die Zahl der Stablängen ist gleich der Anzahl der Stäbe, weshalb die Größe von STABLAENGEN der Spaltenzahl von VERBINDUNG entspricht. Die Matrixstruktur von STABKOEFFIZIENTEN wird vollständig von VERBINDUNG übernommen. In diesem Fall ist VERBINDUNG eine 2x3 Matrix, weshalb Vektor STABLAENGEN drei Elemente und Matrix STABKOEFFIZIENTEN eine 2x3 Struktur zugewiesen bekommt.

Das Ermitteln der Stablängen, der Stabkoeffizienten, und deren Speichern der Einträge in die Matrix STABKOEFFIZIENTEN erfolgt durch eine *for*–Schleife. Die Anzahl der Durchläufe der Schleife richtet sich nach der Anzahl der Stäbe, weshalb sie entsprechend der Anzahl der Spalten von VERBINDUNG wiederholt wird (hier also drei mal).

Hilfsvektor *t* nimmt hierbei für jeden Durchlauf jeweils eine Spalte von VERBINDUNG an und verweist damit auf die beiden entsprechenden Knoten (und hierdurch auf deren

Koordinaten), an die der betreffende Stab angebunden ist. Dadurch ist vorgegeben, welche Knotenkoordinaten voneinander abgezogen und sodann normiert werden müssen um die Stablängen zu erhalten. Gespeichert werden diese für jeden Schleifendurchlauf in den Elementen des Vektors STABLAENGEN.

Die Stabkoeffizienten werden generiert, indem je Schleifendurchlauf nochmals beide Knotenkoordinaten voneinander abgezogen werden (dies entspricht den Stabvektoren, vgl. Abbildung 3.3), und das Ergebnis dann durch die zuvor ermittelte Stablänge geteilt wird.

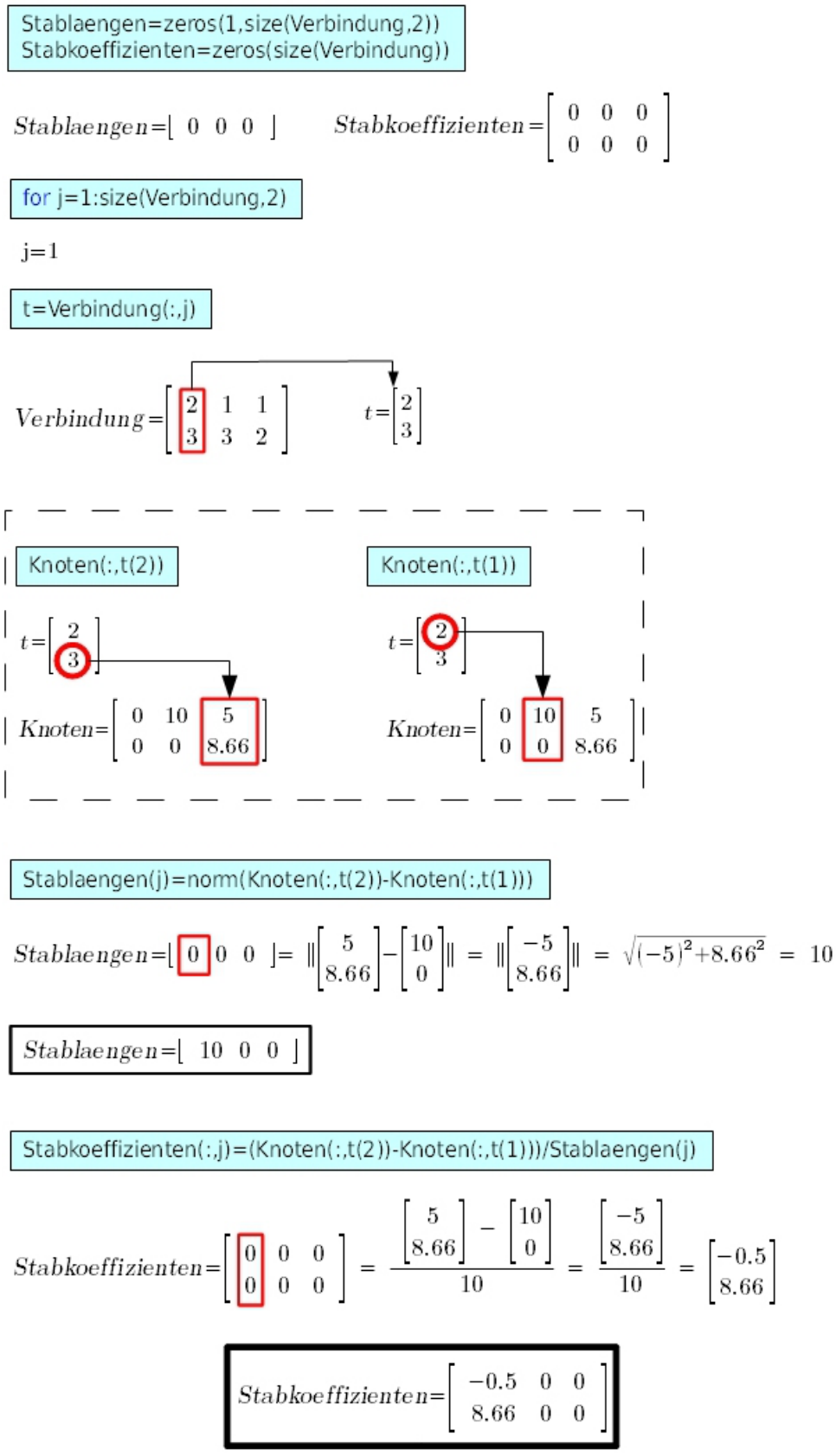


Abbildung 3.7: Schema des Algorithmus zum Errechnen der Stabkoeffizienten anhand Stab 1 (erster Durchlauf der Schleife, j=1)

Nachdem die Schleife abgeschlossen ist, bestehen STABLAENGEN und STABKOEFFIZIENTEN aus folgenden Einträgen:

$$Stablaengen = [10 \quad 10 \quad 10] \qquad Stabkoeffizienten = \begin{bmatrix} -0.5 & 0.5 & 1 \\ 0.866 & 0.866 & 0 \end{bmatrix}$$

3.3.4 Assemblierung der Stabkoeffizienten

```
55  % ------------------------------------------------------------------------
56  % Erstellen der Koeffizientenmatrix
57  % ------------------------------------------------------------------------
58
59  % Initialisieren der Koeffizientenmatrix (Zeilen: Doppelte Knotenzahl, Spalten: Stäbe,Lager)
60  Koeffmatrix=zeros(size(Verbindung,2)+sum(sum(Lager.^2)));
61
62  % Indexvektor
63  u=1:size(Knoten,1):size(Koeffmatrix,1)-1;
64
65  % Assemblierung der Stabkoeffizienten in die Koefizientenmatrix (d.h.Winkel werden eingetragen)
66  for j=1:size(Verbindung,2)
67      s=Stabkoeffizienten(:,j);
68      oberezeile=Verbindung(1,j);
69      unterezeile=Verbindung(2,j);
70      Koeffmatrix(u(oberezeile):u(oberezeile)+1,j)=+s;
71      Koeffmatrix(u(unterezeile):u(unterezeile)+1,j)=-s;
72  end
```

Bei der Assemblierung der Stabkoeffizienten geht es darum, den Stabkoeffizienten ihre vorbestimmte Stelle in der Koeffizientenmatrix zuzuweisen, wobei auf den bereits erwähnten Vorzeichenwechsel geachtet werden muss (vgl. S. 16f.). Dazu wird zunächst die Koeffizientenmatrix initialisiert, d.h. eine quadratische Nullmatrix erstellt. Zur Bestimmung der Dimension eignet sich entweder die doppelte Knotenzahl oder die Summe der zu ermittelnden Stab– und Lagerkräfte. Durch die Bedingung der statischen Bestimmtheit ist beides gleich. Hier wird letzteres verwendet, und zwar durch die Summe der Anzahl der VERBINDUNG–Spalten und der Gesamtsumme der Einträge von LAGER. Die Dimension der Nullmatrix ist in diesem Fall also 6.

Im Folgenden wird ein Indexvektor u erstellt. Der erste Eintrag ist eine 1. In Abständen entsprechend der Zeilenanzahl von KNOTEN (also 2) werden die weiteren Einträge bis zur Dimension von KOEFFMATRIX abzüglich 1 erstellt. Der Indexvektor u hat den Zweck, auf die erste Zeile des entsprechenden Knotens (x–Koordinate) in der

Koeffizientenmatrix zu verweisen, die auf diesen abbildet. Indexvektor *u* lautet hier also: *u*=[1 3 5]

Nun folgt die Schleife zur Zuordnung der Stabkoeffizienten in die Koeffizientenmatrix. Hier wird die Schleife drei mal wiederholt, da im Folgenden für die ersten drei Spalten von KOEFFMATRIX (die für die drei Stäbe stehen) die Stabkoeffizienten eingesetzt werden. Schleifenvariable *j* verweist also auf die entsprechende Spalte, Hilfsvektor *u* dient dem Verweis auf die entsprechende Zeile von KOEFFMATRIX. Für jeden Durchlauf, und damit für jeden Stab, werden also die entsprechenden Stabkoeffizienten und ihr negatives Pendant in den entsprechenden Zeilen der spezifischen Spalte von KOEFFMATRIX eingefügt.

Ein weiterer Hilfsvektor *s* wird eingeführt, der die *j*-te Spalte von STABKOEFFIZIENTEN speichert. *oben* und *unten* werden ebenfalls in der Funktion als Hilfsvariablen definiert und speichern jeweils den oberen bzw. unteren Eintrag der *j*-ten Spalte von VERBINDUNG.

Der Indexvektor *u* gibt unter Rückgriff auf die Variable *oben* für jeden Durchlauf *j* diejenige Zeile von KOEFFMATRIX an, in der der obere Eintrag des Koeffizientenvektors *s* einzusetzen ist. Zusammen mit dem darauf folgenden Eintrag (folgende Zeile) sind die beiden für *s* benötigten und vorbestimmten Einträge in KOEFFMATRIX identifiziert. Die Zuordnung der Einträge für die Stabkoeffizienten mit einem Vorzeichenwechsel erfolgt mit dem Unterschied, dass der Rückgriff von *u* hierbei auf *unten* erfolgt, und *s* negativ eingetragen wird. Es werden demnach insgesamt vier Einträge pro Spalte und Durchlauf vorgenommen, da, wenn das Fachwerk unbeweglich konstruiert wurde, jeder Stab an zwei Knoten angebunden ist und für jeden dieser Knoten zwei Koeffizienten pro Stab einzutragen sind.

Im ersten Durchlauf wird mit *u(oben)*=3 und *u(oben)*+1=4 die Zeilen 3 und 4 ermittelt. Somit wird in der ersten Spalte in Zeilen 3 und 4 der erste Koeffizientenvektor eingetragen. *u(unten)*=5 und *u(unten)*+1=6 geben die Einträge für den Knoten an, für die ein Vorzeichenwechsel vollzogen werden muss (für jeden Durchlauf der untere Eintrag von VERBINDUNG). Am Ende sind sämtliche Stabkoeffizienten eingetragen.

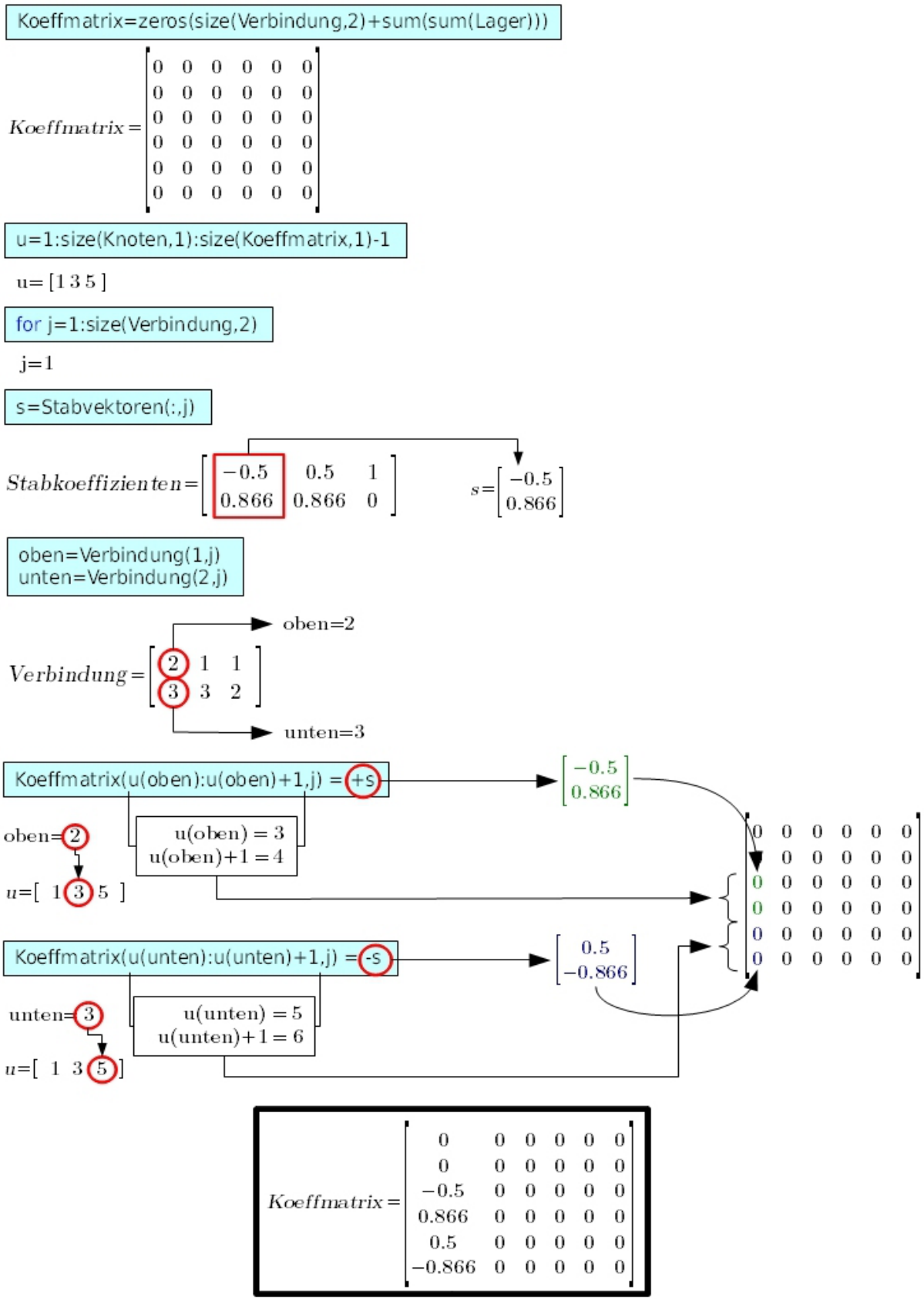


Abbildung 3.8: Schema des Algorithmus zur Assemblierung der Stabkoeffizienten in die Koeffizientenmatrix anhand Stab 1 (erster Durchlauf der Schleife, j=1)

Nachdem die Schleife abgeschlossen ist, lautet KOEFFMATRIX wie folgt:

$$Koeffmatrix = \begin{bmatrix} 0 & 0{,}5 & 1 & 0 & 0 & 0 \\ 0 & 0{,}866 & 0 & 0 & 0 & 0 \\ -0{,}5 & 0 & -1 & 0 & 0 & 0 \\ 0{,}866 & 0 & 0 & 0 & 0 & 0 \\ 0{,}5 & -0{,}5 & 0 & 0 & 0 & 0 \\ -0{,}866 & -0{,}866 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.3.5 Assemblierung der Lagerkoeffizienten

```
74  % Assemblierung der Lagerkoeffizienten in die Koeffizientenmatrix
75  i=1;
76  for j=1:size(Koeffmatrix,2)
77     if Lager(j)~=0
78        Koeffmatrix(j,i+size(Verbindung,2))=Lager(j);
79        i=i+1;
80     end
81  end
```

Die Assemblierung der Lagerkoeffizienten ist weniger aufwändig. Der Fokus liegt nun auf den verbliebenen drei hinteren Spalten der Koeffizientenmatrix, die für die Lagerkoeffizienten bestimmt und bis hierher noch gleich 0 sind.

Zunächst wird eine Indexvariable *i* erstellt. Ihr wird zunächst den Wert 1 zugewiesen. Die Assemblierung erfolgt anschließend über eine *for*–Schleife. Sie wird entsprechend der Dimension von KOEFFMATRIX wiederholt, hier also 6 mal. Dies bezieht sich allerdings nicht auf die einzutragenden Spalten (wie bei der Assemblierung der Stabkoeffizienten), sondern auf die Matrix LAGER. Da die Lagermatrix ein Abbild der Knotenmatrix darstellt, sie also dieselbe Struktur haben, und diese Struktur sich wiederum in der Anzahl der Zeilen der Koeffizientenmatrix wiederfindet, kann diese hier analog als Endzahl verwendet werden. Anders ausgedrückt: Die Summe der Elemente von Lager entspricht jeweils der Anzahl der Zeilen und Spalten von KOEFFMATRIX, weshalb es irrelevant ist ob sich die Endzahl auf das eine oder andere bezieht.

Die Schleife hat den Zweck, mittels einer *if*–Abfrage zu prüfen, ob der entsprechende

Eintrag ein nicht−Null Eintrag ist. Hierbei wird in aufsteigender Reihenfolge der Zeilen und Spalten auf die Elemente von LAGER zugegriffen. Das heißt, für den ersten Durchlauf wird der Eintrag der ersten Zeile und ersten Spalte überprüft, für den zweiten Durchlauf die zweite Zeile der ersten Spalte, anschließend der erste Eintrag der zweiten Spalte usw. Sofern der Eintrag gleich 0 ist, demnach hier kein Lager vorhanden ist, passiert nichts und die Schleife beginnt von vorne zu prüfen für den nächsten Eintrag. Für den Fall, dass ein nicht−Null Eintrag vorhanden ist, wird dieser Wert in genau *die* Zeile von KOEFFMATRIX eingetragen, die *j* entspricht. Da die Abfrage der Elemente der Lagermatrix entsprechend der Zeilenanordnung der Koeffizientenmatrix erfolgt (Knoten 1 in x−Richtung, Knoten 1 in y−Richtung, Knoten 2 in x−Richtung usw.), verweist Schleifenvariable *j* immer auf die Zeile der Koeffizientenmatrix, zu dem das momentan abgefragte Element der Lagermatrix gehört.

$$Lager = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \qquad Lager(j) = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Abbildung 3.9: Zugriffsreihenfolge von Schleifenvariable j ∈ [1;6]

Bleibt zu klären, wie die Spalten der Koeffizientenmatrix den zu speichernden Lagerkoeffizienten zugeordnet werden. Hierfür ist Index *i* zuständig. Die betreffende Spalte wird durch die Summe aus der Anzahl der Stäbe und *i* definiert. Die Anzahl der Stäbe beträgt 3; *i* ist bis der erste Lagerkoeffizient gefunden und eingetragen wird gleich 1. Die Summe aus 3 und 1 ist 4. Damit wird der erste gefundene Lagerkoeffizient in die erste leere und für die erste Lagerkraft vorgesehene Spalte eingetragen, die hier die vierte ist. Sofern dies geschehen ist, also ein Lagerkoeffizient gefunden und eingetragen wurde, wird zu *i* der Wert 1 addiert. *i* weist damit für den nächsten gefundenen Lagerkoeffizienten die fünfte Spalte zu usw.

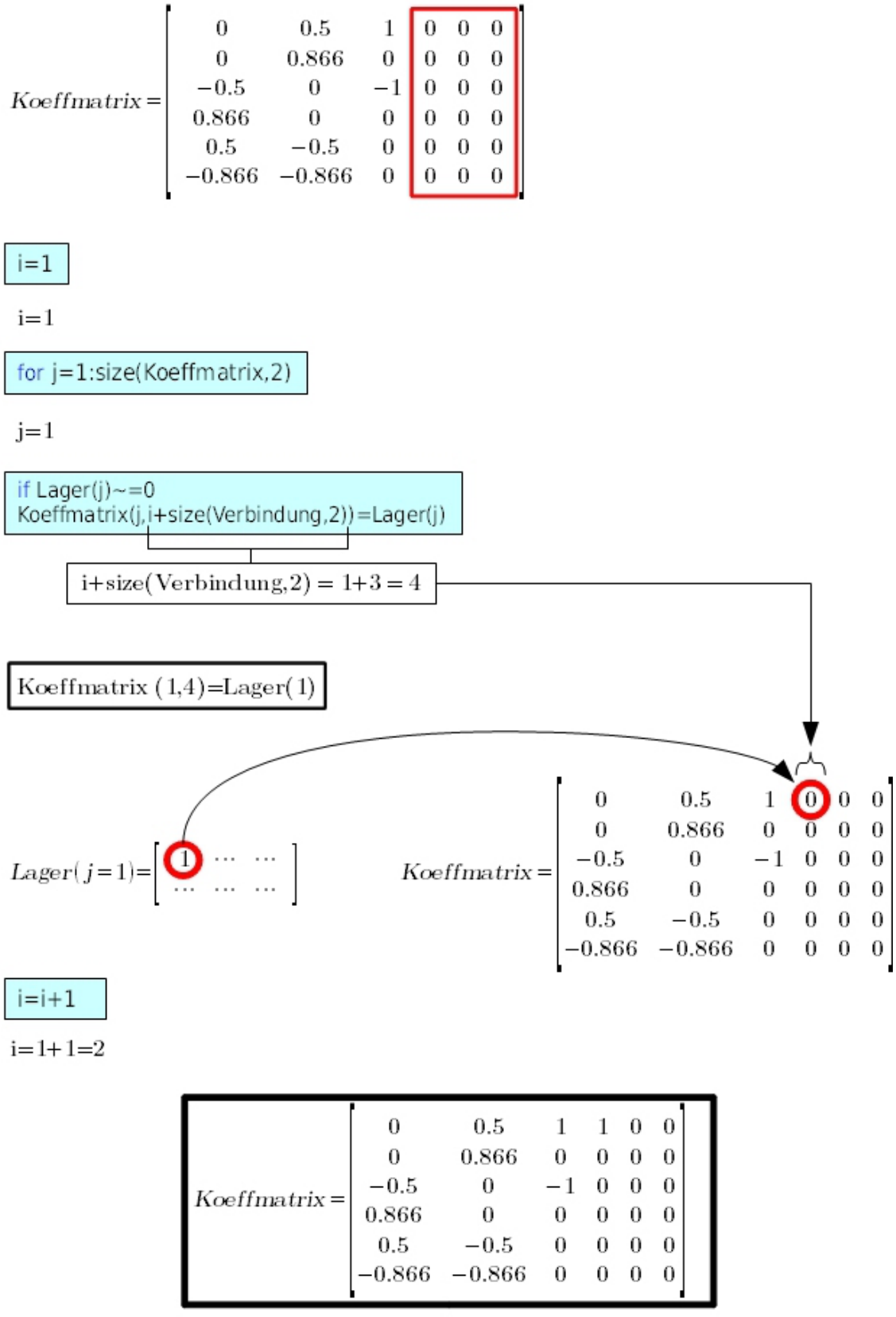


Abbildung 3.10: Schema des Algorithmus zur Assemblierung der Lagerkoeffizienten in die Koeffizientenmatrix anhand der Lagerkraft A_x (j=1)

Damit ist die Assemblierung aller Koeffizienten in die Koeffizientenmatrix abgeschlossen. Sie lautet nun:

$$Koeffmatrix = \begin{bmatrix} 0 & 0{,}5 & 1 & 1 & 0 & 0 \\ 0 & 0{,}866 & 0 & 0 & 1 & 0 \\ -0{,}5 & 0 & -1 & 0 & 0 & 0 \\ 0{,}866 & 0 & 0 & 0 & 0 & 1 \\ 0{,}5 & -0{,}5 & 0 & 0 & 0 & 0 \\ -0{,}866 & -0{,}866 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.3.6 Kinematische Bestimmtheit

```
83  % Prüfen auf hinreichende Bedingung für statische Bestimmtheit
84  if det(Koeffmatrix)==0
85     error ('Koeffizientenmatrix nicht eindeutig lösbar. Tragwerk ist beweglich. Überprüfen Sie Ihre Eingaben.')
86  end
```

Die Unbeweglichkeit im Sinne der Statik wird mit dem Errechnen der Determinante der Koeffizientenmatrix überprüft. Ist die Determinante gleich 0, bricht das Programm mit dem Hinweis ab, dass das Fachwerk beweglich konstruiert wurde und das Gleichungssystem so nicht eindeutig lösbar ist.

3.3.7 Lösen des Gleichungssystems

```
88  % ------------------------------------------------------------------------
89  % Lösen des Gleichungssystems
90  % ------------------------------------------------------------------------
91
92  % Lastenmatrix wird zu einem Vektor überführt und negativ (Ax+b=0 -> Ax=-b)
93  Lasten=-Lasten(:);
94
95  % Lösen des Gleichungssystems
96  Forces=linsolve(Koeffmatrix, Lasten);
```

Der Matlab–Löser eines linearen Gleichungssystems löst Gleichungssysteme der Form $\underline{\underline{A}} \cdot \vec{x} = \vec{b}$ mit $\underline{\underline{A}}$ =Koeffizientenmatrix, $\vec{x}$ =Unbekannte Kräfte und $\vec{b}$ =Äußere Lasten. Um abschließend zu dieser Form zu kommen, wird zunächst die Matrix LASTEN durch den Matlab–eigenen Befehl *(:)* zu einem Vektor umgewandelt. Dies geschieht derart, dass die Matrix spaltenweise übereinander als Vektor angeordnet wird. Auf die

erste Spalte mit ihren Elementen folgt die zweite usw. Dieser Vektor bekommt dann ein negatives Vorzeichen zugewiesen, da die äußeren Lasten für diese Art von Gleichungssystemen auf die rechte Seite des Gleichheitszeichens müssen. Der *Linsolve*–Operator löst nun das lineare Gleichungssystem unter Übergabe der Matrix KOEFFMATRIX und des Vektors LASTEN.

Die zu ermittelnden Stab– und Lagerkräfte werden dann vektoriell unter den Namen FORCES gespeichert. In diesem Fall beinhaltet er sechs Einträge, da es jeweils drei zu ermittelnde Stab– und Lagerkräfte gibt. Die Reihenfolge seiner Einträge entspricht jener der Anordnung der Stäbe in der VERBINDUNG–Matrix durch die Spalten gefolgt von den Lagerkräften in der Anordnung entsprechend den Einträgen der Matrix LAGER. Der erste Eintrag von FORCES gibt also die Stabkraft wieder, die dem Stab der ersten Spalte in VERBINDUNG entspricht. Hinter in diesem Fall dritten und letzten Eintrag einer Stabkraft folgen die Lagerkräfte, und zwar in aufsteigender Reihenfolge zunächst spalten– und dann zeilenweise wie die Lagerkoeffizienten in der Lagermatrix angeordnet sind (vgl. Abbildung 3.9). In diesem Fall ist also der vierte Eintrag von FORCES die Lagerkraft an Knoten 1 in x–Richtung, der zweite Eintrag Lagerkraft an Knoten 1 in y–Richtung, der letzte Eintrag Lagerkraft an Knoten 2 in y–Richtung.

Das Ergebnis der analytischen Berechnung der Kräfte, gespeichert als Vektor FORCES lautet:

$$Forces = \begin{bmatrix} -57.7367 \\ -57.7367 \\ 28.8684 \\ 0 \\ 50.0000 \\ 50.0000 \end{bmatrix}$$

3.4 Verifizieren des Algorithmus

Zur Verifikation des Algorithmus wird er an zwei weiteren und deutlich aufwendigeren statisch bestimmten Fachwerken getestet, zu welchen die Lösungen bekannt sind. Das erste ist dem Buch *Technische Mechanik, Band 1: Statik* von Dietmar Gross u.A. auf den Seiten 154/155 entnommen (Abbildung 3.11). Das zweite Fachwerk befindet sich in *Technische Mechanik, Computerunterstützt* von Jürgen Dankert u.A. auf Seite 80 und 652 (Abbildung 3.12).

Die Musterlösungen können der jeweiligen Literatur auf den genannten Seiten entnommen werden. In beiden Fällen stimmen sie mit den Lösungen nach dem hier entwickelten Quellcode überein. Daher kann davon ausgegangen werden, dass der Algorithmus universell für ebene, statisch bestimmte Stabtragwerke angewendet werden kann.

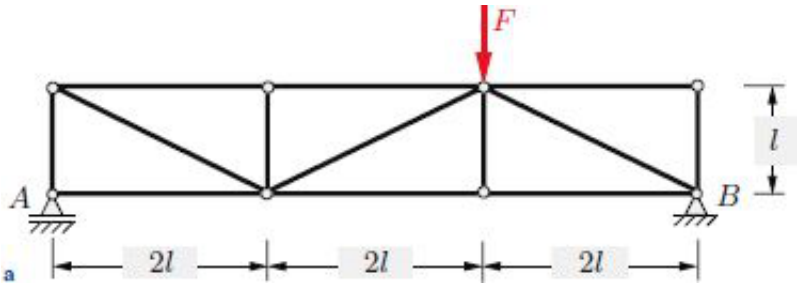


Abbildung 3.11: Fachwerk nach Gross

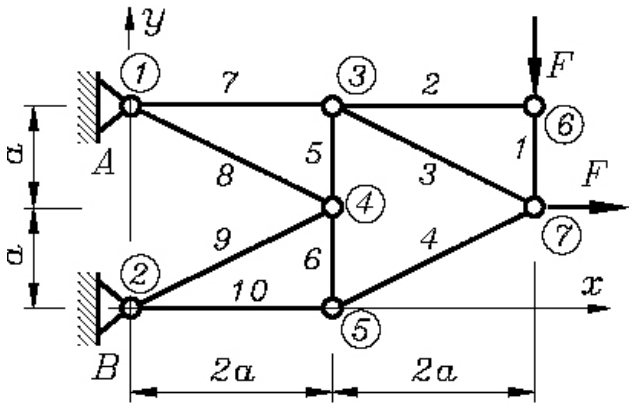


Abbildung 3.12: Fachwerk nach Dankert

3.5 Implementieren der Evolutionsstrategie

3.5.1 Zielfunktion und Materialparameter

Da das vorliegende Fachwerk relativ simpel ist, wird eine übliche (1+1)–Evolutionsstrategie verwendet. Als Vorlage dient das *Minimalprogramm* nach Vorlesung 5, Folie 25 aus Prof. Rechenbergs Lehrveranstaltung *Evolutionsstrategie I* zum Wintersemester 2011/12 an der TU Berlin.[9]

Das Ziel ist eine Gewichtsoptimierung, also eine Herabsenkung des Gesamtgewichts indem die Koordinaten von bestimmten Knoten variiert werden. Die Masse der Lager wird hierbei allerdings nicht berücksichtigt. Deshalb bestimmen ausschließlich die Stäbe mit ihren Parametern Länge L (entspricht STABLAENGEN), Profilfläche A und Dichte ρ das Gewicht. Die Zielfunktion zur Minimierung des Gesamtgewichts lautet:

$$Masse = \sum_{i=1}^{3} \rho \cdot L_i \, A_i \quad \Rightarrow Minimum$$

Da die Profilflächen u.A. aus den zulässigen Spannungswerten des Materials berechnet werden (vgl. Kapitel 3.5.2), müssen Materialwerte definiert, bzw. ein Material ausgewählt werden. Hier wird der Baustahl *S235* verwendet, den folgende Materialwerte[10] kennzeichnen:

$$E = 210 \cdot 10^9 \frac{N}{m^2}, \quad \rho = 7900 \frac{kg}{m^3}, \quad R_e = \sigma_{dF} = 235 \cdot 10^6 \frac{N}{m^2}, \quad \sigma_{zzul} = \frac{R_e}{S}, \quad \sigma_{dzul} = \frac{\sigma_{dF}}{S}$$

mit

E = Elastizitätsmodul, ρ = Dichte, R_e = Streckgrenze, σ_{dF} = Quetschgrenze,
σ_{zzul} = Zulässige Zugspannung, σ_{dzul} = Zulässige Druckspannung,
S = Sicherheitskoeffizient.

Als Sicherheit gegen Fließen wird hier 1,5 gewählt → $S = 1.5$

9 http://www.bionik.tu-berlin.de/institut/skript/E1-12Fo5.ppt
10 Aus: Ulrich Fischer u.A.: Tabellenbuch Metall, Haan-Gruiten 2008, S.44-46 sowie S.117

3.5.2 Zuweisen der Profilflächen

Einem Stab wird entsprechend Betrag und Richtung der Belastung durch die maximal zulässigen Druck– und Zugspannungswerte des Materials immer eine minimal zulässige Profilfläche zugewiesen. Für einen Zugstab geschieht dies durch die zulässige Zugspannung des Materials. Für einen Druckstab geben die zulässige Druckspannung bzw. die kritische Knickkraft die Profilflächen vor (A_{dzul} bzw. A_{kzul}). Bei Unterschreiten einer dieser Flächen, versagt der Stab. Daher werden zunächst beide Profilflächen ermittelt und dann miteinander verglichen. Anschließend wird die *größere* von beiden dem Druckstab zugewiesen, da nur sie die Einschränkung, die sich aus der jeweils anderen Gleichung ergibt, abdeckt. Würde die kleinere von beiden Flächen zugewiesen werden bedeutete dies, dass die jeweils andere Fläche unterschritten und damit Versagen eintreten würde. Mit F_i=*Forces(i)* und L_i=*Stablaengen(i)* gilt:

$$A_i=\begin{cases} A_{kzul,i}, & F_i<0 \ \cap \ A_{kzul,i}>A_{dzul,i} \\ A_{dzul,i}, & F_i<0 \ \cap \ A_{dzul,i}\geq A_{kzul,i} \\ A_{zzul,i}, & F_i>0 \end{cases}$$

Dabei ist :

$$A_{zzul,i}=\frac{F_i}{\sigma_{zzul}}, \quad A_{dzul,i}=\frac{|F_i|}{\sigma_{dzul}}, \quad A_{kzul,i}=\sqrt{\frac{4\cdot|F_i|\cdot L_i^2\cdot S}{\pi\cdot E}}$$

Da eine Druckkraft laut Konvention (und so auch hier) in vektorieller Form negativ ist, muss dies durch die Betragsstriche berücksichtigt werden, um Flächen zu erhalten, die positiv sind. A_{kzul} wird aus der Gleichung der zulässigen Knickkraft[11] hergeleitet:

$F_{kzul}=\frac{\pi^2\cdot E\cdot I}{L^2\cdot S}$. Mit $I=\frac{\pi\cdot r^4}{4}$ für kreisrunde Vollprofile und $r^2=\frac{A}{\pi}$ ist $I=\frac{A^2}{4\cdot\pi}$ und damit $F_{kzul}=\frac{\pi\cdot E\cdot A^2}{4\cdot L^2\cdot S}$.

Diese Gleichung gibt die zulässige Knicklast für eine gegebene Profilfläche an. Stellt man die Gleichung um, wird die kritische Fläche für gegebene (und betragsmäßig

11 Ulrich Fischer u.A.: Tabellenbuch Metall, Haan–Gruiten 2008, S.46

eingehende) Kraft ermittelt:

$$A_{kzul,i} = \sqrt{\frac{4 \cdot |F_i| \cdot L_i^2 \cdot S}{\pi \cdot E}}$$

In der Ausgangslage sind die Stäbe 1 und 2 Druckstäbe, und Stab 3 ein Zugstab (vgl. Kräftevektor FORCES, S.30).

Es ist ferner $A_{kzul,1;2} = 2.29 \cdot 10^{-4} \mathrm{m}^2 > A_{dzul,1;2} = 3.69 \cdot 10^{-7} \mathrm{m}^2$.

Damit sind die Profilflächen:

$A_1 = A_{kzul,1} = 2.29 \cdot 10^{-4} \mathrm{m}^2$, $A_2 = A_{kzul,2} = 2.29 \cdot 10^{-4} \mathrm{m}^2$ und $A_3 = A_{zzul,3} = 1.84 \cdot 10^{-7} \mathrm{m}^2$.

Das Gewicht welches vor der Optimierung besteht lautet somit:

$$Gewicht := qe = 7900 \frac{kg}{m^3} \cdot [10\mathrm{m} \quad 10\mathrm{m} \quad 10\mathrm{m}] \cdot \begin{bmatrix} 2.29 \cdot 10^{-4} m^2 \\ 2.29 \cdot 10^{-4} m^2 \\ 1.84 \cdot 10^{-7} m^2 \end{bmatrix} = 36.22\, kg$$

Dieses Gewicht des noch unveränderten Fachwerks wird nun außerdem unter der Variablen *Anfangsgewicht* gespeichert. Dies dient dazu, das später ermittelte optimale Gewicht mit dem ursprünglichen vergleichen zu können.

3.5.3 Generationenschleife

Die Generationenschleife ist der Kern der Evolutionsstrategie, in der sich die Varriierung der Variablen (hier Knotenkoordinaten), der Vergleich der Zielfunktionswerte und die Selektion der jeweils besseren Variablenmenge wiederholt.

Vor Eintritt in die Generationenschleife wird aber zunächst die Knotenmatrix unter einer weiteren Variablen „KN“ gespeichert, deren Inhalt im Folgenden teilweise variiert wird. Matrix KNOTEN nimmt die Funktion des Elters ein, d.h. sie speichert die Koordinaten für die ein Evolutionsschritt erfolgreich war. Anschließend wird die Variablenzahl und die Startschrittweite definiert. Analog zum Speichern der Matrix

KNOTEN unter einer neuen Variablen, werden danach FORCES, A und STABLAENGEN unter FN, AN und STABLAENGEN_N zwischengespeichert. Bei Programmabbruch bzw. –ende können so wichtige Werte, die aus dem letzten und aktuellen Elter hervorgingen abgerufen werden. Sie dienen damit der Information und der Kontrolle für den Nutzer, erfüllen aber keine Funktion im Programm. Da die Werte der Ausgangslage später auch in die Aufzeichnung der grafischen Ausgabe mitaufgenommen werden sollen (als 0–te Generation), werden dazu das Anfangsgewicht und die Startschrittweite dem *semilogy*–Befehl übergeben (näheres zur grafischen Ausgabe s.u.)

Mit Eintritt in die Generationenschleife werden als erstes die vorgegebenen Knotenkoordinaten variiert (hier die y–Koordinate von Knoten 3). Folgend werden ausgehend der neuen Knotenkoordinaten das neue Gewicht ermittelt, indem (in genau dieser Reihenfolge) Stablängen, Stabkoeffizienten, Koeffizientenmatrix, Kräfte und Profilflächen neu errechnet bzw. erstellt werden. Der Algorithmus ist hierbei derselbe wie in Kapitel 3.3 und 3.5.2. Alle diese Werte werden dabei überschrieben; lediglich bei einem Evolutionsfortschritt werden die wichtigsten von ihnen zwischengespeichert (s.o.). Das neue Gewicht wird dann unter der Variablen *qn* („Qualität neu“) gespeichert.

Nachdem das neue Gewicht ermittelt wurde, wird durch eine *if*–Abfrage das neue Gewicht und das des Elters verglichen. Sofern das neue Gewicht kleiner ist, bedeutet das einen Fortschritt in der Gewichtsoptimierung. Somit wird die neue Knotenmatrix als Elter gespeichert (KNOTEN=KN) und das neue Gewicht zum Maßstab für das der folgenden Generation (*qn*=*qe*). Die Einträge von FN, AN und STABLAENGEN_N werden, da ein neuer Elter definiert wurde, überschrieben. Ferner wird die Schrittweite um den Faktor 1,3 vergrößert.[12] Sofern das neue Gewicht *nicht* kleiner ist, bleibt der vorige Elter bestehen. In dem Fall wird lediglich die Schrittweite um den Faktor $\sqrt[4]{1{,}3}$ verringert.[13]

Nach dem Abgleich des Gewichts folgen die Zeilen der grafischen Ausgabe. Sie dient dazu, den Verlauf der Optimierung aufzuzeichnen um den Prozess nachvollziehen und

12 Siehe http://www.bionik.tu–berlin.de/institut/skript/E1–12Fo5.ppt, Folie 25
13 ebda.

auswerten zu können. Dies geschieht einerseits durch einen logarithmisch skalierten Plot der Werte *Gewicht* (blau) und *Schrittweite* (gelb), und andererseits durch eine bildliche Visualisierung des Fachwerkmodells, die seine Form darstellt. Die Skalierung der Achsen wird in beiden Fällen von Matlab während des Prozesses permanent angepasst, da sich die Werte z.T. stark ändern können. Schließlich und endgültig wird dem Nutzer das Resultat des Optimierungsprozesses (im *Command Window* von Matlab) ausgegeben. Dazu werden das Gewicht vor und nach der Optimierung miteinander ins Verhältnis gesetzt um die prozentuale Gewichtsersparnis zu ermitteln.

3.6 Durchführen der Optimierung

3.6.1 Ergebnisse

Anmerkung: Es sei nochmals darauf hingewiesen, dass alle Eingangsgrößen einheitenlos eingegeben, und daher auch keine Einheiten ausgegeben werden. Zur besseren Veranschaulichung werden die folgenden Ergebnisse aber in den Einheiten *Newton*, *Meter* und *Kilogramm* angegeben werden.

Für die Testläufe wird die y−Koordinate von Knoten 3 (d.h. Variablenzahl $v=1$) für die Dauer von 200 Generationen variiert. In den ersten Tests wurden jeweils Startschrittweiten von $d\leq 1\text{m}$ definiert. In diesen Fällen konvergierte der Knoten auf die Höhe von etwa 2.51m, wodurch das Fachwerk ein Gesamtgewicht von 15.79kg annahm. In Bezug auf das Anfangsgewicht von 36.22kg entspricht dies einer **Gewichtsersparnis von etwa 56.39%**. Bei Suchradien von $d\in[2;4]$ war zu beobachten, dass das Konvergenzverhalten manchmal wechselte, und zwar zu einer Höhe von etwa −240.81m. Bei Tests mit größeren Startschrittweiten konvergierte der Knoten deutlich öfter und schnell häufiger dorthin. Bei Tests ab etwa $d\geq 9\text{m}$ wurde nur noch dieses Konvergenzverhalten beobachtet. In dieser Position nahm das Fachwerk eine Masse von etwa 3.64kg an, was in Relation zu dem Anfangsgewicht von 36.22kg einer **Gewichtsersparnis von etwa 89.94%** entspricht.

Es wurden somit zwei Optima gefunden, und zwar ein lokales bei einer Höhe von etwa +2.51m und ein globales bei einer Höhe von etwa −240.81m.

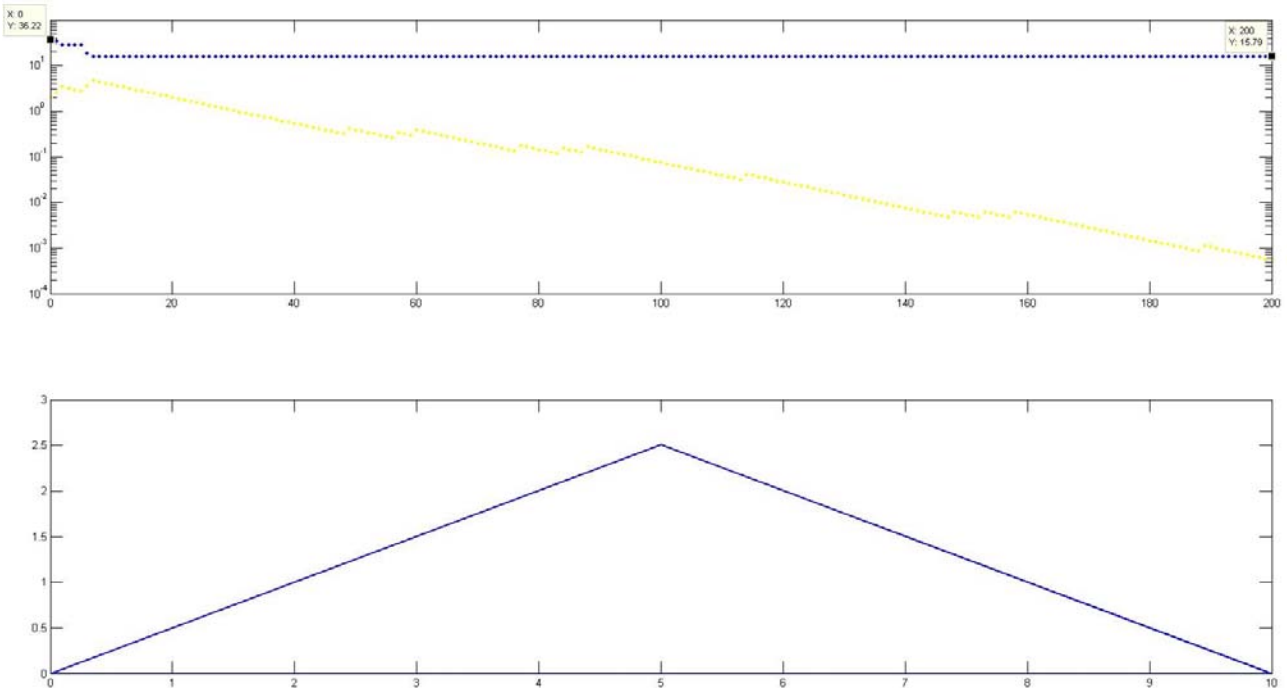


Abbildung 3.13: Testlauf mit Konvergenz zum lokalen Optimum, d=2

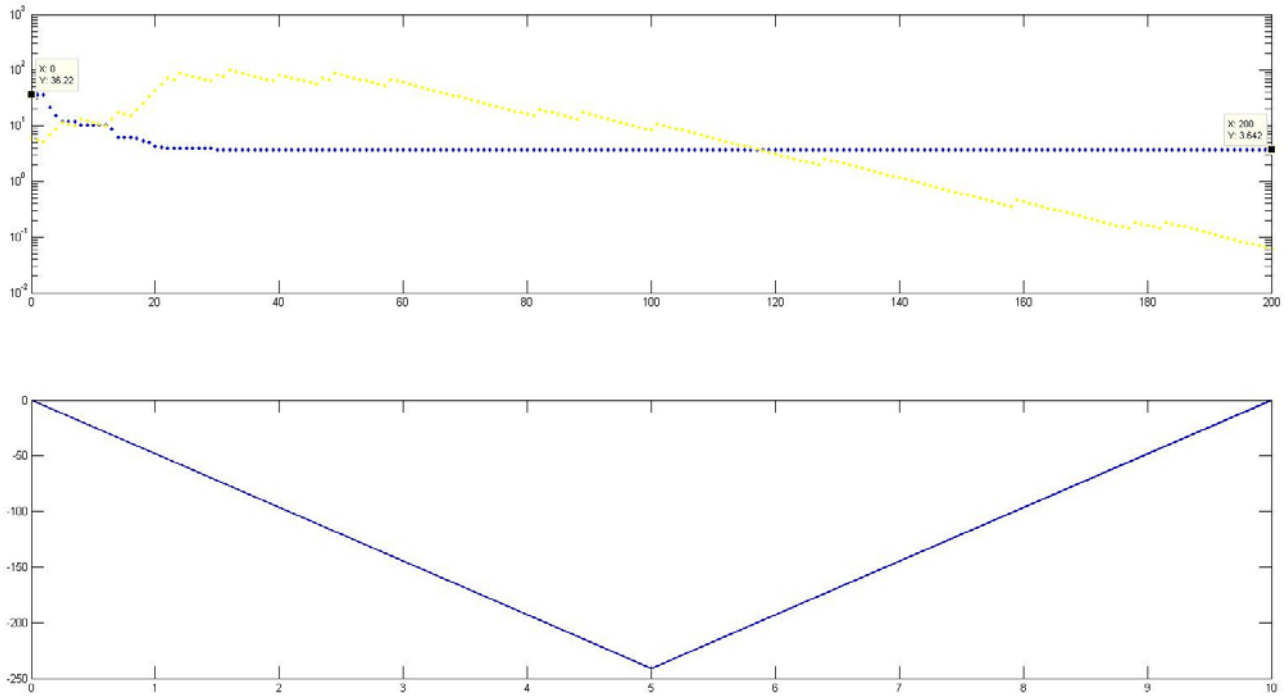


Abbildung 3.14: Testlauf mit Konvergenz zum globalen Optimum, d=6

3.6.2 Wertung

Es stellt sich nun die Frage, weshalb das Konvergenzverhalten auf diese Weise ausfällt, warum Knoten 3 ausgerechnet in diese Positionen konvergiert. Insbesondere das globale Optimum wirkt im ersten Augenblick befremdlich ob des großen Abstandes zum Ausgangszustand.

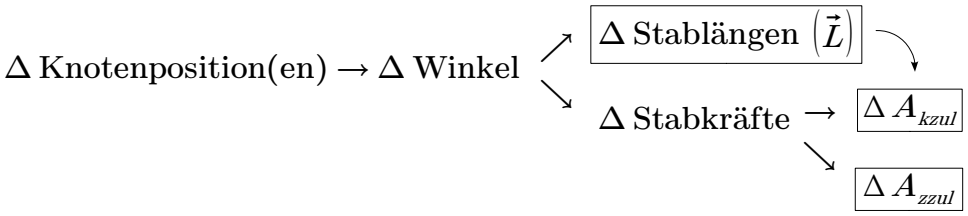
Zur Erinnerung, die Qualitätsfunktion lautet:

$$Masse = \sum_{i=1}^{3} \rho \cdot L_i \, A_i \Rightarrow Minimum$$

oder

$$Masse = \rho \cdot [A(1) \;\; A(2) \;\; A(3)] \begin{bmatrix} Stablaengen(1) \\ Stablaengen(2) \\ Stablaengen(3) \end{bmatrix} \Rightarrow Minimum$$

Die Komponenten der Qualitätsfunktion, die für die Optimierung von Bedeutung sind, sind lediglich die Stablängen und die Querschnittsflächen, da die Dichte einheitlich und konstant bleibt. Die Aufgabe der Gewichtsoptimierung besteht damit im Abgleich der Produkte der einzelnen Stablängen mit seinen jeweiligen Profilflächen um sie in Summe kleinstmöglich werden zu lassen. Die Änderung der einzelnen Stablängen und der einzelnen Flächen stehen aber nicht in einem linearen Bezug zur Lageänderung von Knoten 3, dadurch gehen sie auch nicht in einem linearen Zusammenhang in die zu minimierende Zielfunktion ein. Beide sind jeweils, und dadurch auch voneinander, abhängig von einem nichtlinearen Faktor der in unterschiedlicher Ausprägung die einzelnen Komponenten maßgeblich zu bestimmen scheint, und das sind die Winkelfunktionen (ausgedrückt durch die Stabkoeffizienten, vgl. Abbildung 3.5). Folgendes Schema veranschaulicht die Verkettung der für das Gewicht relevanten Größen mit der Änderung von (beliebigen) Knotenpositionen:



Die Stablängen gehen unmittelbar aus der Lage von Knoten 3 bezüglich Knoten 1 und 2 hervor (vgl. Kapitel 3.3.3), und sind damit von den (nichtlinearen) Winkelbeziehungen abhängig. Die Profilflächen gehen auch auf die Winkelbeziehungen zurück, aber diffiziler und verschachtelter. Zunächst einmal gibt es zwei verschiedene Funktionen zur Zuweisung der Profilflächen, und zwar einmal für Zug– und einmal für Druckstäbe (bei dieser Konfiguration wird dem Druckstab bzw. den Druckstäben die zulässige Knickfläche zugewiesen). Die Profilfläche eines Zugstabs hängt von der auf sie wirkenden Kraft ab, die wiederum von den Stabkoeffizienten, also den Winkelbeziehungen abhängt (vgl. Abbildung 3.5).

Bei dem Druckstab stehen sämtliche Größen, die die Profilfläche bestimmen, von Vorhinein in einem nichtlinearen Verhältnis zu dieser, da sie alle unter einem Quadratwurzel–Ausdruck stehen (wobei der quadratische Anteil der Stablängen sich unter der Wurzel neutralisiert, s.u.). Von denjenigen Größen dieses Terms, die durch die Optimierung veränderbar sind, gehen hier (wie beim Zugstab) ebenfalls die auf sie wirkende Kraft, und aber außerdem die Stablängen in die Gleichung der Profilfläche ein. Der nichtlineare Einfluss der Kraft wird hierbei durch die nichtlineare Wurzelfunktion überlagert. Der quadratische Einfluss der Stablänge gleicht die Wurzelfunktion zwar aus $(\sqrt{L^2}=L)$, steht aber (wie oben beschrieben) mit den Winkeln in einem nichtlinearen Verhältnis.

Zusammenfassend heißt das, dass der nichtlineare Anteil der Winkel mehrfach, teilweise überlagernd, und damit in unterschiedlicher Ausprägung die Profilflächen und die Längen der Stäbe bestimmt. Daher lassen sich die Lösungen mit bloßem Auge nicht nachvollziehen. Zur Kontrolle wurden allerdings die Werte der Lösungen (die Optima) aber dahingehend verifiziert, dass der Algorithmus mehrfach schrittweise ausgeführt und die Zwischenergebnisse nachgeprüft wurden. Ferner erfolgte eine weitere

Kontrolle, indem handschriftlich das Gewicht im globalen Optimum nachgerechnet wurde, wobei das Ergebnis identisch mit dem des Programmes war. Anschließend wurde der Knoten um grob 10 Einheiten (10 Meter) von -240.81 auf -230 verschoben, um experimentell zu überprüfen, ob in kleinerer Höhe das Gewicht tatsächlich nicht geringer ist. Würde dies der Fall sein, stünde das im Widerspruch zum gefundenen globalen Optimum. Der Algorithmus ermittelte dasselbe Ergebnis wie bei der handschriftlichen Rechnung, und zwar ein größeres Gewicht bei einer kleineren Höhe von -230. Schlussendlich wurde, um die Arbeitsweise des Programms und das globale Optimum nochmals zu überprüfen eine extrem große Startschrittweite von $d=10^{12}$ gewählt um entsprechend weit entfernt liegende Positionen zu prüfen. Die Schrittweite verkleinerte sich erwartungsgemäß über die in diesem Fall vorgegebenen 500 Generationen hinweg kontinuierlich um schließlich wieder zu der Position des zuvor ermittelten globalen Optimums zu gelangen.

Somit bleibt festzuhalten, dass alle Kontrollen und Tests die Korrektheit des Algorithmus und des globalen Optimums bestätigten.

4 Zusammenfassung und Ausblick

Das Fachwerk wurde erfolgreich aus den für ein Statik–Problem nötigen Informationen abgebildet. Daraus erschließt sich auch die Möglichkeit, beliebige statisch bestimmte Fachwerke höherer Komplexität zu realisieren. Anhand zweier Beispiele wurde dies getan, wobei die korrekt ausgegebenen Ergebnisse den Algorithmus bestätigten. Im Anschluss daran hat die Evolutionsstrategie ihre Qualitäten bewiesen, da die Optima, insbesondere das globale Optimum, so nicht vorhersehbar waren. Es konnte in den durchgeführten Testläufen der Gewichtsoptimierung durch Steuerung der Startschrittweite sowohl das lokal vorhandene als auch das globale Optimum zuverlässig gefunden werden. Die Gewichtsreduktion die damit um knapp 90% oder von 36.22 auf 3.64kg erzielt wurde, ist außerordentlich.

Bei der Analyse und Kontrolle des Optimierungsergebnisses fiel anschließend auf, dass die Minimierung der augenscheinlich simplen Gleichung des Gesamtgewichts weitaus diffiziler ist, als es den Anschein hatte, da die nichtlinearen Winkelbeziehungen für die durch die Gewichtsoptimierung veränderlichen Parameter einen größeren Einfluss zu haben scheinen als gedacht. Wenngleich mit einem Kennwert, der den nichtlinearen Einfluss über den Optimierungsprozess hinweg wiedergibt nicht gedient werden kann, so wurden stattdessen diverse empirische Kontrollen gemacht. Diese mehrfach, sowohl handschriftlich als auch maschinell durchgeführten Kontrollen haben das durch den Algorithmus ermittelte globale Optimum bestätigt bzw. einer anderen Lösung eines kleineren Gewichts für eine andere Knotenposition widersprochen.

Der Verfeinerung dieses Algorithmus sind selbstverständlich keine Grenzen gesetzt. Die Festigkeits–, Konstruktions– und Tragwerkslehre bieten einen unerschöpflichen Fundus an Möglichkeiten. Es sei insbesondere angemerkt, dass der Faktor der *Stabilität* Potential für eine differenziertere und tiefergehendere Auseinandersetzung birgt. Die hier implementierte zulässige Knickfläche kann als übeschlagsmäßige erste Auslegung verstanden werden, deren Zuverlässigkeit aber nicht in jedweder Konfigurationen gewährleistet ist.

5 Anhang

5.1 Vollständiger Quellcode

```
clear

% ------------------------------------------------------------------------
% Eingangsmatrizen, Konstruktion des Fachwerks
% ------------------------------------------------------------------------

% Äußere Kraft
F=100;

% Knotenkoordinaten, Koordinatensystem in Knoten 1
P1=[00,00];
P2=[10,00];
P3=[05,8.66];

% Knotenmatrix;
Knoten=[P1;P2;P3]';

% Knotenanfang und -ende der Stäbe
S1=[02,03];
S2=[01,03];
S3=[01,02];

% Stabmatrix
Verbindung=[S1;S2;S3]';

% Lagermatrix, Dimension entspricht Knotenmatrix
Lager=zeros(size(Knoten));
Lager(:,1)=[1 1]';
Lager(:,2)=[0 1]';

% Lastmatrix, Dimension enstpricht Knotenmatrix
Lasten=zeros(size(Knoten));
```

```
33 Lasten(2,3)=-F;
34
35 % Prüfen auf notwendige Bedingung für "statisch bestimmt" (Lagerreaktionen+Stäbe=2*Knoten)
36 if (sum(sum(Lager.^2))+size(Verbindung,2)) ~= 2*(size(Knoten,2))
37     error ('Das Tragwerk ist nicht statisch bestimmt, überprüfen Sie Ihre Eingaben')
38 end
39
40 % ------------------------------------------------------------------------
41 % Geometriebestimmung
42 % ------------------------------------------------------------------------
43
44 % Nullvektor und Nullmatrix; Dimension entsprechend der Stäbe, also von "Verbindung"
45 Stablaengen=zeros(1,size(Verbindung,2));
46 Stabkoeffizienten=zeros(size(Verbindung));
47
48 %Generieren der Stablängen und der Stabkoeffizienten
49 for j=1:size(Verbindung,2)
50     t=Verbindung(:,j);
51     Stablaengen(j)=norm(Knoten(:,t(2))-Knoten(:,t(1)));
52     Stabkoeffizienten(:,j)=(Knoten(:,t(2))-Knoten(:,t(1)))/Stablaengen(j);
53 end
54
55 % ------------------------------------------------------------------------
56 % Erstellen der Koeffizientenmatrix
57 % ------------------------------------------------------------------------
58
59 % Initialisieren der Koeffizientenmatrix (Zeilen: Doppelte Knotenzahl, Spalten: Stäbe,Lager)
60 Koeffmatrix=zeros(size(Verbindung,2)+sum(sum(Lager.^2)));
61
62 % Indexvektor
63 u=1:size(Knoten,1):size(Koeffmatrix,1)-1;
64
65 % Assemblierung der Stabkoeffizienten in die Koeffizientenmatrix (d.h.Winkel werden
   eingetragen)
66 for j=1:size(Verbindung,2)
67     s=Stabkoeffizienten(:,j);
68     oben=Verbindung(1,j);
```

```
69      unten=Verbindung(2,j);
70      Koeffmatrix(u(oben):u(oben)+1,j)=+s;
71      Koeffmatrix(u(unten):u(unten)+1,j)=-s;
72  end
73
74  % Assemblierung der Lagerkoeffizienten in die Koeffizientenmatrix
75  i=1;
76  for j=1:size(Koeffmatrix,2)
77      if Lager(j)~=0
78          Koeffmatrix(j,i+size(Verbindung,2))=Lager(j);
79          i=i+1;
80      end
81  end
82
83  % Prüfen auf hinreichende Bedingung für statische Bestimmtheit
84  if det(Koeffmatrix)==0
85      error ('Koeffizientenmatrix nicht eindeutig lösbar. Tragwerk ist beweglich. Überprüfen Sie Ihre
    Eingaben.')
86  end
87
88  % ------------------------------------------------------------------------
89  % Lösen des Gleichungssystems
90  % ------------------------------------------------------------------------
91
92  % Lastenmatrix wird zu einem Vektor überführt und negativ (Ax+b=0 -> Ax=-b)
93  Lasten=-Lasten(:);
94
95  % Lösen des Gleichungssystems
96  Forces=linsolve(Koeffmatrix, Lasten);
97
98  % ------------------------------------------------------------------------
99  % Generieren des Fachwerkgewichts
100 % ------------------------------------------------------------------------
101
102 %Materialwerte (hier: S235)
103 S=1.5; % Sicherheit
104 sigma_zzul=235e6/S;
```

```
105 sigma_dzul=235e6/S;
106 E=210e9;
107 rho=7900; % S. 116
108
109 % Anfangswerte der Stabprofilflächen
110 A_kzul=sqrt((abs(Forces(1:size(Verbindung,2)))'.*Stablaengen.^2*4*S/(pi*E)))';
111 A_dzul=abs(Forces(1:size(Verbindung,2)))/sigma_dzul;
112 A_zzul=abs(Forces(1:size(Verbindung,2)))/sigma_zzul;
113
114 % Initialisieren des Profilflächen-Vektors
115 A=zeros(size(Verbindung,2),1);
116
117 % Zuweisen der Profilflächen
118 for i=1:size(Verbindung,2)
119     if Forces(i)<0
120         if A_kzul(i) > A_dzul(i)
121             A(i)=A_kzul(i);
122         else
123             A(i)=A_dzul(i);
124         end
125     else
126         A(i)=A_zzul(i);
127     end
128 end
129
130 % Qualitätsfunktion (Gewicht)
131 qe=Stablaengen*A*rho;
132 Anfangsgewicht=qe;
133
134 % ------------------------------------------------------------------------
135 % Gewichtsoptimierung mittels Evolutionsstrategie
136 % ------------------------------------------------------------------------
137
138 % Neue Knotenmatrix, Variablenzahl, Startschrittweite
139 Kn=Knoten;
140 v=1;
```

```
141 d=8;
142 Startschrittweite=d;
143
144 % Speichern von Daten des aktuellen Elters (vgl. Zeile 202)
145 Fn=Forces; An=A; Stablaengen_n=Stablaengen;
146
147 % Übergabe der Anfangswerte an die grafische Ausgabe (g=0)
148 subplot (2,1,1)
149 semilogy(0,Anfangsgewicht,'b.')
150 hold on
151 semilogy(0,Startschrittweite,'y.')
152
153 % Generationenschleife
154 for g=1:200
155
156    Kn(2,3)=Knoten(2,3)+d*randn(size(Knoten(2,3)))/sqrt(v);
157
158    % Stabkoeffizienten
159    for j=1:size(Verbindung,2)
160       t=Verbindung(:,j);
161       Stablaengen(j)=norm(Kn(:,t(2))-Kn(:,t(1)));
162       Stabkoeffizienten(:,j)=(Kn(:,t(2))-Kn(:,t(1)))/Stablaengen(j);
163    end
164
165    % Assemblierung
166    for j=1:size(Verbindung,2)
167       s=Stabkoeffizienten(:,j);
168       oben=Verbindung(1,j);
169       unten=Verbindung(2,j);
170       Koeffmatrix(u(oben):u(oben)+1,j)=+s;
171       Koeffmatrix(u(unten):u(unten)+1,j)=-s;
172    end
173
174    % Lösen des Gleichungssystems
175    Forces=linsolve(Koeffmatrix, Lasten);
176
```

```
177  % Stabprofilflächen
178  A_kzul=sqrt((abs(Forces(1:size(Verbindung,2)))'.*Stablaengen.^2*4*S/(pi*E)))';
179  A_dzul=abs(Forces(1:size(Verbindung,2)))/sigma_dzul;
180  A_zzul=abs(Forces(1:size(Verbindung,2)))/sigma_zzul;
181
182  for i=1:size(Verbindung,2)
183      if Forces(i)<0
184          if A_kzul(i) > A_dzul(i)
185              A(i)=A_kzul(i);
186          else
187              A(i)=A_dzul(i);
188          end
189      else
190          A(i)=A_zzul(i);
191      end
192  end
193
194  % Qualitätsfunktion des 'Kindes'
195  qn=Stablaengen*A*rho;
196
197  % Vergleich der Qualitäten Elter/Kind
198  if qn < qe
199      qe=qn; Knoten=Kn; d=d*1.3;
200      % Speichern von neuen Werten des Elters (vgl. Zeile 146)
201      Fn=Forces; An=A; Stablaengen_n=Stablaengen;
202  else
203      d=d/(1.3^0.25);
204  end
205
206  % ---------------------------------------------------------------------
207  % Grafische Ausgabe
208  % ---------------------------------------------------------------------
209  subplot (2,1,1)
210
211  semilogy(g,qe,'b.')     % Plot Qualität (blau)
212      hold on
```

```
213     semilogy(g,d,'y.')      % Plot Schrittweite (gelb)
214
215     x=[Kn(1,:) Kn(1,1)];
216     y=[Kn(2,:) Kn(2,1)];
217
218     subplot (2,1,2)
219     plot(x,y,'LineWidth',2)
220
221     drawnow;
222
223 end % Generationenschleife
224
225 % ------------------------------------------------------------------------
226 % Ausgabe der Gewichtsersparnis
227 % ------------------------------------------------------------------------
228
229 disp 'Gewichtsersparnis in %:'
230 (1-qe/Anfangsgewicht)*100
231
232 % © Jonas Doßmann
```

5.2 Beigefügte CD

...mit folgendem Inhalt:

- Originale Dokumentation im *OpenDocument*–Format sowie eine konvertierte Version im PDF–Format
- Sämtliche verwendete Bilder im Original im *OpenDocument*–Format sowie konvertiert im JPEG–Format
- Quellcode als Matlab–Datei (Bachelorthesis_Dossmann.m)

6 Literaturverzeichnis

- Dankert, Jürgen u.A. Technische Mechanik – Computerunterstützt, Stuttgart 1994
- Fischer, Ulrich u.A.: Tabellenbuch Metall, Haan–Gruiten 2008
- Gross, Dietmar u.A.: Technische Mechanik Band 1 – Statik, Berlin Heidelberg 2009
- Kost, Bernd: Optimierung mit Evolutionsstrategien, Frankfurt/Main 2003
- Rechenberg, Ingo: Evolutionsstrategie ′94, Stuttgart 1994

Internetverweise (Stand: 01.10.2012)

- http://www.bionik.tu–berlin.de/institut/skript/E1–12Fo5.ppt