Lehrstuhl für Statik
der Technischen Universität München

Ermittlung optimierter Stabwerkmodelle auf Basis des Kraftflusses als Anwendung plattformunabhängiger Prozesskopplung

Alexander Hörmann

Vollständiger Abdruck der von der Fakultät für Bauingenieur– und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. rer.nat. Ernst Rank

Prüfer der Dissertation:

1. Univ.-Prof. Dr.–Ing. Kai-Uwe Bletzinger
2. Univ.-Prof. Dr.–Ing., Dr.–Ing. habil. Gerhard H. Müller

Die Dissertation wurde am 21.11.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 13.01.2006 angenommen.

Ermittlung optimierter Stabwerkmodelle auf Basis des Kraftflusses als Anwendung plattformunabhängiger Prozesskopplung

Zusammenfassung

Die Arbeit beschreibt eine Methode zur automatisierten Generierung optimierter Stabwerkmodelle für scheibenartige Systeme. Dies geschieht mit Hilfe der Kraftflussberechnung auf Basis einer linear–elastischen Spannungsermittlung.

Das Problem der Kompatibilität zwischen der kontinuierlichen Scheibe und dem beschreibenden diskontinuierlichen Stabwerkmodell konnte durch Minimierung der Differenzarbeit gelöst werden. Durch die damit angepassten Steifigkeitsverhältnisse kann auch der Kraftzustand der Scheibe durch das Stabwerk beschrieben werden.

Die Visualisierung des Kraftflusses dient damit nicht nur dem besseren Verständnis vom Tragverhalten komplexer, insbesondere statisch unbestimmter Systeme, sondern auch als Basis für eine wirtschaftliche Bemessung.

Die Umsetzung des Konzepts erfordert die Kombination unterschiedlicher Anwendungen. Die Verwendung von Softwarekomponenten ist eine effektive Methode, hohe Qualität bei geringer Entwicklungszeit zu gewährleisten. Das steigert nicht nur die Produktivität, sondern verringert insbesondere die zeitliche Verzögerung, bis Forschungsergebnisse Anwendung in der Praxis finden.

Im zweiten Teil der Arbeit wird ein Konzept für ein Softwaresystem erläutert, das eine effektive Kopplung von Prozessen in einem verteilten System ermöglicht. Unter Verwendung einer allgemeinen Kommunikationsschnittstelle können bestehende Softwaremodule gekapselt und auf Basis eines Client–Server–Modells Daten untereinander ausgetauscht werden.

Die System- und Rechnerunabhängige Kopplung einer adaptiven Visualisierungsumgebung mit geeigneten Berechnungsmodulen ermöglicht die Umsetzung der beschriebenen Kraftflussberechnung.

Durch den zusätzlichen Einsatz eines Workflow–Management–Systems werden geeignete Module und Objekte, die auf beliebigen Rechnern verteilt sein können, zur Laufzeit miteinander verknüpft und gesteuert. Damit steht der schnellen Entwicklung von Spezialsoftware unter Verwendung ausgereifter Komponenten zur Lösung individueller Probleme nichts mehr im Wege.

Determination of optimated strut and tie models based on force flow as an application of platform independent remote process interfacing

Abstract

This thesis describes a new method for an automated determination of optimised strut and tie models. This is done by the determination of the force flow of plates based on a linear elastic calculation of stresses.

The problem of kompability of deformation of a continuous plate an a strut and tie model could be answered by mimimizing the strain energy of the differences of displacements. With the adapted truss stiffnesses the distribution of forces of the plate can be described.

The visualisation of force flow not only provides a better understanding of complex statically indetermined systems, but also a basis for economic reinforcement.

The application of this concept needs a combination of different tools. Reuse of software components is an effective way to ensure high quality and short developmental periods. This leads to higher efficiency and reduces the gap between research and the actual application in practice.

The second part of this thesis describes a general and effective concept for a software system, which allows processes within a distributed system to be linked. Using a general communication interface, existing software modules can be encapsulated, and data can be exchanged based on a client server model.

The open system interconnection of an adaptive visualisation tool and appropriate calculation modules enables the implementation of the above described method.

Additionally, a workflow management system can link and control appropriate modules and objects distributed on a computer network in runtime. This enables a rapid development of special software, consisting of sophisticated components, to solve customized problems.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Als Einstieg in die Arbeit soll ein grundsätzliches Problem für Berechnungsingenieure aufgezeigt werden, welches als Motivation für einen nachfolgend beschriebenen Lösungsansatz dient. Ein kurzer Überblick gliedert die Arbeit anschließend und sorgt damit für besseres Verständnis.

1.1 Motivation

Ingenieure werden täglich mit neuen Herausforderungen in den unterschiedlichsten Disziplinen konfrontiert, zu deren Lösung sie zunehmend auf Computerberechnungen angewiesen sind. Besonders im Bereich der Simulation realitätsnaher Modelle sind sie mittlerweile unerlässlich. Die Herausforderung für Ingenieure besteht darin, das Tragverhalten von Systemen schnell und korrekt zu erfassen, um optimierte Lösungen hinsichtlich ihrer konstruktiven Durchbildung zu finden. Das gilt insbesondere für die Bewehrungsführung von Stahlbetonbauteilen.

Am einfachen Beispiel des ebenen Spannungszustandes scheibenartiger Tragwerke soll ein Konzept zur Visualisierung des Kraftflusses entstehen, welches dem besseren Verständnis vom Tragverhalten dient und Ausgangspunkt für eine wirtschaftliche Bewehrung darstellt. Mit Hilfe von Kraftflusspfaden sollen Stabwerkmodelle beschrieben werden, die ihren Ursprung bei Mörsch [Mör12] und Schlaich [Sch93] haben.

Zwar existieren gute Ansätze zur Generierung von Stabwerkmodellen für unterschiedliche Problemfälle, jedoch mit einem entscheidenden Nachteil. Die Kompatibilität in den Verschiebungen zwischen dem tatsächlichen System und dem beschreibenden Stabwerk ist nicht gegeben. Für die Ermittlung dessen Stabkräfte sind im statisch unbestimmten Fall die Verformungen oder die tatsächlichen Steifigkeitsverhältnisse zwingend notwendig. Bis dato wird dafür in der Regel ein Ansatz über „beliebig“ gewählte Stabbreiten gemacht (z.B. [Rüc92]). Diese willkürliche Aussage soll durch eine eindeutig definierbare mechanische Größe ersetzt werden.

In einem teil- oder vollautomatischen Programmsystem sollen die theoretischen Erkenntnisse ihre Anwendbarkeit auf praktische Problemstellungen beweisen. Bei der Ausführung wird die Notwendigkeit einer möglichst einfachen und flexiblen Zusammenführung unterschiedlicher Softwarekomponenten deutlich.

1.2 Zielsetzung

Der enorme Fortschritt in der Rechnerentwicklung, sowie deren globale Verbreitung und Akzeptanz schafft völlig neue Möglichkeiten für technische Anwendungen. Die Übertragung neuartiger Methoden und Denkansätze in den Bereich der Ingenieurwissenschaften ist nur eine logische Konsequenz.

Aber gerade bei komplexen Problemen des Ingenieurwesens stoßen aktuelle Programmsysteme schnell an ihre Grenzen. Dazu gehören Mehrfeldprobleme, wie zum Beispiel die Fluid–Struktur–Interaktion, der Bereich der Optimierung, oder Mehrskalenprobleme.

In der Problemlösung ist die Forschung in vielen Bereichen der kommerziellen Anwendung schon weit voraus, der zeitliche Versatz allerdings bis zur tatsächlichen Anwendbarkeit in der Praxis ist oft noch zu groß. Die Nachfrage der Anwender zeigt deutlich die Notwendigkeit eines unmittelbareren Wissenstransfers.

In Ingenieurbüros werden für neuartige, komplexe Aufgaben immer wieder pragmatische aber umständliche Problemlösungen vorgeschlagen. Alle notwendigen Schritte einer Computersimulation werden durch spezielle Softwaremodule erledigt. Deren Koordination und die Gewährleistung der Kompatibilität obliegen jedoch dem Ingenieur, was zeitintensiv und fehleranfällig ist.

Angehende Kooperationen von Software–Entwicklern in den Bereichen der Ingenieurwissenschaften unterstreicht die steigende Nachfrage nach adaptiven Lösungen für die Praxis.

An dieser Stelle entsteht die Frage nach einem allgemeingültigen und effektiven Konzept zur Entwicklung und Anwendung eines neuartigen Softwaresystems im Ingenieurwesen. Für Anwender sollte es möglichst vielschichtig einsetzbar sein. Unnötiger Overhead sollte im Sinne hoher Transparenz und geringerer Rechenzeiten vermieden werden. Entwicklern sollte ein Austausch oder die Erweiterung beliebiger Komponenten durch einen modularen Aufbau erleichtert werden.

Durch den Einsatz vorhandener Komponenten können problemorientierte Softwarelösungen rasch umgesetzt werden. Die Portierung der Module ist jedoch oft schwer möglich, da sie für spezielle Plattformen oder Rechnerarchitekturen konzipiert sind. Ein Lösungsansatz dafür ist die Kapselung derartiger Prozesse auf unterschiedlichen Rechnern, die dann in einem verteilten System über ein Computernetz

miteinander gekoppelt werden. Die plattformunabhängige Realisierung einer individuell anpassbaren Kommunikationsschnittstelle gewährleistet dabei Flexibilität und Erweiterbarkeit.

Ziel der Arbeit ist die Generierung eines Softwaresystems zur Umsetzung der Kraftflussberechnung. Hier soll das eben angesprochene Softwarekonzept zum Einsatz kommen.

Im Gegensatz zur Kraftflussberechnung folgt der Ablauf komplexer Computersimulationen nicht mehr unbedingt genau vordefinierter Regeln. Sind an einem Rechenlauf mehrere, je nach Anforderung unterschiedliche, im System verteilte Prozesse beteiligt, ist es unter Umständen nicht mehr möglich oder sinnvoll, sie unmittelbar in starren verteilten Anwendungen zu codieren. Deren Kopplung und Steuerung sollte indes in abstrakter Form beschrieben, von einem Workflow–Management–System interpretiert und schließlich ausgeführt werden können. Durch eine grafische Oberfläche werden dabei einzelne modulare Software–Komponenten miteinander verknüpft und ihr zeitlicher Ablauf geregelt. Auch dieser Umstand soll in der Realisierung des Softwaresystems in Betracht gezogen werden.

1.3 Überblick

Aufgrund der Aufgabenstellung gliedert sich die Arbeit grundsätzlich in zwei Bereiche, die schließlich in einer Anwendung zusammengeführt werden.

Zum einen wird die generelle Schwierigkeit aufgegriffen, Ergebnisse von Finite–Elemente–Berechnungen kontinuierlicher Systeme zu interpretieren. Zum besseren Verständnis des Tragverhaltens wird am Beispiel des ebenen Spannungszustandes von Scheiben das kontinuierliche Problem auf ein diskretes Stabwerkmodell zurückgeführt.

Nach der Darstellung bekannter Konzepte zur Ermittlung von Stabwerkmodellen wird ein eigenes Verfahren vorgestellt. Gefolgt von der Beschreibung der Geometrie, werden die Steifigkeitsverhältnisse der einzelnen Stäbe unter Verwendung der Kompatibilität der Deformation von ursprünglicher Scheibe und dem beschreibenden Stabwerkmodell ermittelt. Eine anschließende Schnittgrößenbrechnung ermöglicht neben qualitativen auch quantitative Aussagen über das Tragverhalten. Damit ist der Grundstein für eine optimierte Bewehrung von Stahlbetontragwerken gelegt.

Der zweite Teil der Arbeit befasst sich mit einem Softwarekonzept, welches anpassungsfähig auf neuartige Problemstellungen im Ingenieurwesen reagieren soll. Unter Verwendung ausgereifter Komponenten werden verteilte Softwaresysteme modular und erweiterbar zusammengestellt. Dafür werden grundlegende Möglich-

keiten der Kopplung entfernter Prozesse beleuchtet und anschließend ein Client–Server–basiertes Kommunikationsmodell vorgestellt.

Die Umsetzung des Softwarekonzepts am Beispiel der Visualisierung des Kraftflusses wird in einem weiteren Abschnitt behandelt. Hier werden die benötigten, verteilten Prozesse mit Hilfe der Kapselungstechnik gekoppelt und durch einen zentralen Prozess gesteuert.

Eine Erweiterung des Konzepts der fest programmierten Kopplung von Prozessen in einem verteilten System hin zu einem Workflow–Management–System wird am selben Beispiel verifiziert. Somit können je nach Anforderung passende Applikationen während ihrer Laufzeit miteinander verknüpft werden.

Die Eignung des Softwarekonzepts für Simulationsanwendungen wird im Kapitel der numerischen Beispiele verifiziert. Zunächst aber werden wichtige Grundlagen zusammengefasst, die im weiteren Verlauf der Arbeit benötigt werden.

Kapitel 2

Grundlagen

In diesem Kapitel werden in komprimierter Form die wesentlichen Grundgleichungen und Begriffe der Kontinuumsmechanik und der Finite–Elemente–Methode zusammengefasst, soweit sie für das Verständnis der folgenden Abschnitte notwendig sind. Auf eine allgemeine Darstellung wird hier verzichtet, an Stelle dessen sei auf die einschlägigen Werke der Literatur verwiesen, wie Marsden und Hughes [Mar83], Malvern [Mal69], Holzapfel [Hol00], Zienkiewicz und Taylor [Zie00] sowie Hughes [Hug00].

2.1 Kontinuumsmechanik

In Bezug auf die in dieser Arbeit betrachteten ebenen Systeme wird ein unveränderliches kartesisches Koordinatensystem verwendet. Bei der Beschreibung von Deformationsvorgängen eines Materialpunktes wird die bei Strukturproblemen übliche Lagrangesche Betrachtungsweise verwendet.

2.1.1 Kinematik

Die Beziehungen zwischen Verformungen und Verzerrungen eines Körpers werden durch die kinematischen Gleichungen hergestellt. Ein beliebiger Punkt **P** auf einem Körper wird in seiner Referenzkonfiguration durch seinen Ortsvektor **X** bestimmt. Eine Bewegung des Körpers in den Momentanzustand **x** wird durch den zugehörigen Verschiebungsvektor **u** beschrieben.

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \tag{2.1}$$

Die Differentiation des Ortsvektors in der Momentankonfiguration führt auf den Deformationsgradienten **F**.

$$d\mathbf{x} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}\, d\mathbf{X} = \mathbf{F}\, d\mathbf{X} \qquad \text{mit } \mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{2.2}$$

Die Determinante J des Deformationsgradienten (Jacobi–Determinante) setzt die deformierten und undeformierten differentiellen Volumina in Relation zueinander.

$$J = \det(\mathbf{F}) = \frac{dV}{dV_0} \tag{2.3}$$

Als objektives Verzerrungsmaß dient der Green–Lagrange Verzerrungstensor $\mathbf{E}$.

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^{\mathbf{T}}\mathbf{F} - \mathbf{I}) \tag{2.4}$$

Gleichung 2.4 stellt gleichzeitig die kinematische Feldgleichung dar. Zusätzlich sind auf dem Rand Γ_u die Verschiebungsrandbedingungen $\mathbf{u} = \hat{\mathbf{u}}$ einzuhalten.

2.1.2 Spannungen und Gleichgewicht

Der auf die Momentankonfiguration bezogene Spannungstensor 1. Ordnung $\mathbf{t}$ lässt sich mit Hilfe des Cauchy–Theorems durch den Cauchy–Spannungstensor σ und den Normaleneinheitsvektor $\mathbf{n}$ beschreiben.

$$\mathbf{t} = \sigma\mathbf{n} \tag{2.5}$$

Dabei stellt σ die tatsächlichen physikalischen Spannungen am Schnitt dar, der durch den Normaleneinheitsvektor $\mathbf{n}$ definiert ist. Wird der Spannungstensor $\mathbf{t}$ auf die Ausgangskonfiguration bezogen, ergibt sich unter der Bedingung, dass sich die auf ein bestimmtes Flächenelement resultierenden Kraft nicht ändern darf, der unsymmetrische Piola–Kirchhoffsche Spannungstensor 1.Art.

$$\mathbf{P} = \det(\mathbf{F})\sigma\mathbf{F}^{-T} \tag{2.6}$$

Den symmetrischen Piola–Kirchhoffschen Spannungstensor 2.Art erhält man durch Multiplikation der Gleichung 2.6 mit dem inversen Deformationsgradienten.

$$\mathbf{S} = \mathbf{F}^{-1}\mathbf{P} = \det(\mathbf{F})\mathbf{F}^{-1}\sigma\mathbf{F}^{-T} \tag{2.7}$$

Der Piola–Kirchhoffsche Spannungstensor 2.Art $\mathbf{S}$ ist das energetisch konjugierte Gegenstück zum Green–Lagrangeschen Verzerrungstensor $\mathbf{E}$ und lässt sich demnach zur Formulierung der Grundgleichung in Lagrangescher Betrachtungsweise einsetzen.

Mit Hilfe des Gaußschen Integralsatzes und des Cauchy–Theorems können nun aus dem Impulssatz – unter Vernachlässigung der Beschleunigungen – die lokalen Gleichgewichtsaussagen hergeleitet werden.

$$\operatorname{div} \boldsymbol{\sigma} + \rho \mathbf{b} = 0 \tag{2.8}$$

Diese können auch bezüglich der Größen des Ausgangszustandes geschrieben werden.

$$\operatorname{div} (\mathbf{FS}) + \rho \mathbf{b} = 0 \tag{2.9}$$

Der Term $\rho\mathbf{b}$ beinhaltet die über das Volumen auftretende Kraft.

2.1.3 Materialgesetze

Für die Beziehungen zwischen den Spannungen und den Verzerrungen gelten die konstitutiven Gleichungen, auch Stoff- oder Materialgesetze genannt.

Für die Belange dieser Arbeit haben unterschiedliche Werkstoffgesetze keinen Einfluss. Im Folgenden werden deswegen lediglich die grundlegenden Beziehungen linear–elastischen Materialverhaltens vom Typ St. Venant–Kirchhoff dargestellt.

Wie bereits angedeutet, können die Werkstoffgleichungen jedoch nicht bezüglich der mechanischen Größen beliebiger Referenzkonfigurationen aufgestellt werden, sondern müssen im energetischen Sinne zueinander konjugiert sein. Anhand der Piola–Kirchhoffschen Spannungstensoren 2. Art $\mathbf{P}$ und den Green–Lagrange–Verzerrungen $\mathbf{E}$ sollen nun beispielhaft die Beziehungen für große Verschiebungen, aber kleine Verzerrungen dargestellt werden.

$$\mathbf{S} = \mathbf{C} : \mathbf{E} \tag{2.10}$$

Darin ist $\mathbf{C}$ der vierstufige Werkstofftensor. Ist das Material zudem isotrop, reichen zwei Parameter zur vollständigen Beschreibung der Materialeigenschaften. Hierzu werden die Lamé–Konstanten λ und μ verwendet, die mit dem Elastizitätsmodul E und der Querdehnzahl ν wie folgt in Beziehung gebracht werden können.

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \qquad \mu = \frac{E}{2(1+\nu)} \tag{2.11}$$

Für diesen Fall lassen sich die Komponenten des Materialtensors wie folgt anschreiben.

$$C_{ijkl} = \lambda\, \delta_{ij}\delta_{kl} + \mu\, (\delta_{ik}\delta_{jl} + \delta_{il}\delta_{kj}) \tag{2.12}$$

Das vorgestellte Materialgesetz erfüllt vollständig die Forderung nach Objektivität. Es ist invariant gegenüber Starrkörperbewegungen sowie Drehungen des verwendeten Koordinatensystems.

Kommen nichtlineare Werkstoffgesetze zur Anwendung, können die grundlegenden Beziehungen im Allgemeinen nur noch in inkrementeller Form aufgestellt werden. Für einen weiter reichenden Überblick sei beispielhaft Belytschko, Liu und Moran [Bel01] genannt.

2.1.4 Energieprinzipien

Neben den bisher aufgestellten Grundgleichungen müssen auch die kinematischen und statischen Randbedingungen, auch als Dirichlet und Neumann Randbedingungen bekannt, zur Lösung herangezogen werden. Damit ergibt sich die starke Form des Randwertproblems zu:

Gleichgewicht:	$\text{div}\,(\mathbf{F}\,\mathbf{S}) + \rho\,\mathbf{b} = \mathbf{0}$	in	Ω
statische Randbedingungen:	$\mathbf{F}\,\mathbf{S}\,\mathbf{N} = \hat{\mathbf{t}}$	auf	Γ_S
Kinematik:	$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I})$	in	Ω
kinematische Randbedingungen:	$\mathbf{u} = \hat{\mathbf{u}}$	auf	Γ_u
Stoffgesetz:	$\mathbf{S} = \mathbf{C} : \mathbf{E}$	in	Ω

Eine analytische Lösung für diese Differentialbeziehungen ist nur in Sonderfällen möglich. Häufig bilden integrale Formulierungen, die sogenannte schwache Form, die Basis für Näherungslösungen.

Das Prinzip von HU–WASHIZU oder das Prinzip von HELLINGER–REISSNER verwenden sowohl Verschiebungs- als auch Spannungs- und Verzerrungsfreiheitsgrade. Das PRINZIP DER VIRTUELLEN VERSCHIEBUNGEN (PvV) ist eines der Energieprinzipien, in dem die Verschiebungen alleine die primären Variablen darstellen. Es dient im weiteren Verlauf zur Herleitung der Methode der Finiten Elemente und damit zur Lösung des oben beschriebenen Randwertproblems.

Das Prinzip der virtuellen Verschiebungen besagt, dass sich ein mechanisches System dann im Gleichgewicht befindet, wenn die durch eine Variation verursachte

Arbeit verschwindet. Dabei wird eine infinitesimal kleine und geometrisch verträgliche virtuelle Verschiebung $\delta\mathbf{u}$ am System aufgebracht. Es beschreibt anschaulich die Bilanz aus innerer Arbeit δW^{int} und äußerer Arbeit δW^{ext}.

$$\delta W(\mathbf{u}) = \delta W^{int}(\mathbf{u}) - \delta W^{ext}(\mathbf{u}) = \int_{\Omega} \mathbf{S} : \delta\mathbf{E}\, d\Omega - \int_{\Omega} \rho\, \mathbf{b}\, \delta\mathbf{u}\, d\Omega - \int_{\Gamma_S} \hat{\mathbf{t}}\, \delta\mathbf{u}\, d\Gamma_S \;=\; 0 \tag{2.13}$$

Zusätzlich treten drei Nebenbedingungen auf. Die Kinematik und das Werkstoffgesetz gelten im Gebiet Ω, während Verschiebungsbedingungen auf dem Rand Γ_u eingehalten werden müssen. Das Prinzip der virtuellen Verschiebungen kann auch aus dem PRINZIP VOM MINIMUM DER POTENTIELLEN ENERGIE hergeleitet werden.

$$\Pi(\mathbf{u}) = \int_{\Omega} \Pi^{int}(\mathbf{E})\, d\Omega - \int_{\Omega} \rho\, \mathbf{b}\, \mathbf{u}\, d\Omega - \int_{\Gamma_S} \hat{\mathbf{t}}\, \mathbf{u}\, d\Gamma_S \;\rightarrow\; min.$$

2.2 Die Finite–Elemente–Methode

Das Prinzip der virtuellen Verschiebungen ist ein möglicher Ansatz für die Herleitung der Finite–Elemente–Methode. Mit Hilfe der direkten Steifigkeitsmethode wird damit eine computerorientierte, numerische Lösung für das in Kapitel 2.1.4 benannte Problem beschrieben.

Besonderes Augenmerk gilt dann dem im Rahmen der vorliegenden Arbeit betrachteten ebenen Spannungszustand. Bei der Elementformulierung wird auf das verschiebungsformulierte, isoparametrische vierknotige Element zurückgegriffen.

2.2.1 Allgemeine Einführung

Für erste Betrachtungen werden geometrisch, sowie materiell lineare Zusammenhänge angenommen. Relationen zwischen Spannungen und Dehnungen bei kleinen Verzerrungen sind ebenfalls linear angesetzt. Als Spannungs- und Dehnungsmaß werden üblicherweise die Piola–Kirchhoff–Spannungen 2. Art und die Green–Lagrangeschen Verzerrungen verwendet. Die zueinander konjugierten Größen gestatten die Formulierung des PvV, das im Zuge der Finite–Elemente–Methode diskretisiert wird.

Dabei wird die zu untersuchende Struktur in eine Anzahl endlicher Elemente unterteilt. Innerhalb dieser wird das Feld der unbekannten Verformungen **u** aus Gleichung 2.13 durch geeignete Ansatzfunktionen in Abhängigkeit von diskreten Knotenwerten approximiert. Somit entsteht aus dem kontinuierlichen Problem ein System von n Gleichungen mit n Unbekannten, dessen Lösung die Bedingungen aus Kapitel 2.1.4 in integraler Form erfüllen. Die erwähnten Nebenbedingungen im Gebiet und auf dem Rand sind einzuhalten.

Aus den diskreten Knotenverformungen lässt sich über die Kinematik die Deformation des Elementes bestimmen. Daraus werden mit Hilfe der konstitutiven Beziehung die Spannungen zurück gerechnet.

Neben der hier ausschließlich betrachteten Verschiebungsmethode seien an dieser Stelle noch die Gleichgewichtsmethode und weitere gemischte und hybridgemischte Methoden zur Problemlösung genannt.

Bei der Beschreibung der grundlegenden Beziehungen wird im Folgenden die Voigt–Notation verwendet, wie sie im Allgemeinen bei der Beschreibung der Finite–Elemente–Methode (FEM) gebräuchlich ist.

2.2.2 Der ebene Spannungszustand

Ist ein Tragwerk eben und nur in seiner Ebene belastet, so treten in dieser Ebene Spannungen auf. Zudem können zwei Zustände unterschieden werden. Beim hier nicht betrachteten ebenen Verzerrungszustand wird die Ausdehnung senkrecht zur betrachteten Ebene behindert, wodurch zusätzliche Spannungen in dieser Richtung auftreten. Im Fall des ebenen Spannungszustandes sind die Verformungen senkrecht zur Ebene nicht behindert, die betreffenden Spannungskomponenten σ_z, τ_{xz}, τ_{zx}, τ_{yz} und τ_{zy} verschwinden.

Der komplexe räumliche Spannungszustand kann dadurch gemäß Abbildung 2.1 vereinfacht werden.

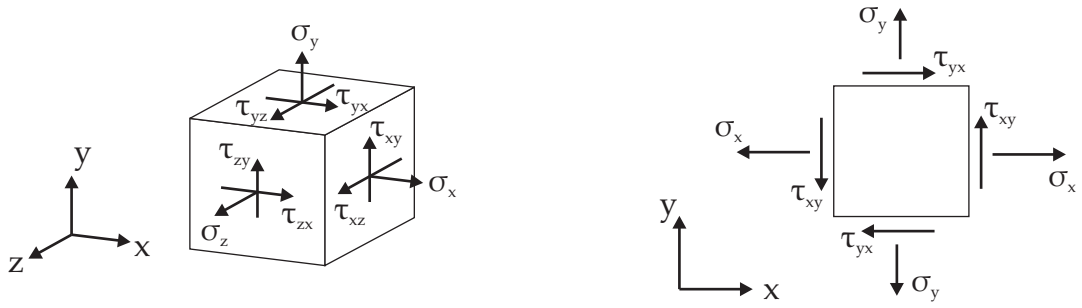


Abbildung 2.1: Räumlicher und ebener Spannungszustand

Das zugrunde liegende Tragsystem wird als Scheibe bezeichnet. Unter Annahme konstanter Dicke t lässt sich Gleichung 2.13 für den ebenen Spannungszustand vereinfachen.

$$t \int_A \delta\boldsymbol{\varepsilon}^T \boldsymbol{\sigma}\, dxdy - \int_A \mathbf{p}^T\, \delta\mathbf{u}\, dxdy - \int_{\partial A} \mathbf{P}^T\, \delta\mathbf{u}\, d\partial A \;=\; 0 \tag{2.14}$$

Darin sind $\mathbf{p}$ die im Gebiet A auftretenden Lasten und $\mathbf{P}$ die Lasten auf dem Rand ∂A, wie Abbildung 2.2 verdeutlicht.

$$\mathbf{p} = \left\{ \begin{array}{c} p_x \\ p_y \end{array} \right\}, \quad \mathbf{P} = \left\{ \begin{array}{c} P_x \\ P_y \end{array} \right\} \tag{2.15}$$

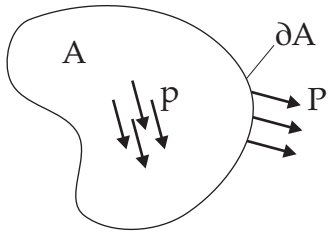


Abbildung 2.2: Belastung beim ebenen Spannungszustand

Der Verformungszustand wird im betrachteten Fall ausschließlich durch die zwei Verschiebungskomponenten u_x und u_y beschrieben.

$$\mathbf{u} = \left\{ \begin{array}{c} u_x \\ u_y \end{array} \right\} \tag{2.16}$$

Aus den Verschiebungen lassen sich über die kinematischen Zusammenhänge die Verzerrungen ε_x, ε_y und γ_{xy} ermitteln.

$$\boldsymbol{\varepsilon} = \left\{ \begin{array}{c} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{array} \right\} = \left[\begin{array}{cc} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{array} \right] \left\{ \begin{array}{c} u_x \\ u_y \end{array} \right\} = \mathbf{D}\,\mathbf{u} \tag{2.17}$$

Das Materialgesetz gibt die Beziehung zwischen Spannungen und Verzerrungen an. Für das isotrope elastische Kontinuum gilt das Hookesche Gesetz.

$$\boldsymbol{\sigma} = \left\{ \begin{array}{c} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{array} \right\} = \frac{E}{1-\nu^2} \left[\begin{array}{ccc} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{array} \right] \left\{ \begin{array}{c} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{array} \right\} = \mathbf{C}\,\boldsymbol{\varepsilon} \tag{2.18}$$

Die Schnittgrößen n_x, n_y und n_{xy} können unter Einbeziehung der Scheibendicke aus den Spannungen errechnet werden.

$$\left\{ \begin{array}{c} n_x \\ n_y \\ n_{xy} \end{array} \right\} = \sigma \, \mathbf{t} \tag{2.19}$$

2.2.3 Diskretisierung mit Hilfe des isoparametrischen Konzepts

Eine analytische Lösung für die Beziehungen des ebenen Spannungszustandes ist ebenfalls nur in Sonderfällen möglich. Als Näherungslösung wird hier die Methode der Finiten Elemente eingesetzt. Dabei wird die Struktur zunächst in Elemente unterteilt. Die darauf bezogenen Größen sind mit dem Kopfzeiger e versehen.

Beim isoparametrischen Konzept werden sowohl die Geometrie als auch die Verformungen eines Elementes durch die selben Formfunktionen beschrieben. Sie interpolieren die unbekannten Knotengrößen im Element. Am Beispiel des bilinearen vierknotigen Elements aus Abbildung 2.3 wird im Folgenden die Diskretisierung des Funktionals 2.14 vorgenommen, welche auf ein lineares Gleichungssystem führt.

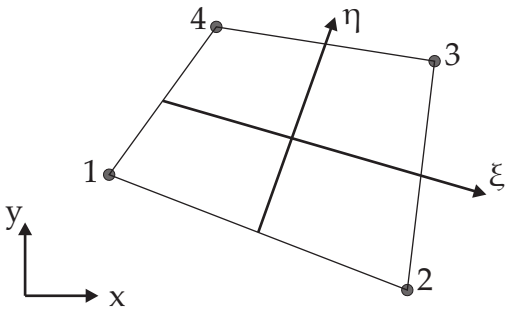


Abbildung 2.3: Vierknotiges Element mit seinen natürlichen Koordinatenachsen

Die Lage eines Punktes innerhalb eines Elementes in natürlichen Koordinaten wird durch die Koordinaten dessen Eckknoten nach Gleichungen 2.20 und 2.21 und die bilinearen Formfunktionen N_i nach Gleichungen 2.22 bis 2.25 bestimmt.

$$x(\xi, \eta) = \sum_{i=1}^{4} N_i^e \cdot x^i \tag{2.20}$$

$$y(\xi, \eta) = \sum_{i=1}^{4} N_i^e \cdot y^i \tag{2.21}$$

$$N_1^e = \frac{1}{4}(1-\xi)(1-\eta) \tag{2.22}$$

$$N_2^e = \frac{1}{4}(1+\xi)(1-\eta) \tag{2.23}$$

$$N_3^e = \frac{1}{4}(1+\xi)(1+\eta) \tag{2.24}$$

$$N_4^e = \frac{1}{4}(1-\xi)(1+\eta) \tag{2.25}$$

Die Verschiebung eines Punktes im Element wird ebenfalls in natürlichen Koordinaten des Elementes und dessen Knotenverschiebungen ausgedrückt, hier in Komponenten- und Matrizenschreibweise.

$$u_x(\xi,\eta) = \sum_{i=1}^{4} N_i^e \cdot u_x^i \tag{2.26}$$

$$u_y(\xi,\eta) = \sum_{i=1}^{4} N_i^e \cdot u_y^i \tag{2.27}$$

$$\mathbf{u}(\xi,\eta) = \begin{bmatrix} N_1^e & 0 & N_2^e & 0 & N_3^e & 0 & N_4^e & 0 \\ 0 & N_1^e & 0 & N_2^e & 0 & N_3^e & 0 & N_4^e \end{bmatrix} \mathbf{u^e} = \mathbf{N^e}\,\mathbf{u^e} \tag{2.28}$$

Die Verzerrungen $\boldsymbol{\varepsilon}$ werden durch die Verformungsfreiheitsgrade ausgedrückt.

$$\boldsymbol{\varepsilon}(\xi,\eta) = \begin{bmatrix} \frac{\partial N_1^e}{\partial x} & 0 & \frac{\partial N_2^e}{\partial x} & 0 & \frac{\partial N_3^e}{\partial x} & 0 & \frac{\partial N_4^e}{\partial x} & 0 \\ 0 & \frac{\partial N_1^e}{\partial y} & 0 & \frac{\partial N_2^e}{\partial y} & 0 & \frac{\partial N_3^e}{\partial y} & 0 & \frac{\partial N_4^e}{\partial y} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_2^e}{\partial x} & \frac{\partial N_3^e}{\partial y} & \frac{\partial N_3^e}{\partial x} & \frac{\partial N_4^e}{\partial y} & \frac{\partial N_4^e}{\partial x} \end{bmatrix} \mathbf{u^e} = \mathbf{B}\,\mathbf{u^e} \tag{2.29}$$

Nach der direkten Steifigkeitsmethode kann das PvV (Gleichung 2.14) auch am Element aufgestellt werden. Dessen Beiträge zu den inneren und äußeren virtuellen Arbeiten sind in 2.30 bis 2.32 aufgeführt.

$$t \int_A (\delta\mathbf{u}^e)^T\, \mathbf{B}^T\, \mathbf{C}\, \mathbf{B}\, \mathbf{u^e}\, dxdy \tag{2.30}$$

$$- \int_A \mathbf{p}\, \mathbf{N^e}\, \delta\mathbf{u^e}\, dxdy \tag{2.31}$$

$$- \int_{\partial A} \mathbf{P}\, \mathbf{N^e}\, \delta\mathbf{u^e}\, d\partial A \;=\; 0 \tag{2.32}$$

Der Vektor $\mathbf{u}^e$ beinhaltet die unbekannten Knotenverschiebungen des Elementes. Aus dem Ausdruck 2.30 entsteht später die Elementsteifigkeitsmatrix $\mathbf{k}^e$.

$$\mathbf{k}^e = t \int_A \mathbf{B}^T \, \mathbf{C} \, \mathbf{B} \, dxdy \tag{2.33}$$

Das Integral wird programmtechnisch meist mit der Gauß–Integration bestimmt. Dafür muss der **B**–Operator an den Gaußpunkten berechnet werden.

Im so genannten Elementlastvektor **f** werden die Rand- und die Gebietslasten zusammengefasst.

$$\mathbf{f} = \int_A \mathbf{p} \, \mathbf{N^e} \, dxdy + \int_{\partial A} \mathbf{P} \, \mathbf{N^e} \, d\partial A \; = \; 0 \tag{2.34}$$

Schließlich werden alle Elementsteifigkeiten $\mathbf{k}^e$ zu einer Systemsteifigkeitsmatrix **K** assembliert und mit Hilfe des Gleichgewichts zu einem Gesamtgleichungssystem zusammengestellt, die Verschiebungsnebenbedingungen eingearbeitet und anschließend gelöst (vgl. Gleichung 2.35).

$$\mathbf{K} \, \mathbf{u} = \mathbf{f} \tag{2.35}$$

Die Rückrechnung der Spannungen erfolgt dann wieder auf Elementebene. Dazu werden die Verzerrungen an den Gaußpunkten aus den bekannten Knotenverschiebungen mit dem B–Operator berechnet. Anschließend lassen sich die Spannungen mit Hilfe des Stoffgesetzes ermitteln und im Element beliebig interpolieren. Alternativ dazu können die Spannungen auch direkt an der gewünschten Stelle über den B–Operator ermittelt werden.

$$\sigma^e = \mathbf{C} \, \boldsymbol{\varepsilon}^e = \mathbf{B} \, \mathbf{u}^e \tag{2.36}$$

Die Spannungswerte dienen als Grundlage zur Ermittlung der Kraftflusspfade.

2.3 FEM und objektorientierte Programmierung

Heutzutage werden immer höhere Ansprüche an Simulationssoftware gestellt, die seitens der Entwickler so schnell wie möglich umgesetzt werden müssen. Im Sinne einer flexiblen Erweiter- und Anpassbarkeit an neue Problemstellungen ist darüber nachzudenken, inwiefern das Paradigma objektorientierter Programmierung eine

sinnvolle Anwendung im Zusammenhang mit der Berechnung nach der Finite–Elemente–Methode findet.

Die Simulation komplexer Vorgänge reicht zum Teil weit über die Grenzen des herkömmlichen Ingenieurwesens hinaus. Die damit einhergehende Beschreibung spezieller Datentypen und den zugehörigen Funktionalitäten hat die Entwicklung objektorientierter Programmiersprachen in den vergangenen Jahren stark vorangetrieben. Darin werden die Daten und Funktionen anders als in den prozeduralen Programmiersprachen nicht mehr unabhängig voneinander betrachtet.

Das Paradigma der objektorientierten Programmierung versucht Objekte zu schaffen, die den Objekten des zu simulierenden Systems entsprechen. Ein Objekt kapselt dabei seine eigenen Daten und verwaltet sie mit den zugehörigen Methoden.

Wenn mehrere Objekte einen identischen Aufbau haben, sich lediglich in der Wertebelegung unterscheiden, können sie einer Klasse untergeordnet werden. Sie dient als Muster und definiert die Datenstruktur und Methoden der einzelnen Objekte, die auch als Instanzen der Klasse bezeichnet werden. Klassen können hierarchisch aufgebaut werden. Unterklassen erben dabei alle Eigenschaften ihrer Oberklasse und können um weitere Funktionalitäten ergänzt werden. Dadurch ist eine hohe Modularität gewährleistet. Schnittstellen nach außen sind schnell und unproblematisch zu generieren.

Die im Verlauf dieser Arbeit vorgestellte Methode zur Kraftflussberechnung an Scheiben beschreibt deren Tragwirkung durch Stabwerkmodelle aus Kraftflusspfaden. Im Zuge dessen sind unterschiedliche Teilaufgaben zu erledigen, die durch bestehende Softwarekomponenten und deren Kopplung übernommen werden.

Unter Anderem werden Spannungsberechnungen von einem Finite–Elemente–Programm (CARAT) benötigt. Ein eigenes objektorientiertes Fachwerkprogramm (TRUSS) übernimmt die Berechnung der Stabwerkmodelle. Der neu entwickelte Algorithmus zur Findung von Trajektorien (ALFIT) sorgt für die Ermittlung von ausgewählten Hauptspannungstrajektorien aus Spannungsfeldern. Ein Visualisierungstool (AVS/Express) ist für die graphische Aufbereitung und Darstellung zuständig.

Wenn eine hohe Modularität und damit eine problemlose Austauschbarkeit gefordert wird, macht es Sinn jede Applikation als eigenständiges Objekt zu betrachten, mit den zugehörigen Methoden, der Verwaltung seiner eigenen Datenhaltung und klar definierter Schnittstellen nach außen. Deswegen wird im Folgenden eine objektorientierte Umsetzung des Konzepts zur Rechnerunabhängigen Kopplung verfolgt, um die Aufgabe der Kraftflussberechnung zu lösen.

Die dafür notwendige Kommunikation zwischen unterschiedlichen Programmiersprachen und Betriebsystemen wird durch das objektorientierte Paradigma erleichtert.

2.4 Mixed Language Programming (MLP)

Es gibt mehrere Gründe, welche zur Generierung von Programmpaketen führen, in denen verschiedene Programmiersprachen miteinander vermischt werden. Jede Programmiersprache hat seine Vor- und Nachteile. Ziel der gemischt–sprachigen Programmierung ist unter anderem die Nutzung der Vorteile der unterschiedlichen Sprachen, ohne deren Nachteile in Kauf nehmen zu müssen.

Ein weiteres Ziel ist die Verwendung bereits bestehender Programmteile oder kompletter Softwareapplikationen zur Verknüpfung zu neuartigen Programmsystemen. Alter Quellcode in alten Programmiersprachen ist meist ausgereift und gut dokumentiert, kann aber meist nicht ohne Weiteres portiert werden. Neue Anforderungen an Software macht die Verwendung von modernen Programmiersprachen notwendig. Die Kombination alter und neuer Bausteine in einem Softwarepaket wird als *Mixed Language Programming* bezeichnet und wirft allerhand Fragen auf, die programmiertechnisch gelöst werden müssen.

Zu den im Ingenieurwesen am häufigsten verwendeten Programmiersprachen gehört zweifelsohne FORTRAN. Durch deren Aufbau und Sprachumfang ist diese Sprache immer noch hervorragend zur Umsetzung komplexer Algorithmen in effizienten Maschinencode geeignet. Gerade im Ingenieurwesen hat sich daraus ein umfangreicher Fundus an ausgereiften Problemlösungen angesammelt, auf die im Sinne effizienter Softwareentwicklung auch zurückgegriffen werden sollte. Der Nachteil von FORTRAN liegt in der gering ausgeprägten Unterstützung bei der Strukturierung von Daten, was durch aufwendige Prozeduren zur Datenhaltung wett gemacht werden muss.

Anfang der 70er Jahre kam mit C durch Ken Thompson und Dennis Ritchie eine neue systemunabhängige Programmiersprache auf [Ker90]. Sie besitzt umfassende Möglichkeiten zur Beschreibung und Strukturierung der Daten und des Programmablaufs.

C++ ist eine objektorientierte Sprache, entwickelt unter der Leitung von Bjarne Stroustrup [Str98], welche alle Funktionen von C enthält. Ihr Konzept ist die Integration von algorithmischen Programmteilen und Strukturierung der Daten. Die Entwicklung von Programmen zur computergestützten Simulation komplexer technischer Probleme wird damit erheblich erleichtert, im Sinne der Korrektheit, Robustheit, Erweiterbarkeit und Wiederverwendbarkeit.

Für die in der vorliegenden Arbeit notwendige Kombination von FORTRAN, C, und C++ ist in Anhang B eine technische Anleitung an Hand eines kurzen Beispiels gegeben.

2.5 Das Konzept der Software–Kapselung

Die Software–Kapselung ist eine Technik für die Einbindung bestehender Software–Bausteine in neue Applikationen ohne den Aufwand der umständlichen Konvertierung oder Neuentwicklung. Ein häufiger Anlass für die Kapselung ist bei dynamischen Verbindungen zweier Anwendungssysteme gegeben. Diese sind immer dann notwendig, wenn ein Prozess einer Anwendung einen anderen Prozess einer entfernten Anwendung aufruft (Remote Procedure Call) oder direkt auf sich dort befindliche Daten zugreifen muss (Remote Method Invocation). In beiden Fällen handelt es sich um eine Datenfernübertragung zwischen Systemen in unterschiedlichen Umgebungen. Die *Common Object Request Broker Architecture* (CORBA), *Distributed Component Object Model* (DCOM) oder auch *MQ–Series* dienen eben diesem Zweck.

Die Software–Kapselung ist die allgemeinste Form des Begriffs *wrapping*, der im Zusammenhang mit der Einbindung bestehender Software in objektorientierte Systeme geprägt wurde. Dabei wird die eigentliche Implementierung sauber von der Schnittstelle getrennt. Die Umsetzung ist nach Mowbray und Zahavi [Mow99] je nach Anforderung auf unterschiedliche Weise möglich:

- über entfernte Prozeduraufrufe (RPC)
- durch Dateiübergabe
- mit Sockets bzw. Andockungsanschlüssen (Application Programming Interface, API)
- über eine Anwendungsprogrammschnittstelle
- mit Hilfe von Skriptprozeduren
- mittels Makrotechnik
- durch gemeinsame Header–Dateien

In dieser Arbeit findet ausschließlich die Prozesskapselung mit Sockets auf Basis einer Client–Server–Beziehung Anwendung.

Je nach Umfang des gekapselten Bereichs kann unterschieden werden in Prozedur-, Programm- und Prozesskapselung. Wird ein vollständiger Prozess in einer Kapsel eingeschlossen und nach außen definierte Schnittstellen geschaffen, so kann dieser als Objekt in einem größeren objektorientierten System agieren.

Werden die notwendigen Methoden geschaffen, können die Objekte sogar in einem heterogenen System verteilt sein. Damit kann das Paradigma der objektorientierten Programmierung auf ganze Programmsysteme erweitert werden.

Kapitel 3

Methoden zur Erzeugung von Stabwerkmodellen

Seit ihrer Entwicklung haben Stabwerkmodelle bei der Bemessung von Stahlbetonbauteilen immer mehr an Bedeutung gewonnen. Das von Schlaich und Schäfer [Sch93] beschriebene Verfahren stellt eine Verallgemeinerung der Bemessung mit der Fachwerkanalogie nach Mörsch [Mör12] dar. Damit konnte der Anwendungsbereich des Bemessungskonzepts von balkenartigen auf vornehmlich ebene Bauteile erweitert werden.

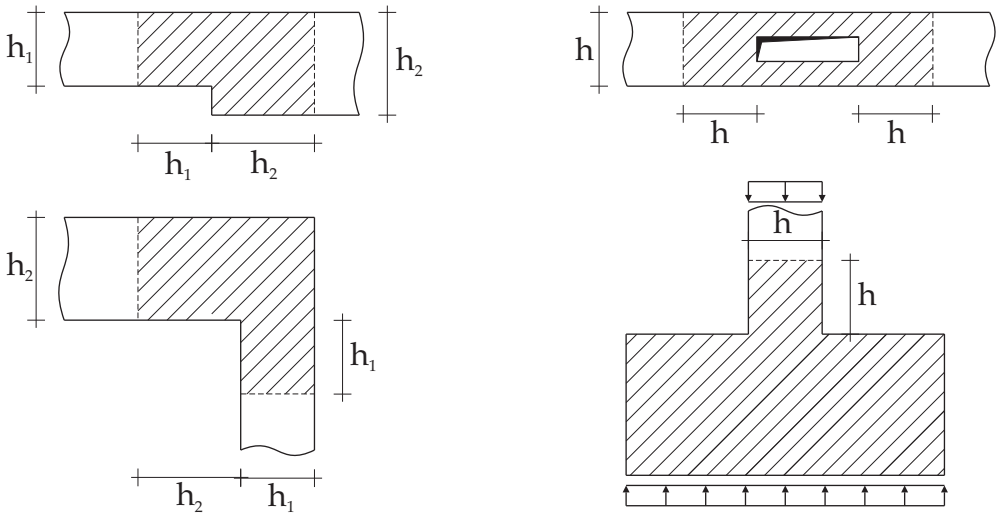
Stabwerkmodelle sind aber nicht nur als Bemessungshilfe nützlich, sie tragen im Wesentlichen zum besseren Verständnis des Tragverhaltens auch komplexer Tragwerke bei. In einem ersten Überblick werden einige bekannte Methoden zur Bestimmung von Stabwerken kurz vorgestellt.

Im weiteren Verlauf wird eine eigene, überwiegend automatisierte Methode zur Generierung von optimierten Stabwerken aus Kraftflusspfaden beschrieben, die sowohl lokales wie globales Gleichgewicht gewährleisten. Unter Wahrung der Kompatibilität werden die Steifigkeitsverhältnisse der Stäbe derart bestimmt, dass das Stabwerkmodell das kontinuierliche Problem richtig beschreibt.

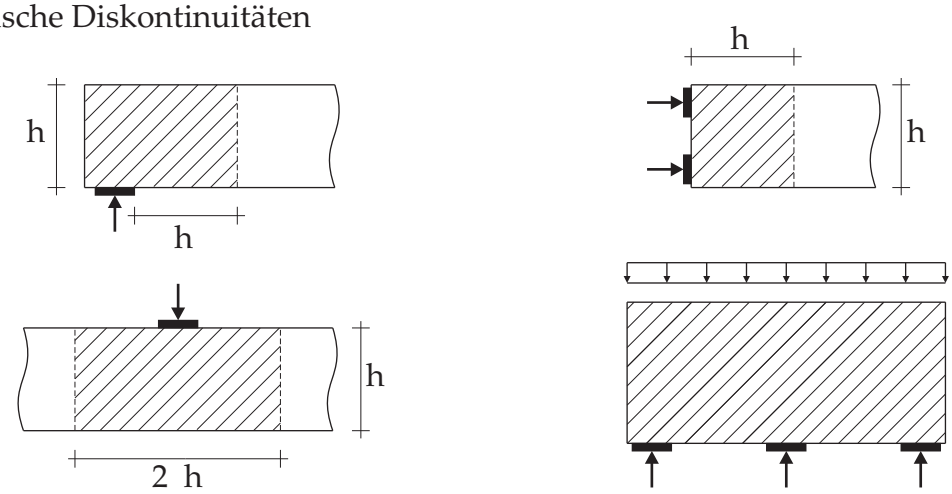
3.1 Stabwerkmodelle als Bemessungshilfe für Stahlbetontragwerke

Schlaich und Schäfer [Sch93] schlagen für die Bemessung von Stahlbetonscheiben eine Unterteilung des Tragwerk in zwei Bereiche vor, die sich im Tragverhalten wesentlich unterscheiden. Das sind zum einen Teile mit Diskontinuitäten (D–Bereiche), und zum anderen hauptsächlich auf Biegung beanspruchten Teile (B–Bereiche). Für die zuletzt genannten Bereiche ist die Bernoulli–Hypothese vom Ebenbleiben der Querschnitte ausreichend genau erfüllt. Während dafür Standardverfahren zur Bemessung existieren, werden die D–Bereiche meist nur „konstruktiv“ erfasst. An diesen Orten ist eine lineare Dehnungsverteilung nicht mehr gegeben.

a) geometrische Diskontinuitäten



b) statische Diskontinuitäten



b) geometrische und statische Diskontinuitäten

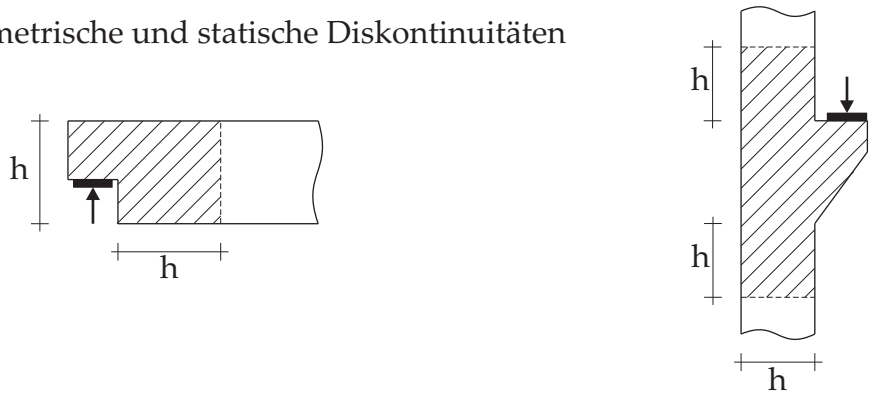


Abbildung 3.1: Bereiche mit Diskontinuitäten bei Scheibentragwerken

Abbildung 3.1 zeigt Tragwerksteile mit geometrischen und/oder statischen Diskontinuitäten, die in den Skizzen jeweils schraffiert eingezeichnet sind.

Dargestellt sind im Wesentlichen Änderungen der Tragwerksform bzw. Orte mit konzentrierter Lasteinleitung. An eben jenen Bereichen treten meist auch Singularitäten in Finite–Elemente–Berechnungen auf, was die automatische Bemessung zudem erschwert, oder gar unmöglich macht.

Für eine hohe Qualität und ausreichende Sicherheit von Tragwerken muss auch für diese Bereiche ein durchgängig nachvollziehbares Verfahren zur Bemessung vorgeschrieben werden. Durch geeignete Stabwerkmodelle kann die Lastabtragung durch innere Kräfte im gesamten Tragwerk systematisch erfasst und anschaulich dargestellt werden.

Im heute bestehenden *Eurocode 2* [Eur92] wird daher die Bemessung unter anderem mit Hilfe von Stabwerkmodellen für Bauteile mit einer nichtlinearen Dehnungsverteilung vorgeschlagen (D–Bereiche). Darin heißt es, dass kontinuierliche Tragwerke als statisch bestimmte Stabwerke idealisiert werden können.

Für die Verträglichkeit der Modelle sollen sich die Stäbe an der Schnittgrößenverteilung nach der Elastizitätstheorie richten. Deren Normalkräfte ergeben sich dann aus den Gleichgewichtsbedingungen. Zugkräfte sollen durch die Bewehrung und Druckkräfte durch den Beton und eventuell zusätzliche Druckbewehrung aufgenommen werden. Zudem sind die Knoten der Stabwerkmodelle wegen des mehrachsialen Spannungszustandes genauer zu untersuchen, sowie Verankerungslängen der Bewehrungsstäbe einzuhalten [Sch93].

Bei statisch unbestimmten Problemen ist jedoch die Kenntnis von den Steifigkeitsverhältnissen der Stäbe notwendig, um eine Aussage über deren Schnittgrößen treffen zu können. Dies ist bis dato noch nicht richtig gelöst. Näherungsverfahren z.B. auf Basis von Einflussbreiten wie bei Rückert [Rüc92] geben zwar einen Anhaltspunkt dafür, jedoch ohne auf ein fundiertes Kriterium zurückzugreifen.

Die Bemessung von Stahlbetonbauteilen an sich steht nicht im Mittelpunkt dieser Arbeit, sondern vielmehr die Bereitstellung eines computergestützten Hilfsmittels zur Erzeugung von Stabwerkmodellen, die mit Hilfe der Kontinuitätsbedingung auch den Kraftzustand im zu beschreibenden Kontinuum richtig abbilden kann. Sie sollen als Grundlage für die Modellierung von Stahlbetonscheiben dienen und ein methodisches Konstruieren, Bemessen und Optimieren ermöglichen.

3.2 Vom Kraftfluss zum Stabwerkmodell

Der Begriff *Kraftfluss* ist in der Literatur nicht eindeutig definiert. In verschiedenen Disziplinen wird er auf ganz unterschiedliche Weise gedeutet.

Im Maschinenwesen beispielsweise ist damit kein *Fluss* im wörtlichen Sinn gemeint. Unter Kraftfluss wird hierbei der Weg der Kraftübertragung über mechanische Bauteile verstanden, von der Quelle der Kraft bis hin zu ihrer Senke, z.B. vom Motor über das Getriebe bis zu den Antriebsrädern.

Im Gegensatz dazu wird in der Physik unter dem Fluss ϕ ganz allgemein eine pro Zeiteinheit durch eine gegebene Querschnittsfläche hindurchtretende Menge verstanden. Er ist definiert als das skalare Produkt aus Vektorfeld $\vec{F}$ und Fläche $\vec{A}$ nach Gleichung 3.1.

$$\phi = \vec{F} \cdot \vec{A} = A \cdot \vec{F} \cdot \vec{n} \tag{3.1}$$

Bekannte Beispiele sind unter Anderem der magnetische Fluss, der elektrische (Verschiebungs-) Fluss und der Wärmefluss.

Überträgt man zunächst die erste Definition ins Bauingenieurwesen, kann der Kraftfluss als Pfad gedeutet werden, entlang dem die Kraft von der Lasteinleitung bis hin zum Auflager verläuft. Ein Fluss im Sinne von bewegter Materie ist dies jedoch nicht. Vielmehr wird dadurch ein statischer Gleichgewichtszustand des Systems betrachtet.

Eine einfache und direkte Deutung vom Kraftfluss ist an scheibenartigen Bauteile durch den meist recht komplexen Spannungszustand oft schwierig. Der Weg zum besseren Verständnis der Tragmechanismen ist in der Diskretisierung des kontinuierlichen Problems zu suchen. Dabei hilft die Darstellung der Hauptspannungstrajektorien. Betrachtet man ein infinitesimales Element, dessen Form und Ausrichtung durch Hauptspannungstrajektorien definiert ist, treten an dessen Rändern per Definition nur die Hauptspannungen σ_1 und σ_2 auf (vgl. Abbildung 3.2). Zusammen mit den passenden Querschnittsflächen dA_1 und dA_2 können damit zwei orthogonale Kraftflüsse beschrieben werden, analog zu Gleichung 3.1.

Bereiche des Tragwerks mit annähernd gleichem Verhalten werden zusammengefasst und durch einfache Fachwerkelemente ersetzt (s. Abbildung 3.6). Sie haben einen konstanten und gerichteten Spannungszustand über die komplette Querschnittsfläche A, woraus sich eine Resultierende Normalkraft N durch Gleichung 3.2 berechnen lässt.

$$N = \int_A \sigma \, dA \tag{3.2}$$

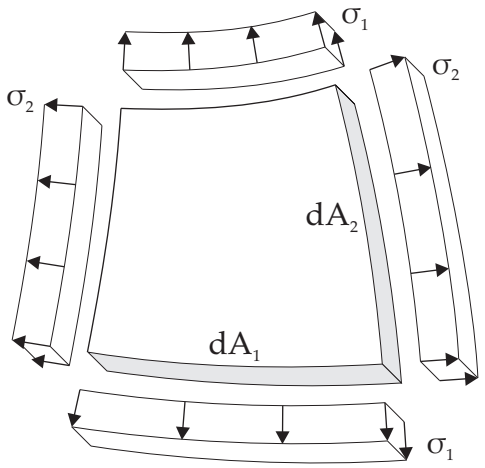


Abbildung 3.2: Bestimmung des Kraftflusses am infinitesimalen Element

Sie ändert weder Betrag noch Richtung innerhalb des Stabes. Damit weisen sie die einfachste Form der Lastabtragung auf. In Abbildung 3.3 sind die Zusammenhänge am Fachwerkstab zur Verdeutlichung dargestellt.

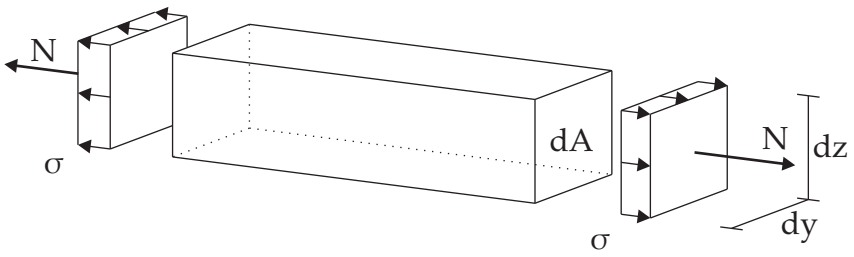


Abbildung 3.3: Beziehungen am Fachwerkstab

In diesem Fall ist die Normalspannung σ die vektorielle Größe, welche auf der gesamten Querschnittsfläche des Stabes dA wirkt. Durch die Analogie der Gleichungen 3.1 und 3.2 kann die Normalkraft der Fachwerkstäbe auch als Kraftfluss gedeutet werden.

Nutzt man derartige Fachwerkelemente, um kontinuierliche Systeme diskret zu beschreiben, kann der Kraftfluss in Form von Normalkräfte der Stäbe sehr einfach sichtbar gemacht werden. Dabei werden zwei wesentliche Aspekte in der Deutung des Tragverhaltens hervorgehoben. Zum einen wird die Richtung der inneren Kräfte bereichsweise aufgezeigt und zum anderen ein lückenloser Kraftfluss im System beschrieben. Das Gleichgewicht ist im gesamten System eingehalten. Abbildung 3.4 zeigt am Beispiel einer Kragscheibe deren Modellierung durch ein denkbares Stabwerk.

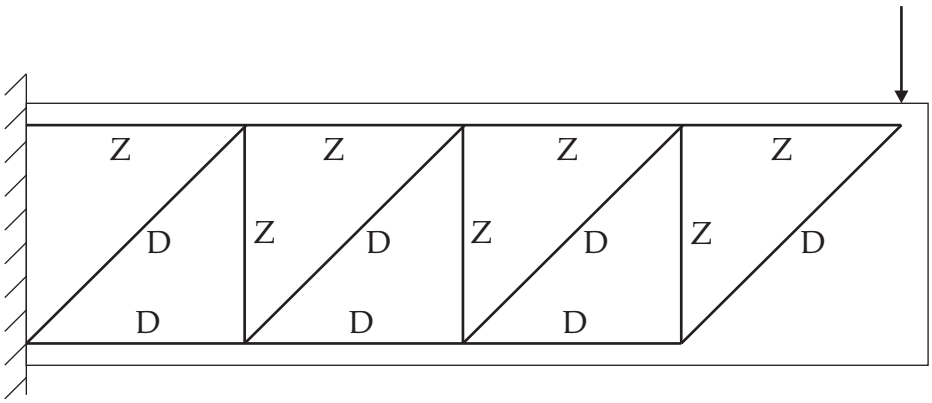


Abbildung 3.4: Stabwerkmodell am Beispiel eines Kragarms

Alle Normalkräfte können unter Einhaltung des Gleichgewichts berechnet werden. Das Biegemoment der tatsächlichen Kragscheibe wird durch ein Kräftepaar von Ober- und Untergurt aufgenommen. Die vertikalen und diagonalen Stäbe übernehmen die Querkraft aus Schub.

Neben dem hier aufgeführten Modell existieren weitere schlüssige Stabwerkmodelle. Hier stellt sich die Frage nach einem optimierten Stabwerk, welches den tatsächlichen Zustand am besten wiedergeben kann.

Durch die Diskretisierung des Spannungszustandes kann der Weg der Kraft vom Lastangriffspunkt bis ins Auflager nachvollzogen werden. In diesem Sinne definiert Wiedemann [Wie96] in Anlehnung an Michell [Mic04] Kräftepfade als Fachwerk- oder Netzstrukturen, bei denen die einzelnen Elemente jeweils einachsig tragen.

Im weiteren Verlauf wird der Begriff *Kraftflussberechnung* als Ermittlung und Berechnung eines optimierten Stabwerkmodells verstanden. Die Darstellung des Kraftflusses wird in diesem Zusammenhang als Visualisierung der Tragwirkung eines Bauteils durch ein geeignetes Stabwerkmodell gedeutet.

Das dient nicht nur dem leichteren Verständnis vom Tragverhalten. Gleichzeitig kann damit für Stahlbetonbauteile eine vernünftige und wirtschaftliche Bewehrungsführung gefunden werden. Vor allem für komplexe und neuartige Geometrien wird dadurch eine sinnvolle Alternative zu den recht konservativen Bemessungsmodulen gängiger FE–Programme eröffnet.

Im Fall einfacher Tragwerke mit definierten Spannungsverteilungen, wie dies bei überwiegend Biegebeanspruchten Bauteilen der Fall ist (B–Bereiche), ist die Findung eines passenden Stabwerkes leicht möglich, und bereits in mehreren Arbeiten untersucht worden. An dieser Stelle sei auf Mörsch [Mör12], dem Begründer der Fachwerkanalogie verwiesen.

3.3 Bestehende Ansätze zur Stabwerk–Modellfindung von scheibenartigen Tragwerken

Findet man ein in sich im Gleichgewicht befindliches Stabwerkmodell, ohne dass die Fließgrenze des Werkstoffs überschritten wird, ist die linear-elastische Berechnung für eine Bemessung ausreichend [Sch93].

Wird der Bereich linearer Dehnungs- und Spannungsbereiche (B-Bereiche) bei Bauteilen verlassen (vergleiche Abbildung 3.1), so ist die Findung von Stabwerkmodellen in der Regel nicht einfach.

Derzeit existieren unterschiedliche manuelle und halbautomatische Ansätze zur Entwicklung von Stabwerkmodellen. Aus den damit verbundenen Untersuchungen resultiert ein Sammelsurium an Standardfällen, wie sie beispielhaft in [Sch93] oder [Leo77] aufgeführt werden. In Abbildung 3.5 sind drei wesentliche Beispiele herausgegriffen. In den dazugehörige Stabwerkmodellen sind Zug- (Z) und Druckstäbe (D) gekennzeichnet.

Im Folgenden werden einige Methoden aus der Literatur aufgeführt und eine Abgrenzung zur eigenen Arbeit vorgenommen. Das anschließende Kapitel mit numerischen Berechnungen greift unter anderem obige Beispiele wieder auf.

3.3.1 Modellfindung von Hand

Für sehr einfache oder bekannte Fälle ist die Konstruktion von Hand möglich (vgl. [Rüc92]). Der zeitliche Aufwand ist jedoch hoch und die Ergebnisse subjektiv und meist nicht optimal. Eine Verbesserung verspricht die Zuhilfenahme von Hauptspannungsfeldern aus linear–elastischen FE–Berechnungen. An ihnen lassen sich die Stäbe leichter orientieren, gerade bei etwas schwierigeren oder unbekannten Problemen. Hierbei werden Bereiche mit ähnlicher Ausrichtung der Hauptspannungen zusammengefasst und durch Stäbe ersetzt (vgl. Abschnitt 3.2). Die Wahl der Bereiche und damit die Anzahl der Stäbe bereitet die größten Schwierigkeiten. Abbildung 3.6 zeigt die Einfeldscheibe mit zwei Einzellasten und das dazu gehörige Hauptspannungsfeld aus linear elastischer Rechnung.

Bereiche mit sehr kleinen Spannungen tragen wenig zur Lastabtragung bei, weswegen sie ausgeblendet werden. Andere Bereiche zeigen eine starke Ausrichtung der Hauptspannungen, die jeweils durch einen Stab zusammengefasst werden. Symmetriebedingungen können hier ebenfalls mit in die Betrachtungen einfließen.

Eine anschließende Berechnung des Stabwerkmodells von Hand ist auch bei kinematischen Stabwerkmodellen ohne Probleme möglich, während Stabwerksprogramme nicht ohne Weiteres angewendet werden können.

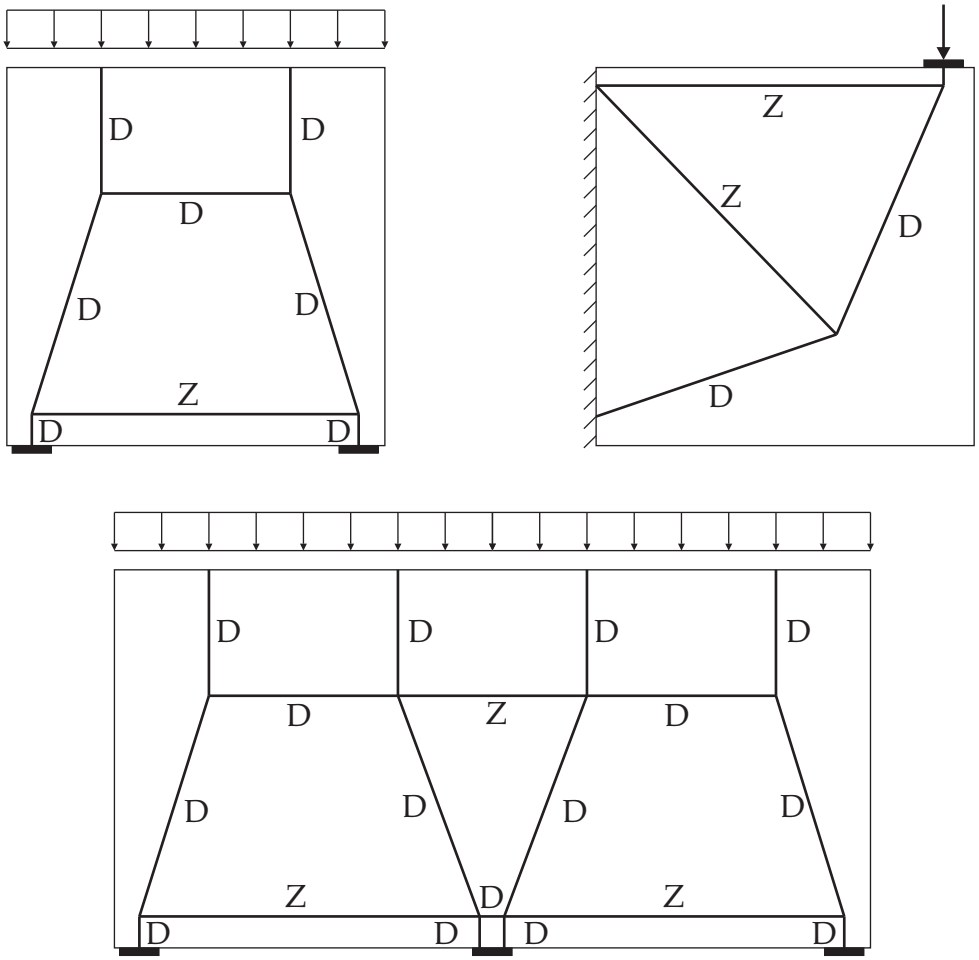


Abbildung 3.5: Stabwerkmodelle unterschiedlicher Scheibenprobleme

3.3.2 Modellierung anhand vorgefertigter Muster

Eine andere Herangehensweise ist die Anwendung von Musterstabwerken, die nur noch auf die tatsächlichen Bauteilabmessungen skaliert werden müssen (vgl. [Rüc92]). Der Nachteil hierbei ist die eingeschränkte Anwendbarkeit durch einen vorgefertigten Musterkatalog. Durch die Skalierung stellt sich dabei auch nicht unbedingt die optimale Geometrie ein. Das Beispiel einer Konsole soll die Problematik der Verzerrung verdeutlichen.

Abbildung 3.7 zeigt links das Musterstabwerk für eine Konsole. Zugstäbe sind gestrichelt, Druckstäbe mit durchgezogener Linie eingezeichnet. Wird die Geometrie, wie in der Mitte zu sehen ist, nur leicht verzerrt, kann das zugehörige Stabwerk noch ohne Probleme mit skaliert werden. Im rechten Teil des Bildes sind die Seitenverhältnisse der Konsole derart stark verändert worden, dass hier die Balkentheorie Anwendung findet. Das Stabwerk für einen Kragarm beschreibt das Tragverhalten in diesem Fall am besten.

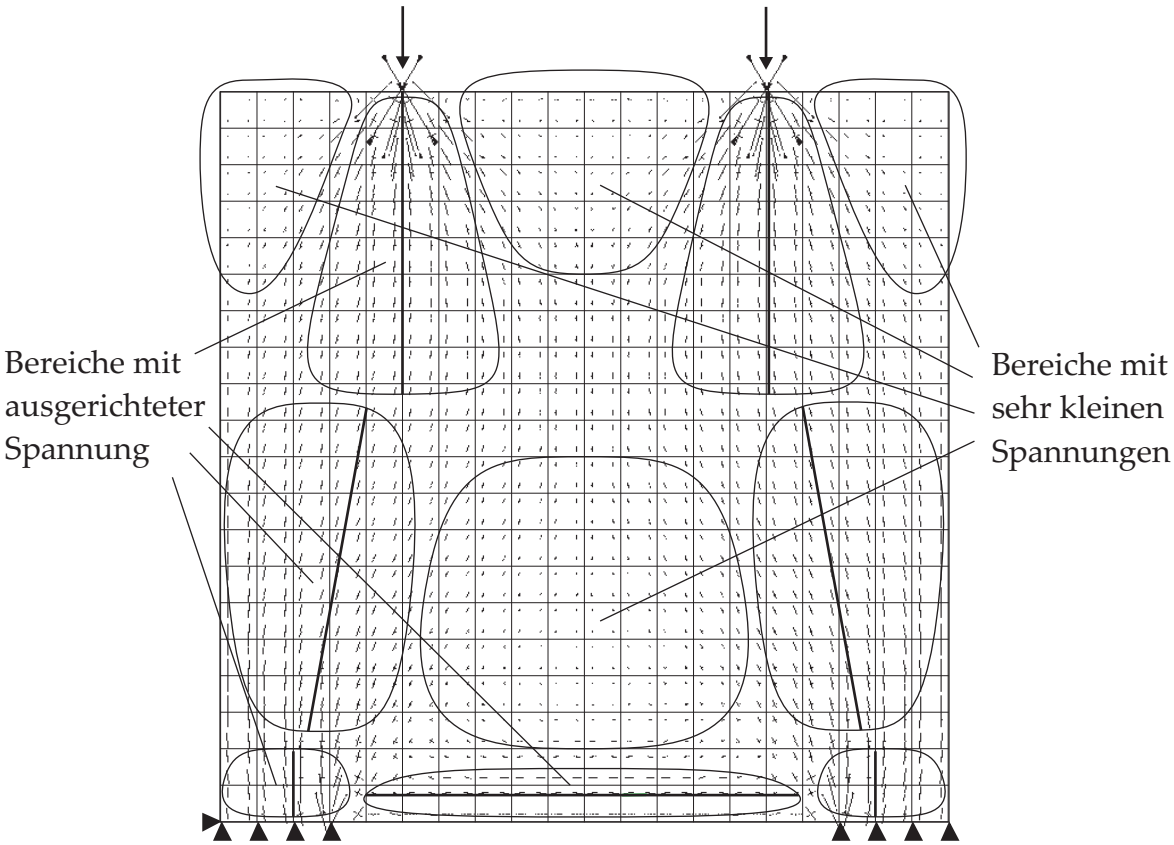


Abbildung 3.6: Modellfindung von Hand mit Hilfe des Hauptspannungsfeldes

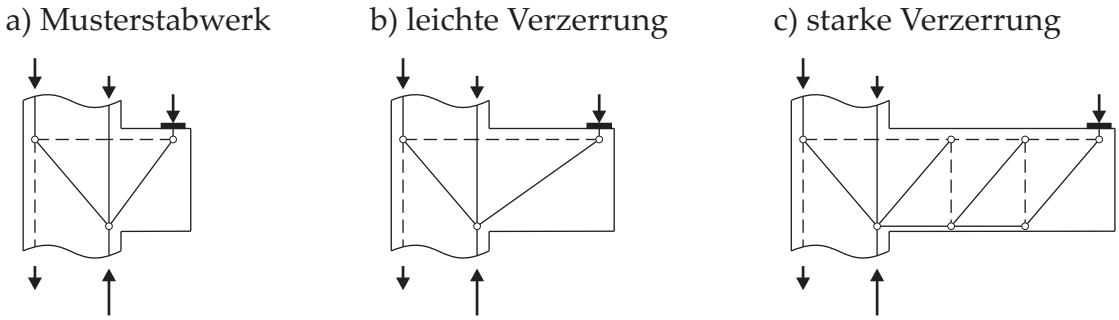


Abbildung 3.7: Problem der Stabwerkmodelle aus vorgefertigten Mustern

3.3.3 Modellfindung mit Hilfe der Lastpfadmethode

Die von Jennewein [Jen89] beschriebene *Lastpfadmethode* geht von der Vorstellung aus, dass die Last von ihrer Quelle zu den Auflagern verläuft. Die Lage des Lastpfades lässt sich punktweise durch die Spannungsresultierenden an geeigneten Schnitten ermitteln. Diese müssen vom Anwender festgelegt werden, um dann daraus ein realitätsnahes Modell für die Lastabtragung gewinnen zu können. Da die Aussagekraft des Modells maßgeblich von der Schnittführung abhängt, ist auch diese Methode nicht objektiv und hängt stark von der Erfahrung und dem Geschick des

Anwenders ab. Abbildung 3.8 zeigt in einem vertikalen und einen horizontalen Schnitt die jeweiligen Verläufe der darauf senkrechten Spannungen und deren Resultierende.

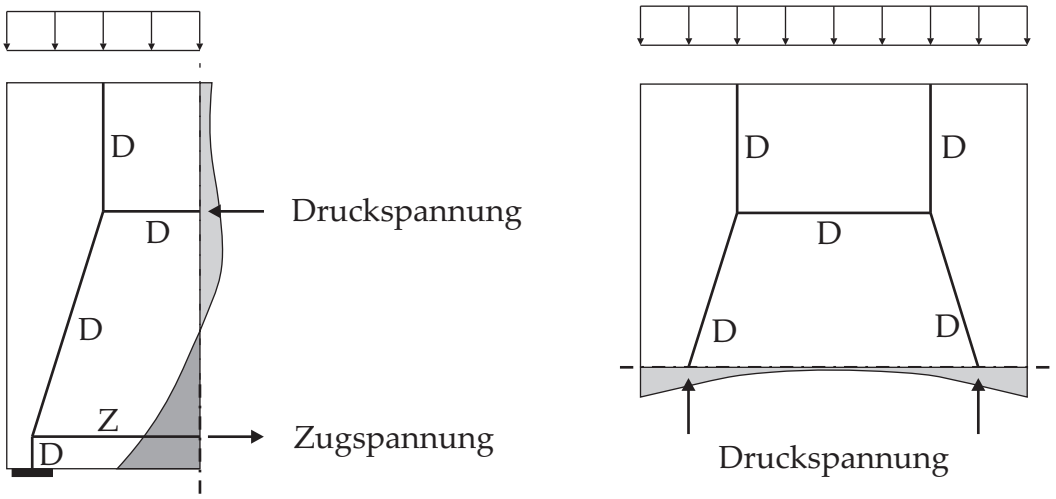


Abbildung 3.8: Spannungen und resultierende Kräfte am Schnitt

Die rechte Seite der Abbildung macht einen weiteren Nachteil der Methode klar. Am horizontalen Schnitt treten auch Schubspannungen auf, die jedoch nicht berücksichtigt werden. Schnitte, die an jeder Stelle senkrecht zur Hauptspannungsrichtung verlaufen, wären besser geeignet. Orthogonalen Trajektorien zum Beispiel erfüllen diese Voraussetzung.

3.3.4 Modellierung aus Trajektorienfeldern

Rückert schlägt in seiner Dissertation [Rüc92] vor, Stabwerkmodelle aus Trajektorienfelder zu generieren. Bei diesem Verfahren wird zunächst ein Trajektoriennetz mit bestimmbarer Dichte geschaffen. Zwei Paare definieren jeweils einen Bereich, an dem Gleichgewicht herrscht (vgl. Bild 3.9 links). Die Resultierenden an den Rändern repräsentieren den Spannungszustand und befinden sich im Gleichgewicht. Wenn sie als Stäbe interpretiert werden, entsteht daraus ein kinematisches Stabwerk (vgl. Bild 3.9 rechts). Die Resultierenden liegen wieder auf einer Trajektorie. Dieser Umstand wird für die in dieser Arbeit näher beschriebene Methode zur Ermittlung von Stabwerkmodellen genutzt.

Das derart erzeugte Stabwerksystem muss nun interaktiv den Anforderungen angepasst werden. Dabei wird die Dichte der Trajektorien verringert, Lasten und Lagerungsbedingungen eingefügt und Stäbe an die Randbedingungen angepasst. Das erfordert tiefer reichende Kenntnisse vom Ingenieur, ist zeitraubend und kann nicht in ein voll automatisiertes Programmsystem eingebettet werden. Der Vorteil einer interaktiven Eingriffsmöglichkeit ist dafür ihre Variabilität.

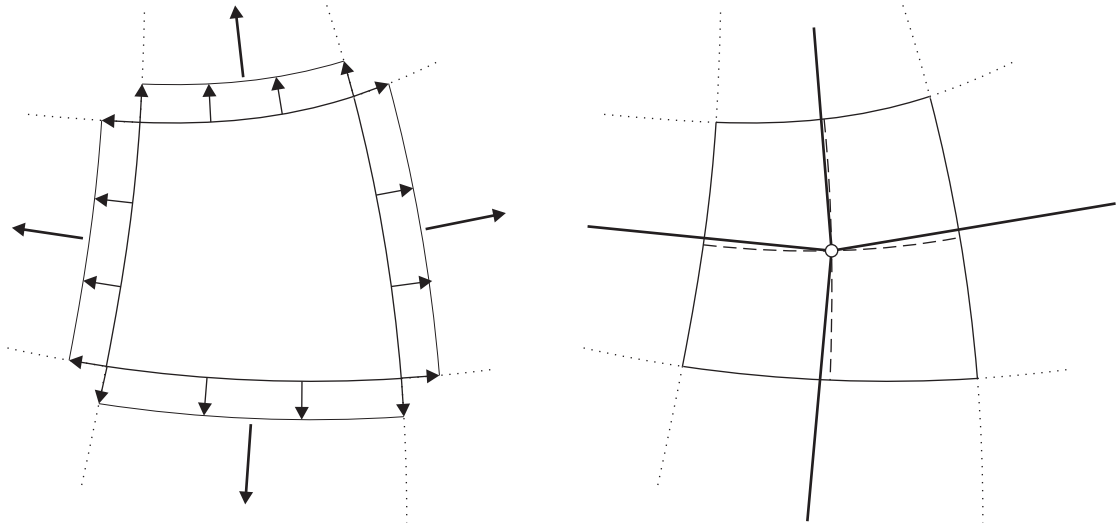


Abbildung 3.9: Trajektorienschar und Stabnetz

3.3.5 Modellierung anhand von Differentialgleichungen

Eine etwas andere Herangehensweise liefert Fonseca in seiner Dissertation [Fon95]. Sie beruht auf der Analogie der Differentialgleichungen der Gleichgewichtsbedingungen in der Statik und der Kontinuitätsgleichung in der Strömungslehre. Bei Scheiben können damit zwei „Kraftflüsse“ betrachtet und analog wie Stromlinien gezeichnet werden. Unter Vernachlässigung von Kräften im Gebiet kann das Gleichgewicht für den ebenen Spannungszustand in zwei globale Richtungen abgeleitet werden (vgl. Abbildung 2.1 rechts).

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} = 0 \tag{3.3}$$

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} = 0 \tag{3.4}$$

Das Pendant dazu ist die Kontinuitätsgleichung einer quellenfreien, zweidimensionalen Strömung. Betrachtet wird der stationäre Zustand eines inkompressiblen Fluids.

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 0 \tag{3.5}$$

Die Strömungsgleichung berücksichtigt beide globale Richtungen gemeinsam und beschreibt damit das Geschwindigkeitsfeld. Daraus lassen sich Trajektorien zu so genannten Stromlinien ableiten.

Die Gleichungen 3.3 bzw. 3.4 hingegen beschreiben jeweils nur eine globale Spannungsrichtung. Damit ist die Anwendung auf wenige Beispiele mit überwiegend vertikalem oder horizontalem Kraftfluss eingeschränkt. Kombinationen aus horizontalen und vertikalen Lasten und/oder Lagern sind nur bedingt zu erfassen.

3.3.6 Michell–Strukturen

Wiedemann [Wie96] beschreibt den Entwurf von Fachwerken nach dem Michellprinzip durch Lineare Programmierung. Er stellt die Frage nach den optimalen Kräftepfaden bei vorgegebenem Entwurfsraum und Belastung, und will sie durch Fachwerkstäbe beschreiben. Damit ist er nur ein Vertreter des Gebiets der Topologieoptimierung. Ein Überblick findet sich z.B. bei Maute [Mau98].

Entwurfsgeometrische Grundsätze zur Bestimmung derartiger Kräftepfade gehen auf Maxwell [Max69] und Michell [Mic04] zurück. Michell wies nach, dass optimale Kräftepfade gewissen Orthogonaltrajektorien folgen müssen. Sie zeichnen sich dadurch aus, dass entlang ihres Verlaufs die größten Dehnungsbeträge auftreten. Mit anderen Worten verlaufen die Kräftepfade entlang der orthogonalen Hauptdehnungstrajektorien oder Hauptspannungstrajektorien.

Jedoch konnte Michell nur Lösungen für ganz bestimmte einfache Lastfälle und unter idealisierenden Voraussetzungen finden. Abbildung 3.10 zeigt die Grundformen und verformungsverträgliche Kombinationen davon. Erst durch Arbeiten von Cox [Cox58] und Hemp [Hem58] sind derartige Michellstrukturen auch für beliebige Lastaufnahmen anzuwenden.

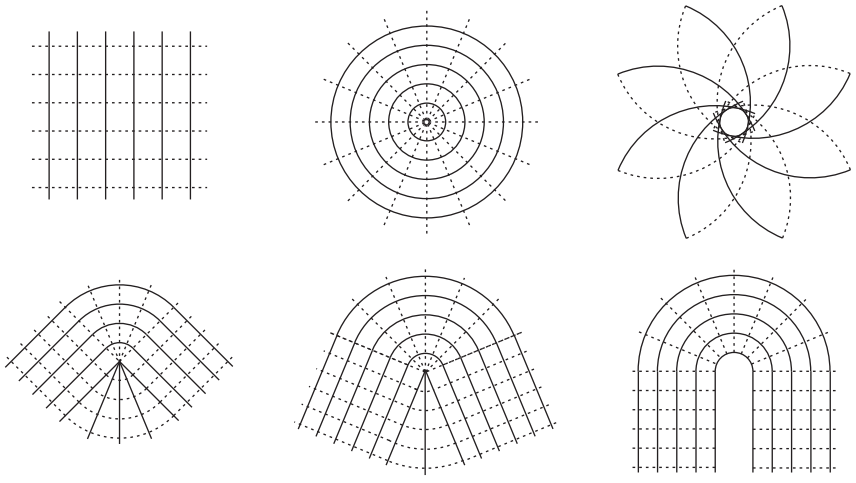


Abbildung 3.10: Michellstruktur

3.4 Entwicklung einer Methode zur automatischen Generierung optimierter Stabwerkmodelle

Die Darstellung des Kraftflusses wird, wie bereits erwähnt, als Visualisierung der Tragwirkung eines Bauteils durch ein geeignetes Stabwerkmodell gedeutet. Im Zentrum der Betrachtung dieses Abschnitts steht die Kraftflussberechnung, also die Ermittlung und anschließende Berechnung der Stabwerkmodelle.

Die hier vorgestellte Methode generiert Stabwerkmodelle durch Orientierung von Fachwerkstäben an ausgewählten Hauptspannungstrajektorien, die hier als Kraftflusspfade gedeutet werden. Die Grundidee ist dabei der von Michell ähnlich, was jedoch nicht für dessen Einschränkungen in der Anwendbarkeit gilt.

Vorab sollen wichtige Anmerkungen zu den Hauptspannungstrajektorien und den Kraftflusspfaden zum Verständnis einer Kraftflussberechnung beitragen.

3.4.1 Hauptspannungstrajektorien

Der Begriff Trajektorie bedeutet übersetzt Bahnkurve. Im mathematischen Sinne werden Trajektorien genauer spezifiziert als Linien, welche jede Kurve einer Kurvenschar unter gleich bleibendem Winkel, meist orthogonal, schneidet.

In der Fluidmechanik sind Niveau- und Strömungslinien die häufigste Anwendung von Trajektorien. Im ersten Fall werden Punkte mit dem gleichen Gesamtenergieniveau miteinander verbunden. Strömungslinien hingegen werden als Verbindungslinien von Richtungstangenten des Geschwindigkeitsfeldes gedeutet und sind zu den Niveaulinien orthogonal.

In diesem Sinne wird eine Trajektorie nachfolgend als Kurve verstanden, die an jedem Punkt tangential zur Richtung eines Vektorfeldes verläuft. Magnetfelder, elektrische Felder, Strömungsfelder, Spannungsfelder, etc. sind derartige Vektorfelder, d.h. Gebiete in denen an diskreten Punkten Informationen über Ort, Wert der Feldgröße und deren Richtung bekannt sind. Abbildung 3.11 zeigt beispielhaft wesentliche Punkte eines Vektorfeldes, durch die eine Trajektorie beschrieben wird.

In der Strukturmechanik sind im Wesentlichen die Hauptspannungstrajektorien von Bedeutung. Das sind Verbindungslinien der Richtungstangenten der Hauptspannungen einer Struktur und damit zwei orthogonale Kurvenschaaren [Sch93]. Sie zeigen den Verlauf der Hauptspannungen im Gebiet und besitzen folgende Eigenschaften.

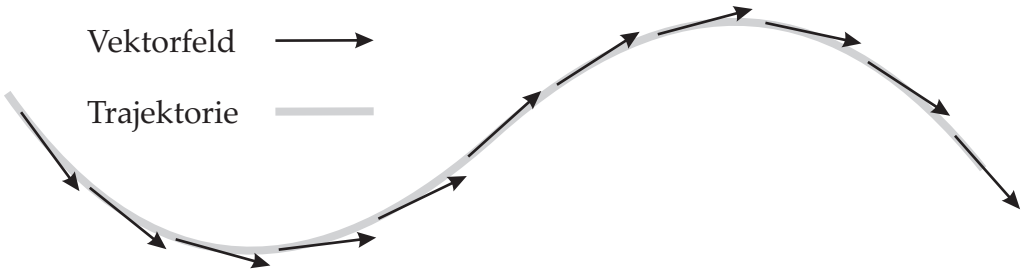


Abbildung 3.11: Vektorielle Größen eines Vektorfeldes mit zugehöriger Trajektorie

Orthogonalität

In jedem Punkt des Systems können Hauptspannungen aus der Bedingung heraus bestimmt werden, dass die Schubspannung verschwindet (siehe dazu Abbildung 3.17). Die Hauptspannungen σ_1 und σ_2 stehen senkrecht aufeinander. Die daraus resultierenden Hauptspannungstrajektorien schneiden sich deswegen stets orthogonal, wie Abbildung 3.12 zeigt.

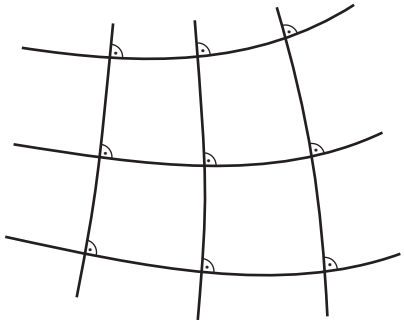


Abbildung 3.12: Orthogonalität von Hauptspannungstrajektorien

Betrag und Krümmung

Hauptspannungstrajektorien stellen Zug- und Druckspannungen dar. Entlang einer Trajektorie kann sich dabei sowohl der Betrag, als auch das Vorzeichen der betreffenden Hauptspannung ändern. Das hängt von der Krümmung der dazu orthogonal verlaufenden Trajektorie ab. Abbildung 3.13 verdeutlicht am Schnittpunkt zweier Trajektorienpaare die notwendige Betragsänderung der Spannung in vertikaler Richtung, wenn die horizontale Trajektorie gekrümmt verläuft. In Konsequenz dazu ändert sich in der vertikalen Trajektorie weder Betrag noch Vorzeichen, wenn die horizontalen Trajektorien nicht gekrümmt sind.

Bündeln sich Trajektorien, ist das ein Anzeichen dafür, dass auch deren Spannungswerte steigen, umgekehrtes gilt bei einer Streuung.

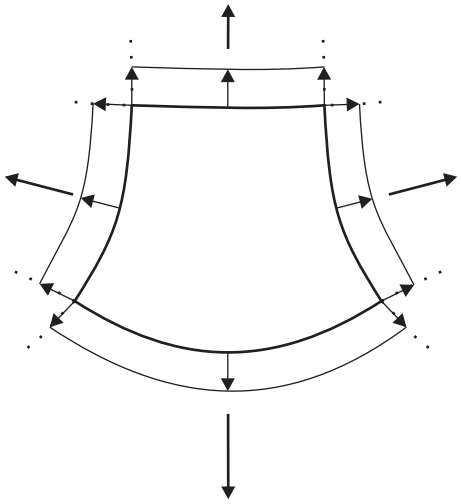


Abbildung 3.13: Krümmung und die daraus resultierende Betragsänderung

Systemränder

An Neumann–Randbedingungen orientieren sich Hauptspannungstrajektorien an der Richtung der von außen angreifenden Belastung (vgl. Abbildung 3.14 rechts). An unbelasteten Rändern sind die Spannungen und damit auch die Hauptspannungstrajektorien senkrecht zum Rand und sind von Betrag gleich Null. Infolge dessen verlaufen die orthogonalen Trajektorien in unmittelbarer Nähe des Randes parallel dazu, wie in Abbildung 3.14 links angedeutet wird.

Diriclet–Randbedingungen geben die Richtung der Kraftübertragung vor. Auch hier orientieren sich die Hauptspannungstrajektorien daran. Abbildung 3.14 links zeigt einen Rand mit horizontal verschieblichem Auflager.

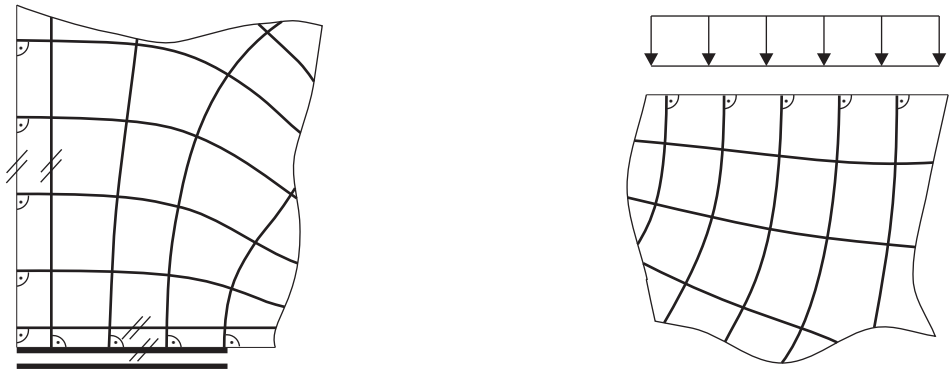


Abbildung 3.14: Diriclet- und Neumann–Randbedingungen

Bei einer automatisierten Suche nach Trajektorien müssen alle oben genannten Eigenschaften berücksichtigt werden. Der Verlauf der Hauptspannungstrajektorien innerhalb eines kontinuierlichen Systems orientiert sich an den Hauptspannungsrichtungen. Weitere allgemeine Aussagen können nicht gefolgert werden. In gewis-

sen Bereichen ähneln die Hauptspannungstrajektorien den Kraftflusspfaden, was bei der Kraftflussberechnung genutzt wird.

3.4.2 Kraftflusspfade

Im Folgenden werden möglichst einfache Stabwerkmodelle als Lösungsansatz zur Beschreibung des Tragverhaltens herangezogen (siehe dazu Kapitel 3). Bei hochgradig statisch unbestimmten Systemen, deren Tragwirkung bei Stahlbetonbauteilen zudem durch die Bewehrungsführung gesteuert werden kann, gibt es unterschiedliche Lösungen für den Kraftfluss.

Um eine generelle Aussage über das Tragverhalten eines Systems treffen zu können, wird in einer ersten Betrachtung das System ohne Bewehrung linear–elastisch berechnet. Unter dieser Voraussetzung werden Kraftflusspfade hinreichend genau durch ganz bestimmte Hauptspannungstrajektorien beschrieben. Abbildung 3.15 zeigt ein Feld von Hauptspannungswerten, in welchem Kraftflusspfade überlagert sind. Die Ähnlichkeit der Kraftflusspfade mit den Hauptspannungstrajektorien in deren Bereich ist offensichtlich.

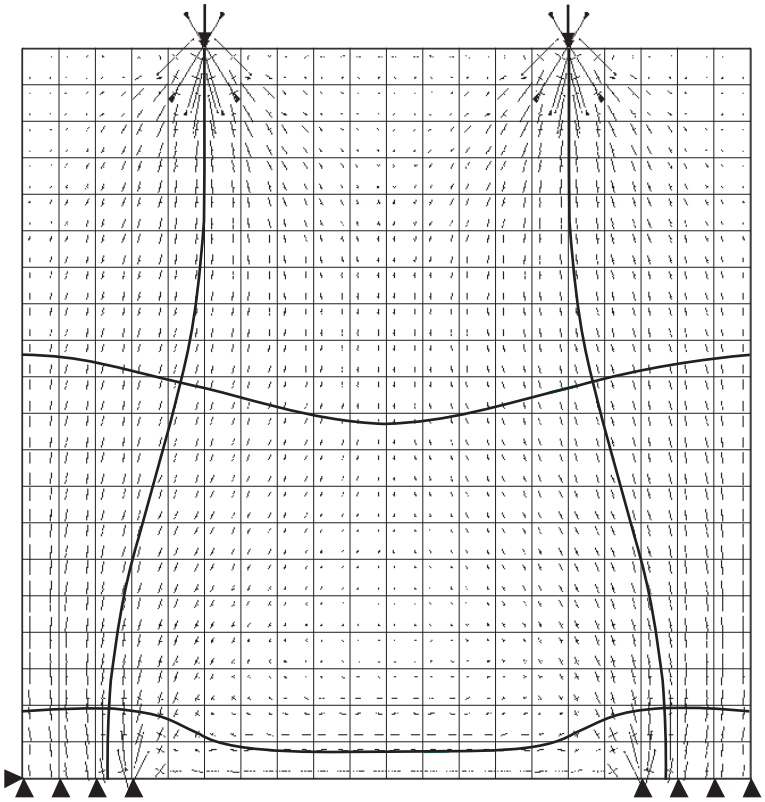


Abbildung 3.15: Vektorfeld mit Hauptspannungen und Trajektorien

Ein Kriterium zur Beschreibung von Kraftflusspfaden kann aus der Gleichgewichtsbedingung des zu generierenden Stabwerkes abgeleitet werden. Laut Definition

zeigen Kraftflusspfade den Verlauf der Lastabtragung von dessen Ursprung bis hin zum Auflager. Diese werden im Weiteren als primäre Kraftflusspfade bezeichnet. Die dazu orthogonalen Pfade werden zur besseren Unterscheidung sekundäre Kraftflusspfade genannt. Das Stabwerkmodell orientiert sich an diesen Pfaden, womit später ein Gleichgewichtszustand mit den äußeren Lasten sichergestellt werden kann.

Damit wird aus dem beliebig feinen Netz von Hauptspannungstrajektorien diejenigen herausfiltert, die den Kraftfluss am besten beschreiben. Sie dienen dann als Ausgangspunkt für das Stabwerkmodell.

Die Anzahl der primären Kraftflusspfade ist abhängig von der Belastung. In Kapitel 6 finden sich hierzu weitere Überlegungen.

3.4.3 Berechnungsablauf

Ausgangspunkt für die Kraftflussberechnung ist ein Finite–Elemente–Modell eines Systems und dessen Belastung. In einer geometrisch sowie materiell linearen Berechnung werden zunächst die Spannungen des Systems mit Hilfe der Finiten–Elemente–Methode ermittelt (siehe dazu Kapitel 2). Daraus lassen sich punktuell Hauptspannungen ermitteln und in einem Feld von Hauptspannungsrichtungen darstellen, welche schon einen ersten Eindruck von der Lastabtragung geben (vgl. Abbildung 3.15).

Mit dessen Hilfe lassen sich zwei orthogonale Trajektorienscharen mit beliebiger Diskretisierung erzeugen. Daraus werden die Trajektorien ausgewählt, die den Kraftflusspfaden ähneln und schließlich zur Bildung eines Stabwerkes beitragen.

Mit der Stabwerksanalogie wird versucht, die Lastabtragung in einem homogenen Bauteil allein durch Normalkraft diskreter Fachwerkstäbe zu beschreiben. Dabei orientieren sich die Stäbe an der Richtung der Trajektorien, und die Knotenpunkte an deren Schnittpunkten. Das so entstandene Stabwerk wird unter Berücksichtigung der statischen und kinematischen Randbedingungen ins Gleichgewicht gebracht, indem freie Knoten verschoben werden.

Die Modellierung der Steifigkeitsverhältnisse ist durch Einhaltung der Kompatibilität in den Verformungen der Scheibe und des Stabwerkes gelöst. Die aus der anschließenden Berechnung resultierenden Normalkräfte geben einen Eindruck von der Beanspruchung des Systems. Unter Berücksichtigung der zugehörigen Querschnittsfläche im Bauteil ist eine realistische Dimensionierung bzw. Bewehrung möglich.

Wegen der hohen innerlichen statischen Unbestimmtheit gibt es jedoch keine eindeutige Lösung. Das Tragverhalten kann damit auf unterschiedliche Weise verstanden werden. Wird die Bewehrungsführung an das Modell angepasst, kann dadurch

das Tragverhalten des realen Bauteils derart gesteuert werden, dass es tatsächlich wie das zugrunde liegende Stabwerk abträgt.

Durch die beschriebene Methode wird ein wahrscheinliches, optimertes Stabwerkmodell generiert, dessen Lösung vom Grad der Diskretisierung abhängig ist.

3.4.4 Ein Algorithmus zur Ermittlung von Trajektorien (ALFIT)

Der entwickelte und im Weiteren beschriebene Algorithmus kann prinzipiell für die Bestimmung einer Trajektorie jeder beliebigen Feldgröße verwendet werden. Hier wird er für die Hauptspannungen eingesetzt.

Grundvoraussetzung dafür ist das Vorhandensein eines Spannungsfeldes. Die eigentliche Trajektorie wird dann durch einen Polygonzug approximiert, dessen Startpunkt und Initialrichtung durch die Last definiert ist. Mit einer adaptiven Schrittweite wird die Lage des nächsten Punktes des Polygons ermittelt. Eine Spannungsauswertung liefert dort die Richtung und die Schrittweite für die Ermittlung des nächsten Polygonpunktes (vgl. Abschnitt 3.4.5). Abbruchkriterium für den Polygonzug ist das Erreichen einer Systemberandung.

Startrichtung einer Trajektorie

Gerade an der Lasteinleitung, also dem Startpunkt für die Trajektorie, ist die eindeutige Zuordnung allein aus den Spannungswerten besonders schwer. Nicht selten treten dort Singularitäten in den Spannungen infolge von Diskontinuitäten auf (s.h. Abbildung 3.16).

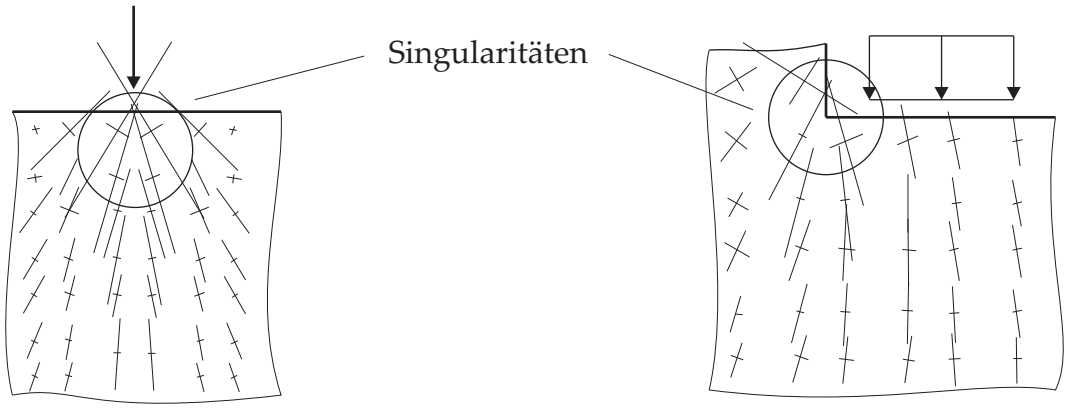


Abbildung 3.16: Singularitäten an Diskontinuitätsstellen

Zudem sind am Strukturrand in der Regel die Spannungsrandbedingungen infolge von Diskretisierungsfehler der Finite–Elemente–Methode verletzt. Aufgrund dessen kann die Ausrichtung der Hauptspannungen am Rand in der Berechnung deutlich von der eigentlichen Richtung im tatsächlichen Tragwerk abweichen.

Im Gegensatz dazu gibt die Neumann–Randbedingung, auch Kraftrandbedingung genannt, als Quelle einer Trajektorie deren Initialrichtung zuverlässig vor. Die Ausrichtung der Kraft ist klar definiert und zeigt direkt am Rand in Richtung der größeren Hauptspannung.

Ermittlung der Hauptspannungen

Im weiteren Verlauf wird die in Kapitel 2 angesprochene linear–elastische Finite–Elemente–Berechnung für den ebenen Spannungszustand zu Grunde gelegt. Die an den Gaußpunkten ausgewerteten Spannungen in X- und Y–Richtung sind auf die jeweiligen Elementecken extrapoliert. Zur Bestimmung der Spannungen im System aus bekannten globalen Koordinaten siehe Abschnitt 3.4.5.

Aus den globalen Spannungen werden die Hauptspannungen σ_1 und σ_2 sowie deren Ausrichtung α wie folgt berechnet (vgl. Abbildung 3.17).

$$\sigma_{1,2} = \frac{\sigma_x + \sigma_y}{2} \pm \frac{1}{2}\sqrt{(\sigma_x - \sigma_y)^2 + 4\tau_{xy}^2} \tag{3.6}$$

$$\tan 2\alpha = \frac{2\tau_{xy}}{\sigma_x - \sigma_y} \tag{3.7}$$

Nach dieser Definition ist $\sigma_1 > \sigma_2$. Der Winkel α ist dabei die Verdrehung des orthogonalen Hauptspannungspaares gegenüber der globalen X-Achse im mathematisch positiven Sinn. Eine eindeutige Zuordnung der Spannungswerte zu ihren Richtungen ist daraus nicht möglich. Bei der Ermittlung der Trajektorie ist eine zusätzliche Information notwendig.

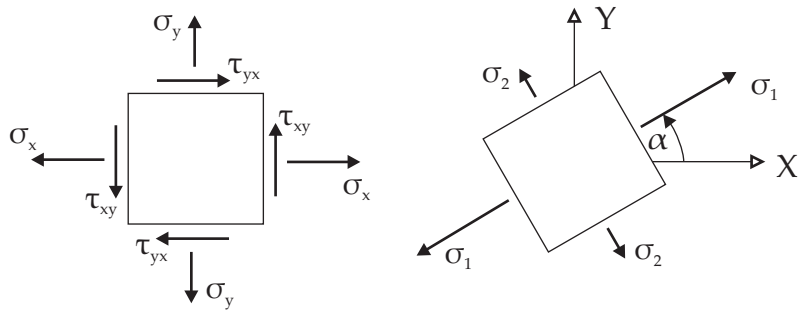


Abbildung 3.17: Globale Spannungen und Hauptspannungen

Bestimmung der Suchrichtung

Im Verlauf einer Trajektorie ändert sich nicht nur deren Richtung, sondern auch der Betrag der zugehörigen Hauptspannung. Somit ist nicht sicher gestellt, dass zu

einer Trajektorie immer dieselbe Hauptspannungskomponente σ_1 bzw. σ_2 gehört. Zudem hat jede Hauptspannung zwei um 180° gedrehte Ausrichtungen.

Ausgehend von den gegenüber den Globalen Koordinatenachsen um den Winkel α gedrehten Hauptspannungen gibt es also vier potentielle neue Richtungen (vgl. Abbildung 3.18). Der Polygonzug wird in der Darstellung von links nach rechts gebildet.

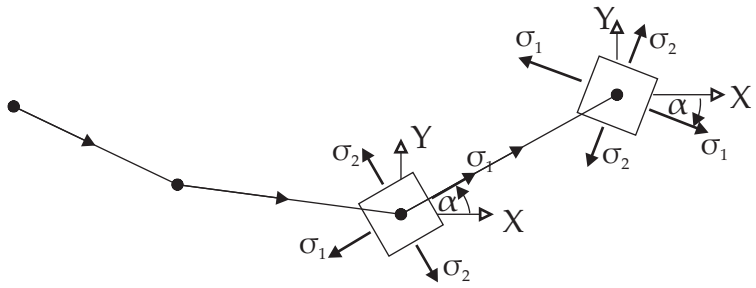


Abbildung 3.18: Das Problem der Suchrichtung bei der Bestimmung der Trajektorie

An Stellen mit großen Richtungsänderungen von einem Schritt zum nächsten führt die Mehrdeutigkeit der Lösung eventuell auf eine falsche Entscheidung. Zur Bestimmung der korrekten Suchrichtung für den nächsten Punkt auf der Trajektorie werden neben α zusätzlich die Richtungsänderung und die Änderung der Hauptspannungswerte σ_1 und σ_2 beobachtet. Die Wahl fällt auf die Richtung mit der geringsten Abweichung gegenüber der Trajektorienrichtung des vorigen Punktes. Die Spannungsänderung ist dabei ein Maß für die Schrittweite, die gegebenenfalls angepasst wird.

Festlegung der Schrittweite

Mehrere Gesichtspunkte fließen in die Bestimmung der Schrittweite für die Ermittlung des nächsten Polygonpunktes mit ein. Sie wird umso kleiner gewählt, je größer die Spannungs- oder Richtungsänderung der letzten beiden Schritte ist, um das sogenannte „Abdriften“ zu vermeiden. Abbildung 3.19 zeigt die Abweichung des beschreibenden Polygons von der exakten Trajektorie, wenn die Schrittweite s zu groß gewählt wird. Die Fehler summieren sich auf, ohne jegliche Korrekturmöglichkeit. Verlaufen die Trajektorien indes eher geradlinig, sind auch keine größeren Spannungsänderungen zu erwarten. Der Polygonzug kann dann auch durch größere Schritte die Trajektorie genau genug beschreiben, wie in Abbildung 3.19 unten zu sehen ist.

Als Maß für die Schrittweite s dient ein Bruchteil der Abmessung des Elementes in welchem sich der aktuelle Polygonpunkt gerade befindet. In Tragwerksbereichen mit großen Spannungsänderungen auf kleinem Raum wird die Schrittweite

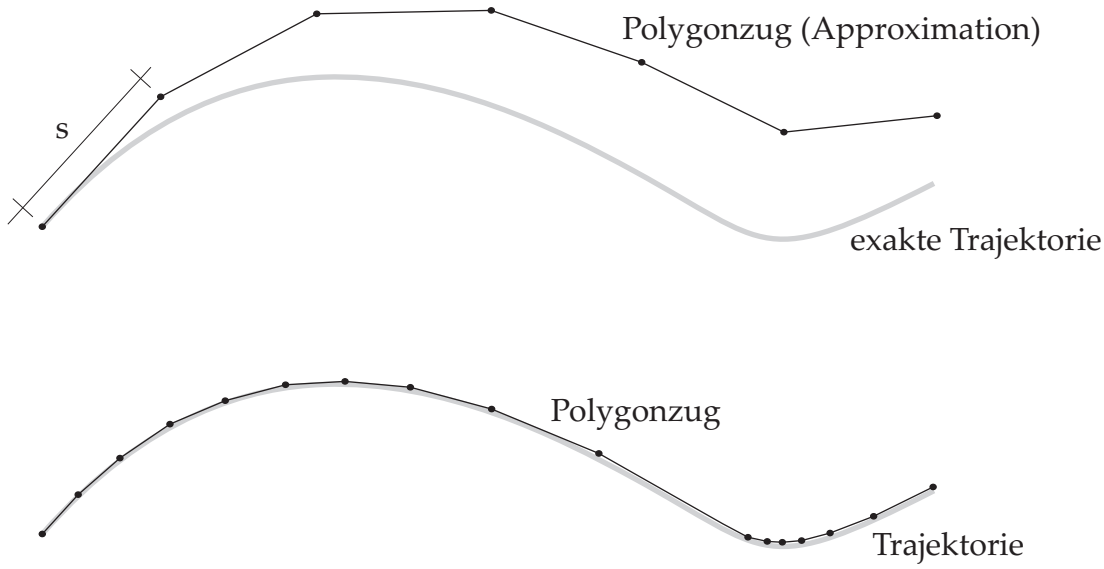


Abbildung 3.19: Gefahr des Abdriftens des beschreibenden Polygons von der tatsächlichen Trajektorie bei falscher Schrittweite

automatisch herabgesetzt. Die Verfeinerung des Finite–Elemente–Netzes an kritischen Orten verstärkt diesen Effekt. Dadurch ist der Suchalgorithmus in der Lage, auch stärkere Krümmungen der Trajektorien genau genug abzubilden. Je kleiner die gewählte Schrittweite ist, desto geringer ist die Abweichung von der theoretisch exakten Trajektorie. Die adaptive Schrittweite sorgt für minimale Rechenzeiten.

Mit einem Faktor r besteht zudem die Möglichkeit interaktiv auf die Schrittweite Einfluss zu nehmen. Parameterstudien an gut modellierten Systemen haben gezeigt, dass sich bei Schrittweiten $s < 0,1 \cdot e$ keine sichtbaren Änderungen der beschreibenden Polygone mehr einstellt. Mit e als mittlere Abmessung des betrachteten Elements (vgl. Abbildung 3.21 rechts) nach Gleichung 3.8 ist der Bezug zum aktuellen Punkt auf dem Polygon gewährleistet und auch einem unregelmäßigen Finite–Elemente–Netz Rechnung getragen.

$$e = \frac{1}{4} \cdot (a + b + c + d) \tag{3.8}$$

Ein derart entstandener, adaptiver Polygonzug beschreibt die tatsächliche Trajektorie damit genügend genau. Bei sehr geringen Richtungsänderungen einer Trajektorie, sollte die Schrittweite des Polygons auch nach oben hin begrenzt werden. Das Maß für die Schrittweite s wird aufgrund obiger Überlegungen daher in Gleichung 3.9 festgelegt. Vergleichsrechnungen haben dessen Praktikabilität bewiesen.

$$s = \frac{r \cdot e}{\left|\frac{\Delta\sigma}{\sigma_m}\right| + cos(\Delta\alpha) + 0,5} \tag{3.9}$$

Der erste Summand im Nenner bezieht sich auf die absolute Änderung der Spannung $\Delta\sigma = \left|\sigma^i - \sigma^{i-1}\right|$ vom letzten Polygonpunkt zum aktuellen Ort. Bezogen auf den Betrag des Mittelwertes $\sigma_m = 0,5\left|\cdot(\sigma^i + \sigma^{i-1})\right|$ ist eine von Absolutwerten unabhängige Größe gegeben. Der Wert der Spannung σ^i kann durch ein vielfaches β der Spannung σ^{i-1} ausgedrückt werden. Die Grenzübergänge für $\beta \to 0$ ($\Delta\sigma = \sigma^{i-1}$), $\beta \to 1$ ($\Delta\sigma = 0$) und $\beta \to \infty$ ($\Delta\sigma \to \infty$) liefern den Bereich in dem sich der erste Summand bewegen kann.

$$\left|\frac{\Delta\sigma}{\sigma_m}\right| = \left|\frac{\sigma^i - \sigma^{i-1}}{0,5\cdot(\sigma^i + \sigma^{i-1})}\right| = \left|\frac{\beta\cdot\sigma^{i-1} - \sigma^{i-1}}{0,5\cdot(\beta\cdot\sigma^{i-1} + \sigma^{i-1})}\right| = \left|\frac{\beta - 1}{0,5\cdot(\beta + 1)}\right| \tag{3.10}$$

$$\lim_{\beta\to 0}\left|\frac{\Delta\sigma}{\sigma_m}\right| = 2\,; \qquad \lim_{\beta\to 1}\left|\frac{\Delta\sigma}{\sigma_m}\right| = 0\,; \qquad \lim_{\beta\to\infty}\left|\frac{\Delta\sigma}{\sigma_m}\right| = 2 \tag{3.11}$$

Ist mit $\Delta\sigma$ eine große Differenz in den Spannungen gegeben, wird die Schrittweite s zur Bestimmung eines neuen Polygonpunktes herabgesetzt. Der Ausdruck $\left|\frac{\Delta\sigma}{\sigma_m}\right|$ bewegt sich im Intervall $[0;2]$.

Genauso wird durch eine größere Richtungsänderung $\Delta\alpha$ zwischen dem letzten und dem aktuellen Schritt als Maß für die Krümmung der Spannungstrajektorie die Schrittweite angepasst. Mit $-45^\circ \leq \alpha \leq +45^\circ$ ist das Resultat von $cos(\Delta\alpha)$ auf den Bereich $\left[0;0,5\sqrt{2}\right]$ beschränkt.

Der feste Wert von $0,5$ im Nenner stellt die Obergrenze der Schrittweite auf $2\cdot r\cdot e$ sicher, falls weder eine Richtungsänderung $(\Delta\alpha) = 0$, noch eine Spannungsänderung $(\Delta\sigma) = 0$ vom letzten zum aktuellen Berechnungsschritt auftritt. Damit ist die Schrittweite s auch nach unten beschränkt und kann Werte im folgenden Bereich einnehmen.

$$s \in \left[(2 + 0,5\sqrt{2} + 0,5)^{-1}\cdot r\cdot e\,;\; 2\cdot r\cdot e\right] = [0,312\cdot r\cdot e\,;\; 2\cdot r\cdot e] \tag{3.12}$$

Parameterstudien mit größeren Schrittweiten haben gezeigt, dass die Trajektorien, welche mit unterschiedlichen Schrittweiten berechnet wurden, geringfügig voneinander abweichen. Diese Abweichung ist umso kleiner, je homogener der Spannungszustand ist. Bei komplexen Spannungszuständen ist es demnach ratsam, die Schrittweite möglichst klein zu wählen. Dem Anwender wird deswegen zusätzlich die Möglichkeit eingeräumt, interaktiv mit dem Vorfaktor r auf die Schrittweite Einfluss zu nehmen.

Bestimmung der globalen Koordinaten des neuen Polygonpunktes

Ist die Richtung und die Schrittweite für einen weiteren Punkt P^{i+1} auf dem Polygonzug bekannt, werden dessen globale Koordinaten in der XY–Ebene mit den Gleichungen 3.13 und 3.14 aus den Koordinaten des letzten Punktes P^i berechnet (vgl. Abbildung 3.20).

$$x^{i+1} = x^i + s \cdot cos\alpha \tag{3.13}$$

$$y^{i+1} = y^i + s \cdot sin\alpha \tag{3.14}$$

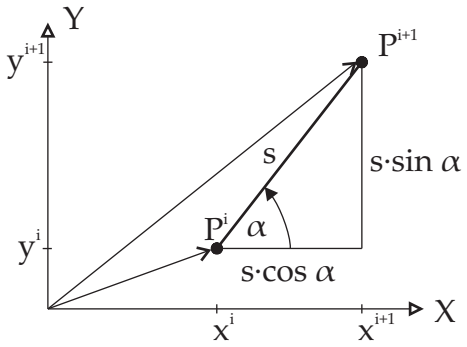


Abbildung 3.20: Bestimmung eines neuen Trajektorienpunktes

Auf diese Weise werden weitere Polygonpunkte auf der Struktur ermittelt, bis ein Bauteilrand erreicht ist. Sollte ein weiterer Punkt nicht mehr innerhalb eines Elementes der Struktur zu liegen kommen, wird der letzte Punkt als Schnittpunkt mit dem Strukturrand bestimmt.

Der Vorgang wiederholt sich für alle definierten Startpunkte, also alle Lasteinleitungsstellen. Die durch die Polygonzüge approximierten Trajektorien beschreiben annähernd den Weg des Kraftflusses von einer Krafteinleitungsstelle durch die Struktur bis zum Auflager. Aus diesem Grund werden sie hier als primäre Kraftflusspfade bezeichnet.

3.4.5 Zur Spannungsermittlung an beliebigen Punkten

Die Ermittlung der Spannungen an einem beliebigen Punkt innerhalb eines Systems stellt gerade bei unregelmäßigen und/oder stark verzerrten Elementnetzen eine besondere Herausforderung dar, wenn lediglich die globalen Koordinaten des Punktes bekannt sind.

Zunächst muss das Element ermittelt werden, in welchem der Punkt liegt, sowie dessen Lage dort in natürlichen Koordinaten. Elementbezogene globale Spannungskomponenten an den Elementknoten sind dann der Ausgangspunkt für die hier verwendete bilineare Interpolation.

Bestimmung des Elementes

Für die klare Zuordnung eines Punktes zu einem Element eines Finite–Elemente–Netzes sind zwei Schritte notwendig. Zunächst wird der euklidische Abstand d_i vom betrachteten Punkt P zu allen Systemknoten N_i ermittelt und der nächstliegende bestimmt. Abbildung 3.21 zeigt den betreffenden Systemknoten N in einem Ausschnitt eines Finite–Elemente–Netzes. Der Punkt P liegt demzufolge in einem der vier Elemente, welche am gemeinsamen Systemknoten N angrenzen.

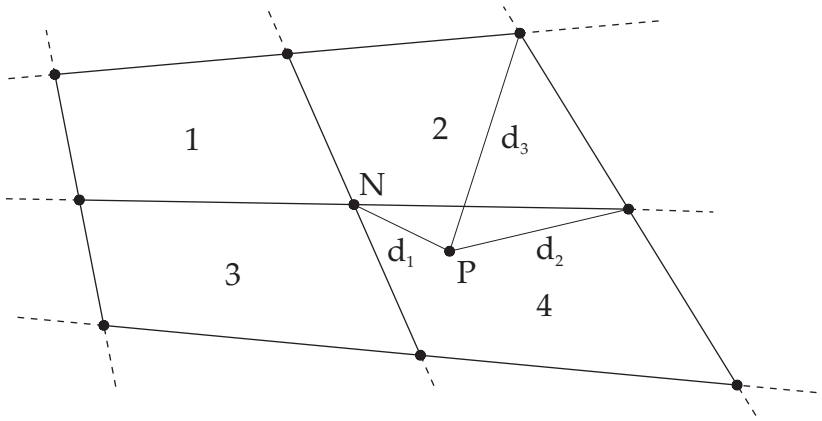


Abbildung 3.21: Bestimmung des kleinsten Abstandes von P zu einem Systemknoten

In einem zweiten Schritt wird das betreffende Element durch ein einfaches geometrisches Kriterium gefunden. Ist die Summe der „Innenwinkel“ α_i gleich 360° liegt P im Element, wie aus Abbildung 3.22 rechts zu erkennen ist. Bei einer Winkelsumme von weniger als 360° befindet sich P außerhalb des Elements (vgl. Abbildung 3.22 links). Die Definition der „Innenwinkel“ ist den Bildern zu entnehmen.

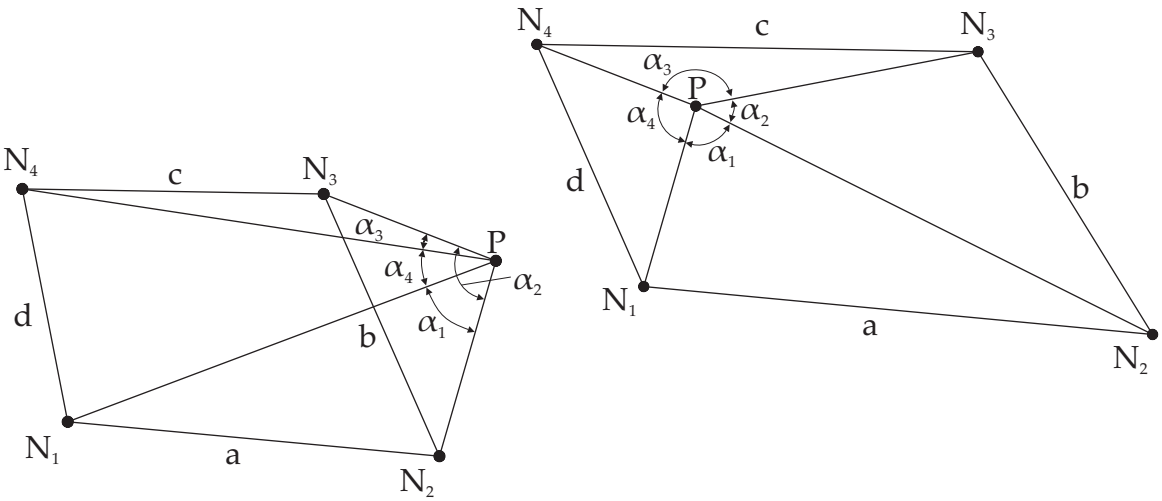


Abbildung 3.22: Der Systemknoten N ist Punkt P am nächsten

Ermittlung der natürlichen Koordinaten des Punktes

Zur Bestimmung von Trajektorien sind Spannungsinformationen an beliebigen Punkten innerhalb eines Systems gesucht. Bei dem verwendeten isoparametrischen Element ist ein bilinearer Ansatz für die Formfunktionen in natürlichen Koordinaten zu Grunde gelegt. Durch Interpolation der Knotenwerte mit den Formfunktionen lassen sich alle Werte im Elementinneren ausdrücken.

Werden ausschließlich rechtwinklige Elemente verwendet, können die elementbezogenen natürlichen Koordinaten ξ_p und η_p eines Punktes $P(x_p, y_p)$, der durch seine globalen Koordinaten beschrieben ist, direkt bestimmt werden. Im Allgemeinen jedoch muss P, in natürlichen Koordinaten ausgedrückt, iterativ ermittelt werden. Grund dafür ist die Ortsabhängigkeit der Basisvektoren. Für diese Aufgabe wurde ein Algorithmus entwickelt, der die natürlichen Koordinaten von P in wenigen Iterationsschritten mit einer Genauigkeit im Promillebereich ermittelt.

Ausgehend von einem Punkt $P(x_p, y_p)$ in globalen Koordinaten sind seine natürlichen Koordinaten ξ_p und η_p gesucht. Weiter ist das Element und dessen Knotenkoordinaten gegeben, in welchem sich der betreffende Punkt befindet. Ein beliebiger Punkt $P^k(x^k, y^k)$ ist durch die bilinearen Formfunktionen und die Knotenkoordinaten des Elementes über die Ausdrücke 3.15 und 3.16 beschrieben.

$$x^k = \sum N_i(\xi^k, \eta^k) \cdot x_i \tag{3.15}$$

$$y^k = \sum N_i(\xi^k, \eta^k) \cdot y_i \tag{3.16}$$

Darin sind N_i mit $i = 1..4$ die vier bilinearen Formfunktionen.

$$N_1^e = \frac{1}{4}(1-\xi)(1-\eta) \tag{3.17}$$

$$N_2^e = \frac{1}{4}(1+\xi)(1-\eta) \tag{3.18}$$

$$N_3^e = \frac{1}{4}(1+\xi)(1+\eta) \tag{3.19}$$

$$N_4^e = \frac{1}{4}(1-\xi)(1+\eta) \tag{3.20}$$

Die Punkte $P(x_p, y_p)$ und $P^k(x^k, y^k)$ sind dann identisch, wenn ihr Abstand mit Δx^k und Δy^k gleich Null ist.

$$\Delta x^k = x^k - x_p \tag{3.21}$$

$$\Delta y^k = y^k - y_p \tag{3.22}$$

Als Startwert für die iterative Suche nach $P(\xi_p, \eta_p)$ in natürlichen Koordinaten wird $P^0(\xi = 0, \eta = 0)$ verwendet. Die natürlichen Koordinaten des Punktes P^{k+1} lassen sich damit aus den Koordinaten des Schrittes k und dem Inkrement $\Delta\xi^k$ bzw. $\Delta\eta^k$ berechnen.

$$\xi^{k+1} = \xi^k + \Delta\xi^k \tag{3.23}$$

$$\eta^{k+1} = \eta^k + \Delta\eta^k \tag{3.24}$$

Für die Bestimmung der Inkremente $\Delta\xi^k$ und $\Delta\eta^k$ wird die Bedingung $\Delta\widetilde{x} = 0$ und $\Delta\widetilde{y} = 0$ für die Annäherung von P^k an P komponentenweise in einer Taylorreihe entwickelt. In Abbildung 3.23 sind die Zusammenhänge für den ersten Iterationsschritt mit $k = 0$ nochmals aufgezeigt.

$$\Delta\widetilde{x} = x^k + \frac{\partial x}{\partial \xi} \cdot \Delta\xi^k + \frac{\partial x}{\partial \eta} \cdot \Delta\eta^k - x_p \tag{3.25}$$

$$\Delta\widetilde{y} = y^k + \frac{\partial y}{\partial \xi} \cdot \Delta\xi^k + \frac{\partial y}{\partial \eta} \cdot \Delta\eta^k - y_p \tag{3.26}$$

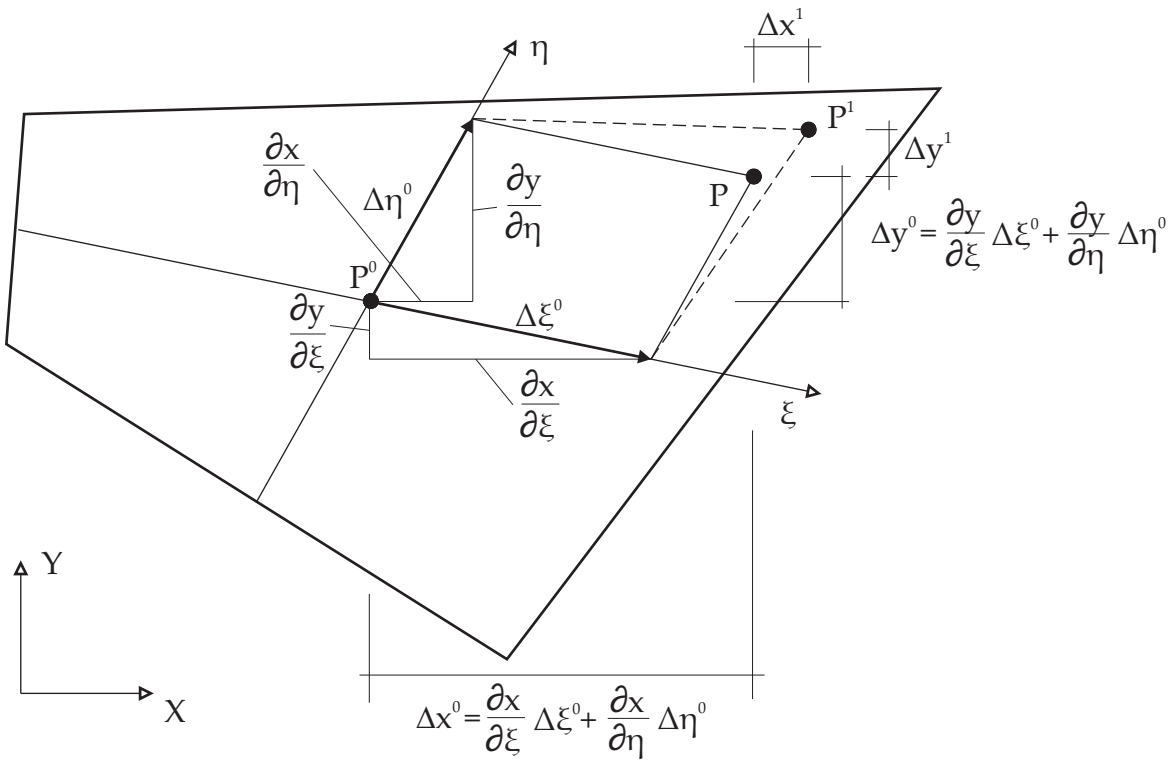


Abbildung 3.23: Erster Iterationsschritt zur Bestimmung der natürlichen Koordinaten von Punkt P

Zusammen mit den Gleichungen 3.21 und 3.22 können die Gleichungen 3.25 und 3.26 nach $\Delta\xi^k$ und $\Delta\eta^k$ aufgelöst werden.

$$\Delta\xi^k = \frac{\frac{\partial x}{\partial \eta} \cdot \Delta y^k - \frac{\partial y}{\partial \eta} \cdot \Delta x^k}{\frac{\partial x}{\partial \eta} \cdot \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi} \cdot \frac{\partial y}{\partial \eta}} \tag{3.27}$$

$$\Delta\eta^k = \frac{\frac{\partial x}{\partial \xi} \cdot \Delta y^k - \frac{\partial y}{\partial \xi} \cdot \Delta x^k}{\frac{\partial x}{\partial \xi} \cdot \frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta} \cdot \frac{\partial y}{\partial \xi}} \tag{3.28}$$

Als Abbruchkriterium für die inkrementelle Suche werden die Fehler Δx^k und Δy^k bzw. der euklidische Abstand $\epsilon = \sqrt{(\Delta x^k)^2 + (\Delta y^k)^2}$ vom Punkt P^{k-1} zu P^k betrachtet, und genügend klein gewählt. Damit sind die natürlichen Koordinaten von Punkt $P(\xi_p, \eta_p)$ bestimmt.

Interpolationsvorschrift

Da die Spannungswerte jedes Elementes an dessen Eckknoten bekannt sind, können sie nun auf den Punkt $P(\xi_p; \eta_p)$ mit der selben bilinearen Vorschrift (analog zu den Gleichungen 3.15 und 3.16) interpoliert werden:

$$\sigma_x = \sum N_i(\xi, \eta) \cdot \sigma_x^i \tag{3.29}$$

$$\sigma_y = \sum N_i(\xi, \eta) \cdot \sigma_y^i \tag{3.30}$$

$$\tau_{xy} = \sum N_i(\xi, \eta) \cdot \tau_{xy}^i \tag{3.31}$$

$$\sigma_z = \sum N_i(\xi, \eta) \cdot \sigma_z^i \tag{3.32}$$

3.4.6 Abbruchkriterien für Trajektorien

Der fortschreitende Suchalgorithmus zur Approximation einer Trajektorie durch einen Polygonzug hat drei Abbruchkriterien.

Gebietsberandung

Sobald der Polygonzug einen Systemrand erreicht und sich mit dem nächsten Punkt außerhalb des Gebiets befindet, wie Abbildung 3.24 zeigt, wird ein Schnittpunkt mit dem Rand gebildet. Die Methode basiert auf der Schnittpunktsermittlung zweier

Strecken, und wird in Abschnitt 3.4.8 bei der Suche nach dem Schnittpunkt zweier Trajektorien näher erläutert.

Für die Suche nach dem Schnittpunkt ist eine Strecke durch den letzten Polygonabschnitt $\overline{P^k P^{k+1}}$ gegeben. Bei kleinen Schrittweiten des Polygons, wie sie hier gegeben sind, ist die zweite Strecke die Elementkante $\overline{N_2 N_3}$ am Systemrand. Es ist jedoch nicht bekannt, welche Kante des Elementes auf dem Systemrand liegt. Deswegen werden Schnittpunkte von $\overline{P^k P^{k+1}}$ mit allen Elementrändern gesucht, was zu einem eindeutigen Ergebnis führt.

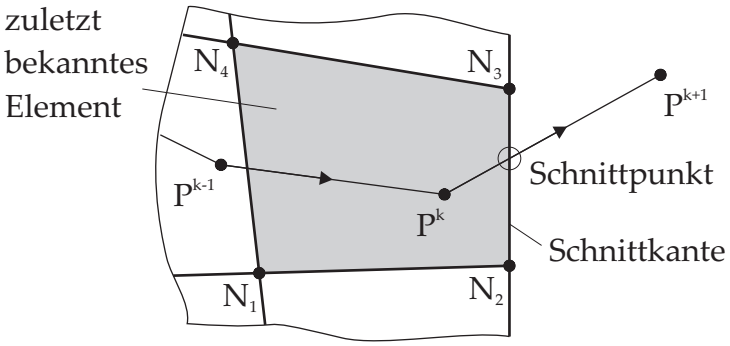


Abbildung 3.24: Ermittlung eines Schnittpunktes bei Erreichen des Systemrandes

Stagnation

Erreicht der Spannungszustand im Laufe der Trajektoriensuche den Wert Null, kann keine weitere Suchrichtung errechnet werden. Die Trajektorie stagniert an diesem Punkt. Es macht durchaus Sinn, Tragwerksbereiche bei der Kraftflussberechnung auszusparen, die nicht zur Lastabtragung beitragen.

Geschlossene Trajektorie

In einigen Sonderfällen zeigen sich geschlossene Trajektorien, wie Abbildung 3.25 verdeutlicht. In einem derartigen Fall muss deren Beschreibung durch ein Polygon an geeigneter Stelle abgebrochen werden, um eine Endlosschleife zu vermeiden. Hierfür können unterschiedliche Methoden herangezogen werden.

Eine ähnliche Problematik stellen Spiralen dar, die nahezu deckungsgleich aufeinander liegen, und augenscheinlich nicht von geschlossenen Trajektorien zu unterscheiden sind. Im Folgenden wird eine eher pragmatische Lösung umgesetzt, die eine Höchstanzahl an Polygonpunkten definiert und bei deren Erreichen die Suche abbricht.

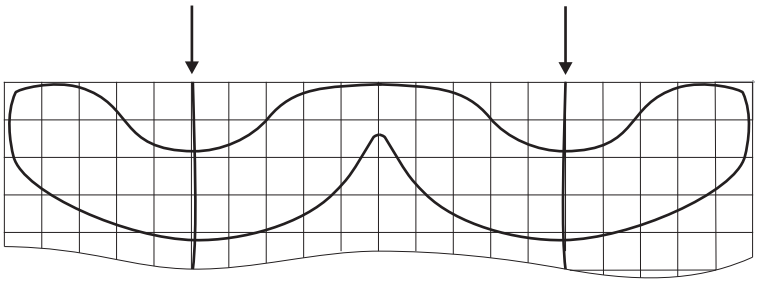


Abbildung 3.25: Geschlossene Trajektorie

3.4.7 Bestimmung von sekundären Kraftflusspfaden

Im Allgemeinen haben Trajektorien einen gekrümmten Verlauf, was rückschließend auch für die primären Kraftflusspfade gilt. Der Zusammenhang zwischen den Hauptspannungen und den Krümmungsradien der Flächen, auf denen die Spannungen senkrecht stehen, wird durch die Spannungsgleichung von Lamé für den allgemeinen Fall wie folgt beschrieben (vgl. Rückert [Rüc92]).

$$\frac{\partial \sigma_1}{\partial s_1} + \frac{\sigma_1 - \sigma_2}{\rho_{23}} + \frac{\sigma_1 - \sigma_3}{\rho_{32}} = 0 \tag{3.33}$$

$$\frac{\partial \sigma_2}{\partial s_2} + \frac{\sigma_2 - \sigma_3}{\rho_{31}} + \frac{\sigma_2 - \sigma_1}{\rho_{13}} = 0 \tag{3.34}$$

$$\frac{\partial \sigma_3}{\partial s_3} + \frac{\sigma_3 - \sigma_1}{\rho_{12}} + \frac{\sigma_3 - \sigma_2}{\rho_{21}} = 0 \tag{3.35}$$

Dabei sind s_i die Koordinaten eines krummlinigen Koordinatensystems entlang der orthogonalen Trajektorien, σ_{ij} sind die Hauptspannungen in Richtung der Trajektorien und ρ_{ij} die Krümmungsradien der Schnittflächen. Abbildung 3.26 zeigt die Zusammenhänge für den betrachteten ebenen Spannungszustand. Dabei lassen sich die Gleichungen 3.33 bis 3.35 wie folgt vereinfachen:

$$\sigma_3 = 0\,, \qquad \frac{\partial \sigma_3}{\partial s_3} = 0\,, \qquad \frac{1}{\rho_{12}} = \frac{1}{\rho_{21}} = 0\,, \qquad \frac{1}{\rho_{32}} = \frac{1}{\rho_{31}} = 0 \tag{3.36}$$

$$\frac{\partial \sigma_1}{\partial s_1} + \frac{\sigma_1 - \sigma_2}{\rho_2} = 0 \quad \text{, mit } \rho_2 = \rho_{23} \tag{3.37}$$

$$\frac{\partial \sigma_2}{\partial s_2} + \frac{\sigma_2 - \sigma_1}{\rho_1} = 0 \quad \text{, mit } \rho_1 = \rho_{13} \tag{3.38}$$

Eine Umformulierung der Gleichungen 3.37 und 3.38 lässt den Zusammenhang der Hauptspannungen und der zugehörigen Trajektorienkrümmungen erkennen.

$$\frac{\partial \sigma_1}{\partial s_1} \cdot \rho_2 + \frac{\partial \sigma_2}{\partial s_2} \cdot \rho_1 = 0 \tag{3.39}$$

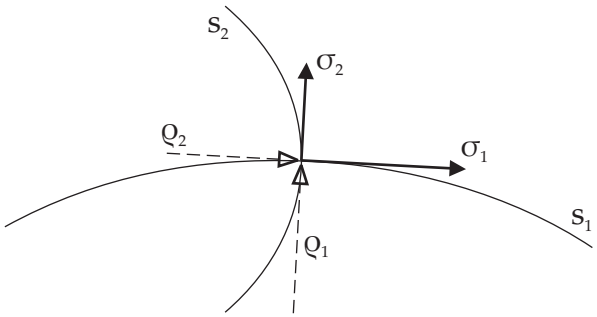


Abbildung 3.26: Hauptspannungen und Krümmungsradien der Trajektorien

Das Spannungsgefälle $\partial\sigma/\partial s$ entlang der Trajektorien verhält sich negativ umgekehrt proportional zu ihren Krümmungsradien. Anders ausgedrückt, muss an jeder Krümmung einer Trajektorie eine Kraft orthogonal dazu wirken, um Gleichgewicht zu erzeugen.

Der Kraftfluss im System wird zunächst durch die primären Kraftflusspfade dargestellt, die von der Lasteinleitung bis zum Auflager meist gekrümmt verlaufen. An diesen Krümmungen sind aufgrund des Ausdruckes 3.39 kontinuierliche Umlenkkräfte für die Einhaltung des Gleichgewichts erforderlich. Für eine diskrete Betrachtung sind alle wesentlichen Charakteristika des primären Kraftflusspfades (Trajektorie) zu berücksichtigen.

In Abbildung 3.27 ist oben die tatsächliche Trajektorie und der approximierte Polygonzug dargestellt. In Bildmitte sind die wesentlichen Bereiche der Kurve hervorgehoben, die zwei wesentlichen Bereiche maximaler Krümmung und der Wendepunkt dazwischen. Zudem geben die zwei Endpunkte die Lage der Trajektorie an. Im unteren Bild ist ein stark reduzierter Polygonzug abgebildet, der die wesentlichen Eckpunkte der Trajektorie widerspiegelt.

In obigem Beispiel werden Abschnitte mit der selben Krümmungsrichtung im Punkt maximaler Krümmung zusammengefasst. Der stark reduzierte charakteristische Polygonzug markiert Orte auf welche die Umlenkkräfte konzentriert sind. Das sind geeignete Orte für sekundäre Kraftflusspfade, die zur Einhaltung des Gleichgewichts notwendig sind (vgl. Abbildung 3.26).

Bestimmung des charakteristischen Polygonzugs

Bei der Charakterisierung der eigentlichen Trajektorie werden die Krümmungsinformationen des beschreibenden Polygons herangezogen. Abschnitte mit geringen

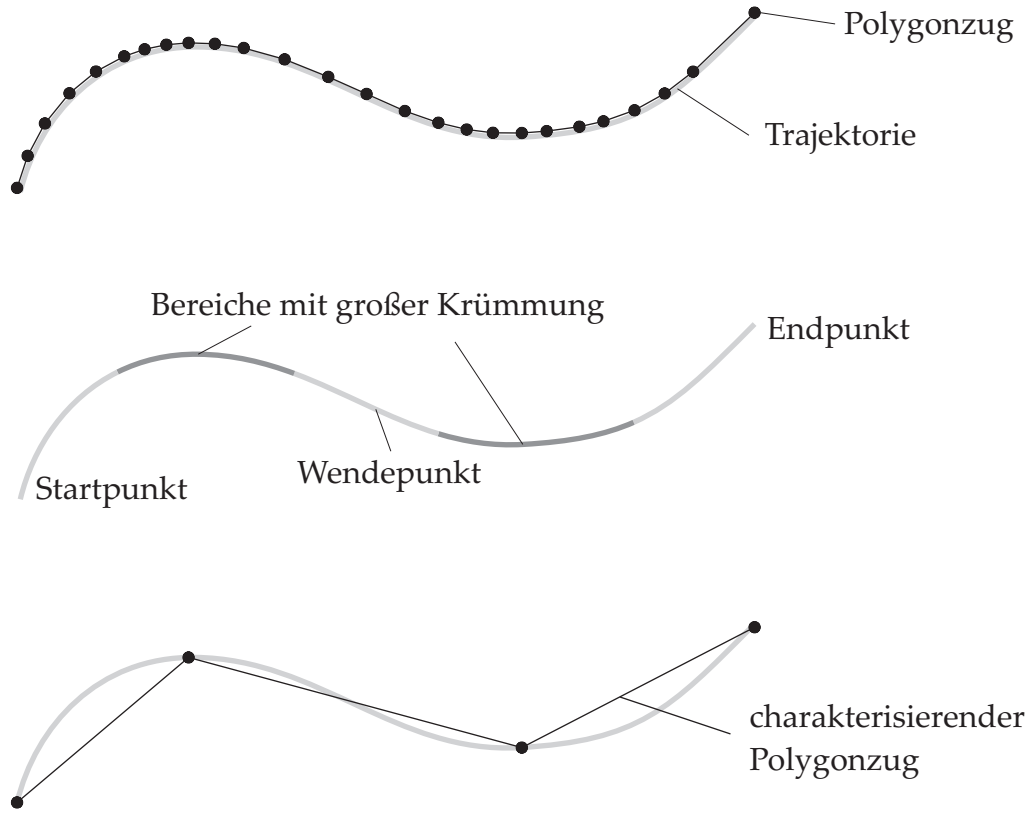


Abbildung 3.27: Erfassung der Charakteristik einer Kurve

Richtungsänderungen $\Delta\beta$, wie Abbildung 3.28 rechts zeigt, werden zusammengefasst und Mittelknoten eliminiert. Ist die Richtungsänderung $\Delta\alpha$ jedoch groß (Abbildung 3.28 links), bleibt der betreffende Abschnitt unverändert.

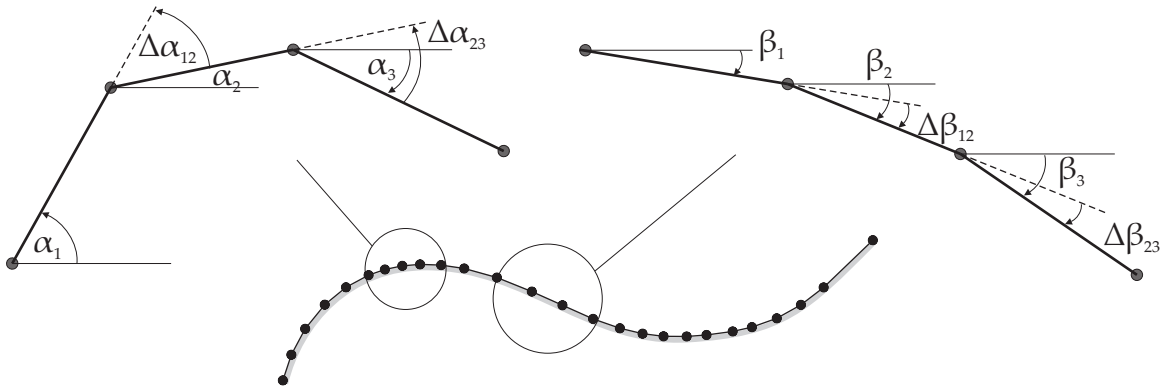


Abbildung 3.28: Bestimmung des charakteristischen Polygonzugs

Als Maß für die Krümmung der Trajektorie kann die Winkeländerung von paarweise aufeinander folgenden Linienabschnitten des Polygonzuges und deren Längen herangezogen werden. Anstelle dessen wird hier das Verhältnis des Stiches f zu seiner Grundlinie c verwendet (vgl. Abbildung 3.29), da die notwendigen Informationen bereits vorliegen.

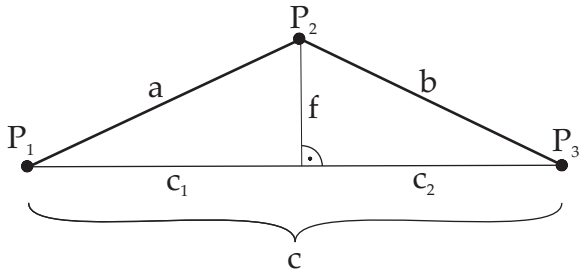


Abbildung 3.29: Die Beschreibung der „Krümmung“ des Polygons durch den Stich f

Gegeben sind die Knotenkoordinaten $P_1(x_1/y_1)$, $P_2(x_2/y_2)$ und $P_3(x_3/y_3)$ der Polygonabschnitte. Die Gleichungen 3.40 bis 3.43 beschreiben die in Abbildung 3.29 dargestellten Zusammenhänge.

$$a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.40}$$

$$b = \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} \tag{3.41}$$

$$c = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \tag{3.42}$$

$$f = \sqrt{0,5 \cdot (a^2 + b^2 - 0,5c^2) - (\frac{a^2 - b^2}{2c})^2} \tag{3.43}$$

Überschreitet das Verhältnis f/c einen interaktiv festlegbaren Wert, ist die Krümmung des Abschnittes signifikant und der betrachtete Knoten wird als charakteristischer Punkt auf dem zu generierenden Polygon definiert. Ist dies nicht der Fall, handelt es sich um einen wenig gekrümmten Abschnitt und der betrachtete Knoten kann eliminiert werden. Abbildung 3.30 zeigt, wie ein Polygonzug (links) in zwei Schritten reduziert wird und sich auf die wesentlichen Charakteristischen Punkte beschränkt (rechts). Darin sind die grau hinterlegten Punkte und damit auch die Abschnitte des Polygons entfernt worden.

Abbildung 3.30: Reduktion des Polygonzuges in zwei Schritten von links nach rechts

An dieser Stelle ist derzeit noch eine Eingriffsmöglichkeit durch den Anwender gegeben, um ein vernünftiges Kriterium für die Generierung des charakteristischen

Polygonzugs vorzunehmen. Zudem eröffnet sich damit auch die Möglichkeit von Parameterstudien mit unterschiedlicher Diskretisierung der Stabwerkmodelle.

Die Charakteristik der eigentlichen Trajektorien wird in den Bereichen größter Krümmungen durch wenige Polygonabschnitte beschrieben. Dabei sind die benötigten Umlenkkräfte auf die Knotenpunkte reduziert. Dort wird das Gleichgewicht durch sekundäre Kraftflusspfade hergestellt. Maßgebend für deren Initialisierung ist der charakteristische Polygonzug, der die meisten Punkte enthält. Da die zugrunde liegende Trajektorie die größten Krümmungen und/oder die meisten Krümmungsänderungen aufweist, müssen diese alle durch die sekundären Kraftflusspfade ins Gleichgewicht gebracht werden.

Damit sind Startpunkte festgelegt, von denen aus die Ermittlung der orthogonalen Trajektorien analog zu Abschnitt 3.4.4 erfolgt. Zur Veranschaulichung ist in Abbildung 3.31 der Ausschnitt zweier Trajektorien gezeigt, die durch ihre charakteristischen Polygone beschrieben sind. Die Startpunkte für die orthogonalen Trajektorien sind durch den rechten Polygonzug festgelegt.

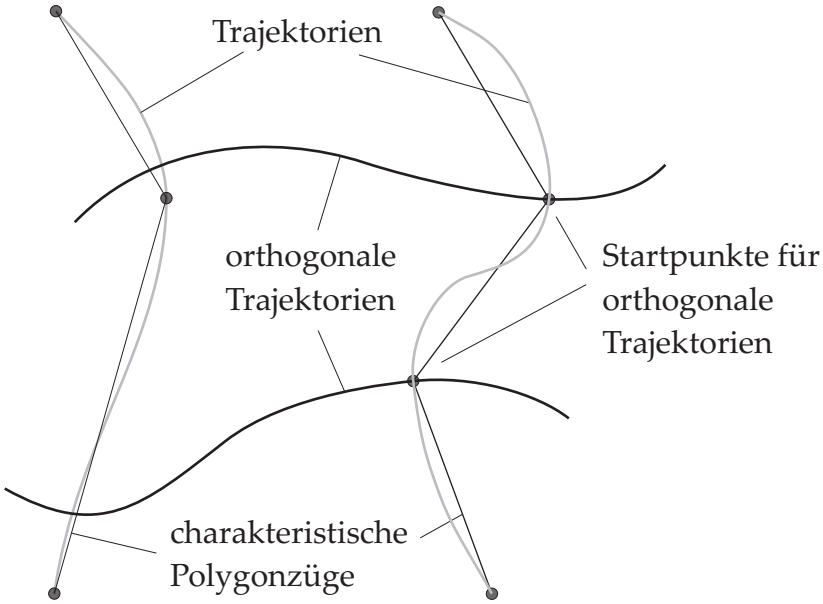


Abbildung 3.31: Startpunkte für die orthogonalen Trajektorien

Damit sind zwei speziell ausgewählte orthogonale Trajektorienscharen definiert, die als primäre und sekundäre Kraftflusspfade gedeutet werden. Sie spiegeln das Tragverhalten des Systems wider. Wie im Abschnitt 3.2 bereits erwähnt, wird daraus ein Stabwerkmodell generiert. Dessen Gleichgewichtszustand fasst den kontinuierlichen Kraftzustand im Tragwerk an diskreten Bereichen zusammen, die auf diskrete Bereiche zusammengefasst sind.

3.4.8 Bestimmung der Stabwerksgeometrie

Die Schnittpunkte der Kraftflusspfade bestimmen die Knotenpunkte des Stabwerkes, die Abschnitte dazwischen werden durch Fachwerkstäbe ersetzt. Für die Ermittlung der Knotenpunkte werden alle Streckenabschnitte der Polygonzüge mit denen der orthogonalen Polygone verglichen. Ist ein Schnittpunkt gefunden, wird dieser vermerkt und die nächste orthogonale Trajektorie untersucht.

Algorithmen zur Schnittpunktsuche von Strecken, welche mit Steigungen arbeiten, zeigen bei vertikalen Geraden eine Singularität. Werden die Geraden parametrisiert, so tritt die Singularität auf, wenn die Geraden um 45° gegen die horizontale geneigt sind. Da sowohl vertikale als auch um 45° geneigte Stäbe recht häufig auftreten, sind diese Methoden für die Implementierung schlecht geeignet.

Die Wahl zur Berechnung der Schnittpunkte fällt deswegen auf eine Methode, die mit Flächenverhältnissen arbeitet. Bei geschickter Implementierung werden numerische Probleme ausgeschlossen.

Ob zwei betrachtete Abschnitte einen Schnittpunkt besitzen und wie dieser ermittelt wird, ist in Abbildung 3.32 illustriert. Im linken Teil des Bildes ist der Fall zweier sich schneidender Strecken s_1 und s_2 dargestellt. Auf der rechten Seite haben die beiden Streckenabschnitte keinen gemeinsamen Punkt.

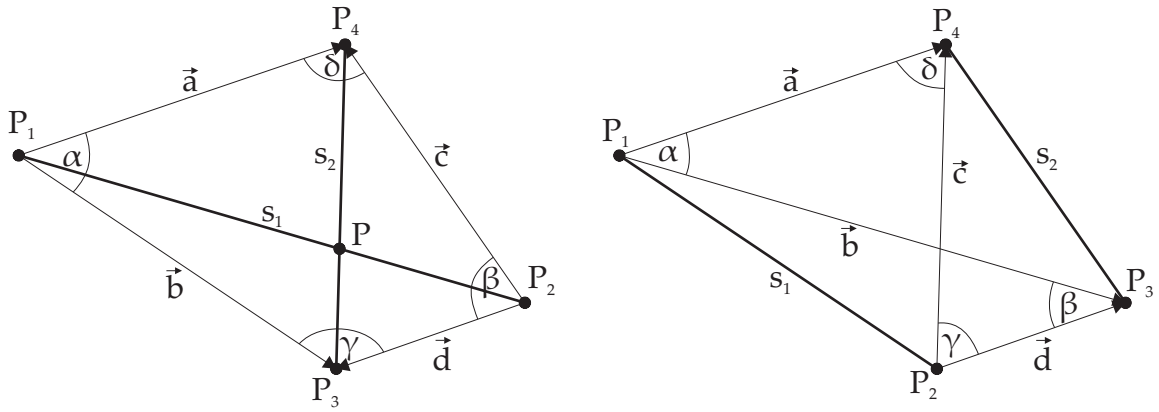


Abbildung 3.32: Schnittpunkt zweier Strecken

Für den Algorithmus werden zunächst vier Vektoren $\vec{a}$, $\vec{b}$, $\vec{c}$ und $\vec{d}$ durch ihre globalen Komponenten definiert. Die Vektoren zeigen jeweils von den beiden Endpunkten der Strecke s_1 zu den beiden Endpunkten der Strecke s_2.

$$a_x = x_4 - x_1, \quad a_y = y_4 - y_1 \tag{3.44}$$

$$b_x = x_3 - x_1, \quad b_y = y_3 - y_1 \tag{3.45}$$

$$c_x = x_4 - x_2, \quad c_y = y_4 - y_2 \tag{3.46}$$

$$d_x = x_3 - x_2, \quad d_y = y_3 - y_2 \tag{3.47}$$

Daraus lassen sich die vier Winkel α, β, γ und δ ermitteln, welche von jeweils zwei benachbarten Vektoren eingeschlossen werden. Schneiden sich die betrachteten Strecken s_1 und s_2 im Punkt P (vgl. Abbildung 3.32 links) beträgt die Summe aller vier Winkel 360° (Winkelsumme eines Vierecks). Es gibt jedoch keinen Schnittpunkt, wie Abbildung 3.32 rechts zeigt, wenn die Winkelsumme kleiner als 360° ist.

Der Punkt P teilt die Strecke $\overline{P_1P_2}$ im Verhältnis m, welches durch die gegebenen Flächen wie folgt ermittelt wird.

$$\frac{\overline{P_1P}}{\overline{P_1P_2}} = \frac{A_{134}}{A_{134} + A_{234}} = m \tag{3.48}$$

Hierbei ist A_{134} die Fläche des Dreiecks, welches durch die Punkte P_1, P_3 und P_4 definiert wird und durch das Vektorprodukt von $\vec{a}$ und $\vec{b}$ berechnet werden kann. Mit der Fläche A_{234} geschieht dies analog.

$$A_{134} = \frac{1}{2} \cdot \left| a_x b_y - a_y b_x \right| \tag{3.49}$$

$$A_{234} = \frac{1}{2} \cdot \left| c_x d_y - c_y d_x \right| \tag{3.50}$$

Die Koordinaten des Schnittpunktes P können dann z.B. mit den Endpunkten der Strecke s_1 und dem Verhältnis m bestimmt werden.

$$x_p = x_1 + m \cdot (x_2 - x_1) \tag{3.51}$$

$$y_p = y_1 + m \cdot (y_2 - y_1) \tag{3.52}$$

Der einzige Sonderfall ist in Abbildung 3.33 zu sehen. Der Schnittpunkt P ist bereits bekannt und Knotenpunkt von vier Polygonabschnitten. Damit ist $m = 1$, und $x_P = x_2$ und $y_P = y_2$.

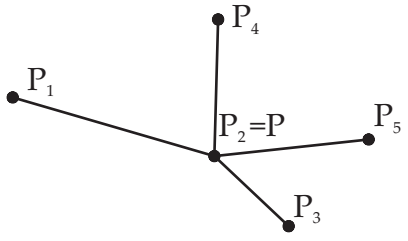


Abbildung 3.33: Sonderfall bei der Bestimmung des Schnittpunktes P

Sind alle Schnittpunkte der Kraftflusspfade gefunden und die Fachwerkstäbe dazwischen definiert, müssen die Randbedingungen des Problems auf das Stabwerkmodell übertragen werden. An statischen Randbedingungen sind die Stäbe in Richtung der Last zu orientieren, damit das Gleichgewicht eingehalten wird (s.h. Abbildung 3.34 rechts). Äquivalentes gilt für kinematische Randbedingungen. Bei verschieblichen Lagerungen kann nur in der unverschieblichen Richtung eine Kraftübertragung erfolgen. Auf der linken Seite der Abbildung 3.34 wird dieser Sachverhalt verdeutlicht.

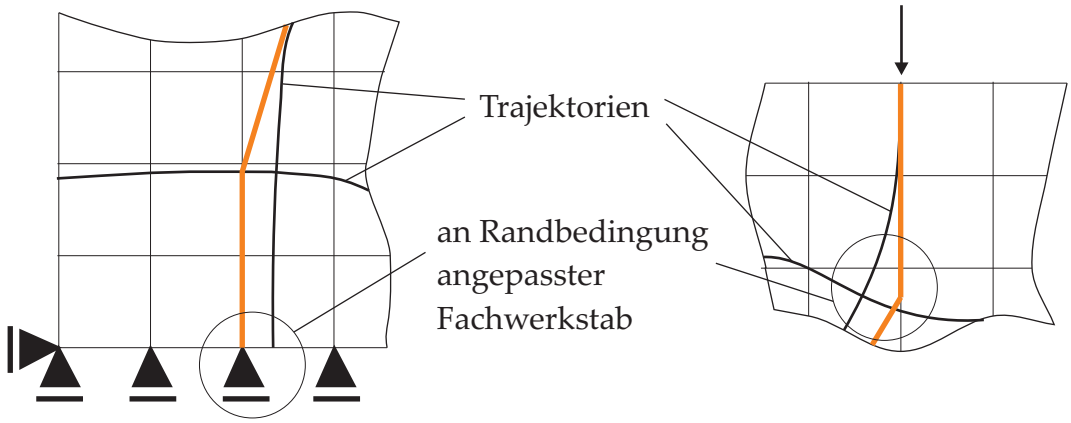


Abbildung 3.34: Anpassung des Fachwerks an die Randbedingungen des Problems

3.4.9 Berechnung des Stabwerkes

Eine zentrale Rolle bei der Kraftflussberechnung spielt die Ermittlung der Stabkräfte. Hierzu wird ein eigenes objektorientiertes Finite–Elemente–Programm zur Berechnung von Fachwerken verwendet (TRUSS).

Bei der Generierung von Stabwerkmodellen treten nicht selten kinematische Systeme auf, wie Abbildung 3.35 links andeutet. Eine konventionelle Berechnung nach der Finite–Elemente–Methode ist damit nicht ohne weiteres möglich. Das Gleichungssystem ist singulär und muss für die Berechnung stabilisiert werden. Dies erfolgt durch Einbau von Diagonalstäben, bis das Fachwerk mindestens statisch bestimmt ist.

Da sich die Aussteifungen nicht an der Lastabtragung beteiligen sollen, werden ihre Steifigkeiten nur einen Bruchteil der übrigen Stabsteifigkeiten ausmachen. Unter Berücksichtigung der Randbedingungen führt eine linear elastische Finite–Elemente–Berechnung mit Hilfe des Gleichgewichts auf die Stabkräfte. Im gezeigten Beispiel eines statisch bestimmten Stabwerkes geschieht dies unabhängig von den Steifigkeitsverhältnissen der Stäbe.

Vergleichsrechnungen haben hier gezeigt, dass die stabilisierenden Stäbe tatsächlich nur unwesentliche Kräfte abbekommen, welche durch minimale Abweichungen der

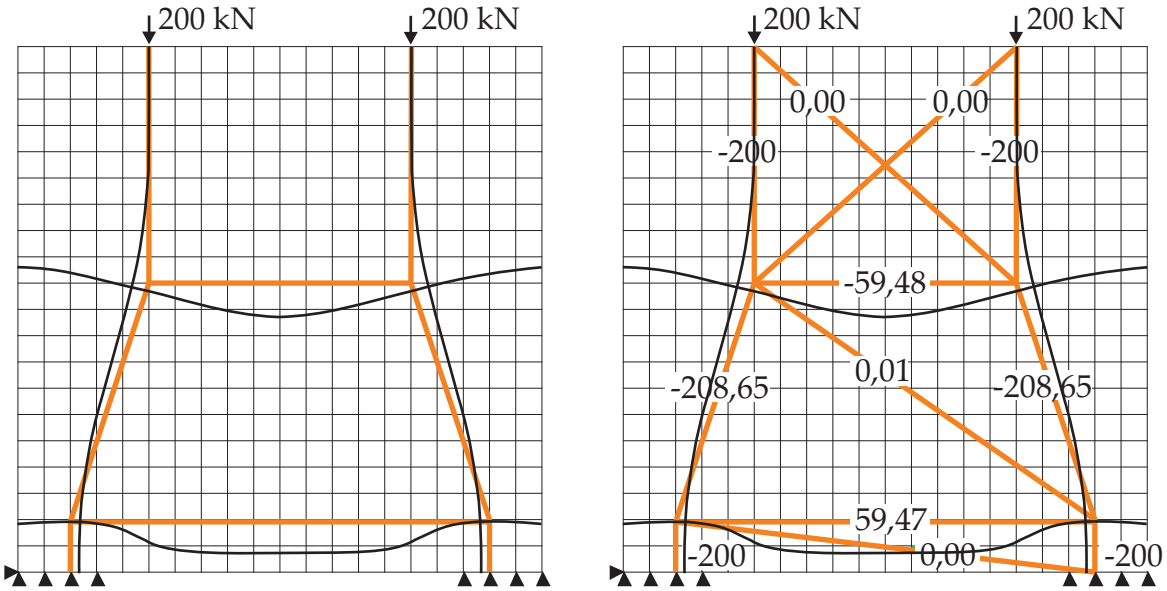


Abbildung 3.35: kinematisches Stabwerk und stabilisiertes Fachwerk

Geometrie von der idealen Gleichgewichtslage zu erklären sind. Durch Variation der Lage der Fachwerkknoten kann die Kraft der zusätzlichen aussteifenden Diagonalstäbe auf ein Minimum reduziert werden.

In allen behandelten Beispielen war dies jedoch nicht notwendig, da die ideale Gleichgewichtslage an sich schon nahezu exakt bestimmt war. Dieser Umstand ist dadurch zu erklären, dass sich das Stabwerk an den Kraftflusspfaden orientiert, welche die Last ausschließlich in ihrer Längsrichtung abträgt. Werden diese Kraftpfade nun durch Stäbe ersetzt, können diese die Last äquivalent durch Normalkräfte abtragen.

Zu bemerken ist jedoch, dass die Kompatibilitätsbedingung dabei verletzt wird, d.h. die Verformung wird nicht richtig abgebildet. Erst bei der Berechnung statisch unbestimmter Probleme wird die Bedeutung korrekt abgebildeter Verformungen ersichtlich. Schnittgrößen lassen sich hier nur mit Hilfe der tatsächlichen Verformungen berechnen, die nur mit den exakten Steifigkeitsverhältnissen der Stäbe zu ermitteln sind. Deren richtige Modellierung wird zwingend notwendig. Abschnitt 3.4.10 stellt sich dieser Herausforderung und beschreibt eine neue Lösungsstrategie.

3.4.10 Einhaltung der Kompatibilität durch Minimierung der Differenzarbeit

Bei der Kraftflussberechnung stellt sich die Herausforderung, die Tragwirkung und Kraftverteilung eines kontinuierlichen Systems auf einfache Weise zu beschreiben. Im vorliegenden Fall wird das Tragverhalten einer Scheibe durch ein diskretes Stabwerkmodell erfasst. Beispiele dafür werden im Kapitel 6 behandelt.

Wie im vorigen Abschnitt deutlich wurde, kann dies nur durch Kenntnis der richtigen Steifigkeitsverhältnisse der einzelnen Stäbe im Stabwerk gelingen, die jedoch zunächst nicht bekannt sind. Diese Problematik betrifft vor allem statisch unbestimmte Stabwerke, deren Schnittgrößen vom richtigen Verformungszustand und den dazugehörigen Ersatzsteifigkeiten abhängen. Das führt auf ein Optimierungsproblem. Ein allgemeiner Überblick über die mathematische Beschreibung und Lösungsalgorithmen findet sich bei Bletzinger [Ble90].

Der im Folgenden beschriebene Lösungsansatz dient zur Ermittlung der Steifigkeitsverhältnisse mit Hilfe der Kompatibilität der Verformungen des kontinuierlichen und diskreten Problems. Zugleich wird die Einhaltung des Gleichgewichts gefordert. Als Ausgangspunkt für die Ermittlung der korrekten Verteilung der Stabsteifigkeiten im Stabwerkmodell dient die linear elastische Finite–Elemente–Berechnung einer Scheibe, mit deren Verformungs- und Spannungszustand. Die daraus entstehenden Kraftflusspfade definieren zunächst eine optimierte Geometrie für das Stabwerkmodell. Die Verteilung der Steifigkeiten wird dann derart vorgenommen, dass sich die Knoten des Stabwerks deckungsgleich mit den betreffenden Punkten der Scheibe verformen. Damit wird der Verformungszustand der Scheibe zumindest punktuell durch das Stabwerkmodell korrekt abgebildet. Die diskreten Stäbe ersetzen damit die betreffenden Tragwerksbereiche der Scheibe durch eine adäquate Steifigkeit und beschreiben deren Tragverhalten. Die Stabkräfte spiegeln den tatsächlichen Spannungszustand in der Scheibe wieder. Im gerissenen Zustand von Stahlbetontragwerken konkretisiert sich das im Zugbereich auf die Bewehrung. Die zur Stabilisierung des Gleichungssystems hinzugefügten Diagonalstäbe werden bei der Steifigkeitsoptimierung (sizing) ausgenommen. Ihr Anteil an der Gesamtsteifigkeit muss entsprechend angepasst werden.

Als Kriterium für die korrekte Verteilung der Steifigkeiten wird das Minimum der Arbeit der Verschiebungsdifferenzen gefordert, die im Weiteren als Differenzarbeit bezeichnet wird. Hierbei werden die unterschiedlichen Knotenverformungen des diskreten Stabwerkes und der betreffenden Punkte der kontinuierlichen Scheibe betrachtet. Zusammen mit der positiv definiten Steifigkeitsmatrix entsteht mit Gleichung 3.53 ein positiver Arbeitsausdruck.

$$f = \frac{1}{2}\,\Delta\mathbf{u}^T\,\mathbf{K}\,\Delta\mathbf{u} \rightarrow min. \tag{3.53}$$

Mit $\Delta\mathbf{u} = \mathbf{u} - \hat{\mathbf{u}}$ ist die Differenz der Verschiebungen der Stabwerkknoten $\mathbf{u}$ und der Verschiebungen der jeweiligen Punkte der Scheibe $\hat{\mathbf{u}}$ beschrieben. Der Vektor $\hat{\mathbf{u}}$ ist durch die Scheibenberechnung vorgegeben, während $\mathbf{u}$ den Deformationszustand des Stabwerks für die aktuelle Steifigkeitsverteilung angibt. Zur Einhaltung der Kompatibilität ist die Differenz $\Delta\mathbf{u}$ an jedem Knoten zu minimieren. Für die bestehende Optimierungsaufgabe wird die Einhaltung der Stationaritätsbedingung bezüglich der Entwurfsvariablen gefordert. Im vorliegenden Fall sind das die Dehnsteifigkeiten der einzelnen Stäbe. Da sowohl der E–Modul als auch die Stablängen als konstant anzunehmen sind, werden lediglich die Querschnittsflächen A_i variiert. Das Minimum der Differenzarbeit f aus Gleichung 3.53 lässt sich also durch die Nullstellen der partiellen Ableitungen bezüglich der Entwurfsvariablen ermitteln. Für deren Ermittlung kommt eine Taylor–Reihenentwicklung mit zwei Gliedern zum Einsatz, auch als Newton–Algorithmus bezeichnet (s. Gleichung 3.54 und Abbildung 3.36).

$$\frac{\partial f}{\partial A_i} = \nabla\mathbf{f} + \mathbf{H}\,\Delta A_i = 0 \qquad \text{mit} \qquad \mathbf{H} = \frac{\partial^2 f}{\partial A_i \partial A_j} \tag{3.54}$$

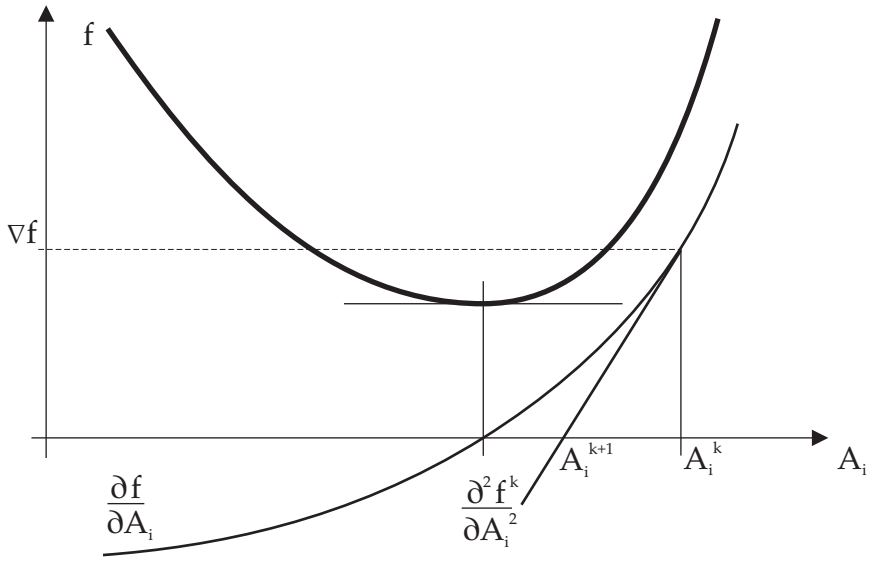


Abbildung 3.36: Zielfunktion f mit deren Ableitungen

Hierin ist mit $\Delta A_i = A_i^{k+1} - A_i^k$ die Verbesserung der Querschnittsfläche A_i des Stabes i gegeben. Aus Gleichung 3.54 lässt sich somit ein Update für die Querschnittsfläche A_i ableiten. Das Vorgehen wird in Abbildung 3.36 verdeutlicht.

$$A_i^{k+1} = A_i^k - \mathbf{H}^{-1}\nabla\mathbf{f} \qquad \text{mit} \qquad \nabla\mathbf{f} = \frac{\partial f}{\partial A_i}\,|_{\mathbf{A}^k} \tag{3.55}$$

Hierbei ist mit der Hesse–Matrix die Schrittweite gegeben, mit der eine Verbesserung der Entwurfsvariablen $\mathbf{A}$ in richtung $\nabla\mathbf{f}$ gesucht wird. Die dafür notwendi-

gen Sensitivitäten bezüglich der Entwurfsvariablen A_i sind aus der Ableitung der Zielfunktion wie folgt herzuleiten.

$$\frac{\partial f}{\partial \mathbf{A}} = \frac{1}{2}\,\Delta\mathbf{u}^T\,\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\Delta\mathbf{u} + \Delta\mathbf{u}^T\mathbf{K}\,\frac{\partial \Delta\mathbf{u}}{\partial \mathbf{A}} \tag{3.56}$$

Die partielle Ableitung der Verschiebungsdifferenz lässt sich vereinfachen.

$$\frac{\partial \Delta\mathbf{u}}{\partial \mathbf{A}} = \frac{\partial \Delta\mathbf{u}}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{A}} = \frac{\partial \mathbf{u}}{\partial \mathbf{A}} \tag{3.57}$$

Dieser Ausdruck kann aus dem Systemgleichgewicht (3.58) durch Ableitung nach den Entwurfsvariablen (3.59) bestimmt werden.

$$\mathbf{K}\,\mathbf{u} = \mathbf{P} \tag{3.58}$$

$$\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\mathbf{u} + \mathbf{K}\,\frac{\partial \mathbf{u}}{\partial \mathbf{A}} = \frac{\partial \mathbf{P}}{\partial \mathbf{A}} \tag{3.59}$$

$$\frac{\partial \mathbf{u}}{\partial \mathbf{A}} = \mathbf{K}^{-1}\left(\overbrace{\frac{\partial \mathbf{P}}{\partial \mathbf{A}}}^{=0} - \frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\mathbf{u}\right) = -\mathbf{K}^{-1}\left(\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\mathbf{u}\right) \tag{3.60}$$

Die Systembelastung $\mathbf{P}$ ist von den Entwurfsvariablen unabhängig. zusammen mit Gleichung 3.56 entsteht folgender Ausdruck.

$$\frac{\partial f}{\partial \mathbf{A}} = \frac{1}{2}\,\Delta\mathbf{u}^T\,\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\Delta\mathbf{u} - \Delta\mathbf{u}^T\mathbf{K}\,\mathbf{K}^{-1}\left(\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\mathbf{u}\right) = -\frac{1}{2}\,\Delta\mathbf{u}^T\,\frac{\partial \mathbf{K}}{\partial \mathbf{A}}\,\Delta\tilde{\mathbf{u}} \tag{3.61}$$

Darin ist $\Delta\tilde{\mathbf{u}} = \mathbf{u} + \hat{\mathbf{u}}$. Die Gesamtsteifigkeitsmatrix $\mathbf{K}$ entsteht durch die Assemblierung der Elementsteifigkeitsmatrizen $\mathbf{k}_i^e$ der n Stabelemente, bezeichnet mit dem Operator S. Der Kopfzeiger e deutet auf elementbezogene Größen hin.

$$\mathbf{K} = \sum_n S\,\mathbf{k}_i^e \tag{3.62}$$

Die in Gleichung 3.61 vorkommenden Sensitivitäten der Steifigkeitsmatrix lassen sich aus der Summe über alle Elemente n ermitteln.

$$\frac{\partial \mathbf{K}}{\partial A_i} = \sum_n S\,\frac{\partial \mathbf{k}_j^e}{\partial A_i} \qquad \text{mit} \quad \frac{\partial \mathbf{k}_j^e}{\partial A_i} = 0 \quad \text{für} \;\; i \neq j \tag{3.63}$$

Da sich die Variation eines Stabquerschnittes A_i nur auf das betreffende Element auswirkt, vereinfacht sich der Ausdruck unter Annahme eines konstanten E-Moduls wie folgt.

$$\frac{\partial \mathbf{k_i^e}}{\partial A_i} = \frac{1}{A_i}\,\mathbf{k}_i^e = \frac{E}{\ell_i}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.64}$$

Damit lässt sich Beziehung 3.61 elementbezogen formulieren.

$$\frac{\partial f}{\partial A_i} = -\frac{1}{2}\,\Delta\mathbf{u}^{eT}\,\frac{\partial \mathbf{k}_i^e}{\partial A_i}\,\Delta\tilde{\mathbf{u}}^e \tag{3.65}$$

Mit $\Delta\mathbf{u}^e = \mathbf{u}^e - \hat{\mathbf{u}}^e$ bzw. $\Delta\tilde{\mathbf{u}}^e = \mathbf{u}^e + \hat{\mathbf{u}}^e$ sind die auf ein Element bezogenen Knotenverschiebungen aus dem Gesamtverschiebungsvektor $\mathbf{u}$ zu extrahieren. Die Auswertung des Ausdrucks 3.65 für A_i^k ist damit für Gleichung 3.55 als $\nabla\mathbf{f}$ gegeben. Als nächstes ist die Hesse–Matrix $\mathbf{H}$ als zweite Ableitung der Zielfunktion f zu bestimmen.

$$\frac{\partial^2 f}{\partial A_i \partial A_j} = -\frac{1}{2}\left(\frac{\partial \mathbf{u}^T}{\partial A_j}\,\frac{\partial \mathbf{K}}{\partial A_i}\,\Delta\tilde{\mathbf{u}} + \Delta\mathbf{u}^T\,\frac{\partial \mathbf{K}^T}{\partial A_i}\,\frac{\partial \mathbf{u}}{\partial A_j}\right) \tag{3.66}$$

Mit Hilfe von Gleichung 3.60 lässt sich die Hesse–Matrix wie folgt formulieren.

$$\frac{\partial^2 f}{\partial A_i \partial A_j} = \frac{1}{2}\left(\mathbf{u}^T\,\frac{\partial \mathbf{K}^T}{\partial A_j}\,\mathbf{K}^{-1}\,\frac{\partial \mathbf{K}}{\partial A_i}\,\Delta\tilde{\mathbf{u}} + \Delta\mathbf{u}^T\,\frac{\partial \mathbf{K}^T}{\partial A_i}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial A_j}\mathbf{u}\right) \tag{3.67}$$

Die darin vorkommenden Inversen der Gesamtsteifigkeitsmatrix $\mathbf{K}^{-1}$ sind in der Berechnung sehr aufwendig und damit rechenintensiv. In der Regel ist für jede Entwurfsvariable und in jeder Iteration zu deren Bestimmung eine Systemanalyse durchzuführen. Nach Beziehung 3.55 ist eine elementweise Betrachtung deutlich effektiver. Die dafür notwendige Hesse–Matrix auf Elementebene kann durch teilweise Vernachlässigung der Kopplung der Entwurfsvariablen A_i näherungsweise bestimmt werden. Das läuft auf eine Diagonalisierung der Hesse–Matrix hinaus, bei der lediglich die Hauptdiagonalterme berücksichtigt werden. Im Rahmen der Anwendung ist diese Näherung ausreichend genau, da mit der Hesse–Matrix lediglich die Schrittweite der Querschnittsverbesserung bestimmt wird. Viel Wichtiger ist die exakte Betrachtung der Suchrichtung ∇f, also der Auswertung der Sensitivitäten mit den aktuellen Entwurfsvariablen.

$$\frac{\partial^2 f}{\partial A_i^2} = \frac{1}{2}\left(\mathbf{u}^T\,\frac{\partial \mathbf{K}^T}{\partial A_i}\,\mathbf{K}^{-1}\,\frac{\partial \mathbf{K}}{\partial A_i}\Delta\tilde{\mathbf{u}} + \Delta\mathbf{u}^T\frac{\partial \mathbf{K}^T}{\partial A_i}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial A_i}\mathbf{u}\right) \tag{3.68}$$

Aufgrund der Symmetrie der Steifigkeitsmatrix vereinfacht sich der Ausdruck zu.

$$\frac{\partial^2 f}{\partial A_i^2} = \mathbf{u}^T \frac{\partial \mathbf{K}^T}{\partial A_i} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial A_i} \mathbf{u} \tag{3.69}$$

Betrachtet man die Sensitivität der Gesamtsteifigkeitsmatrix $\mathbf{K}$ bezüglich einer Entwurfsvariablen A_i, so sind lediglich die Einträge des betreffenden Stabelementes i nicht gleich Null. Eine identische Matrix entsteht durch Ermittlung der Sensitivitäten auf Elementebene und anschließende Assemblierung S in eine Matrix mit der Dimension der Gesamtsteifigkeit.

$$\frac{\partial \mathbf{K}}{\partial A_i} = \sum_n S \frac{\partial \mathbf{k}_i^e}{\partial A_i} = S \frac{\partial \mathbf{k}_i^e}{\partial A_i} = \frac{\partial \mathbf{K}_i^e}{\partial A_i} = \frac{\mathbf{K}_i^e}{A_i} \tag{3.70}$$

Damit kann unter Berücksichtigung von 3.64 auch die zweite Ableitung von f analog zur Beziehung 3.65 elementweise betrachtet werden.

$$\frac{\partial^2 f}{\partial A_i^2} = \mathbf{u}^{eT} \frac{\partial \mathbf{K}^{eT}}{\partial A_i} \mathbf{K}^{-1} \frac{\partial \mathbf{K}^e}{\partial A_i} \mathbf{u}^e = \frac{1}{A_i} \mathbf{u}^{eT} \mathbf{K}^{eT} \mathbf{K}^{-1} \mathbf{K}^e \mathbf{u}^e \tag{3.71}$$

Darin ist der Gesamtverschiebungsvektor $\mathbf{u}$ durch einen Elementverschiebungsvektor $\mathbf{u}^e$ mit der gleichen Dimension ersetzt. Unter Berücksichtigung der Symmetrie der Steifigkeitsmatrix wird der Ausdruck $\mathbf{K}^{eT} \mathbf{K}^{-1} \mathbf{K}^e$ aus Gleichung 3.71 durch den Term $\mathbf{I}^e \mathbf{K}^e = \mathbf{K}^e$ ersetzt. In der „Einheitsmatrix“ $\mathbf{I}^e$ sind lediglich die betreffenden Hauptdiagonalelemente ungleich Null. Durch diese Vereinfachung wird die Kopplung des betrachteten Elementes i mit seinen benachbarten Elementen in der Gesamtsteifigkeit teilweise vernachlässigt. Alle weitere Elemente verschwinden durch die partiellen Ableitungen der vor- und nachmultiplizierten Sensitivitäten.

$$\frac{\partial^2 f}{\partial A_i^2} = \frac{1}{A_i^2} \mathbf{u}^{eT} \mathbf{k}^e \mathbf{u}^e \tag{3.72}$$

Zusammengefasst ist ein Update für jede Entwurfsvariable A_i mit Beziehung 3.73 auf Elementebene durchführbar. Der Index e für den Bezug auf eine Elementgröße wird hier der Übersichtlichkeit halber weggelassen.

$$A_i^{k+1} = A_i^k + \frac{A_i^k \left(\Delta \mathbf{u}^{kT} \mathbf{k}^k \Delta \tilde{\mathbf{u}}^k\right)}{2 \left(\mathbf{u}^{kT} \mathbf{k}^k \mathbf{u}^k\right)} \tag{3.73}$$

Für die Querschnitte A_i^0 werden einheitliche Werte abhängig von den Systemabmessungen angenommen. Die guten Konvergenzeigenschaftenen des Algorithmus in den behandelten Beispielrechnungen deutet auf eine gute Näherung der Schrittweite (Hesse–Matrix) des Newton–Algorithmus hin.

Kapitel 4

Flexible Kopplung entfernter Prozesse

Eine verteilte Anwendung besteht aus einer Menge kommunizierender Betriebssystemprozesse, die nicht über einen gemeinsamen Speicher verfügen und auf unterschiedlichen Rechnern verteilt sein können [Müh92]. Ein zugrunde liegendes verteiltes System, das sich aus mehreren Rechnern zusammensetzt, ermöglicht die Kommunikation der Betriebssystemprozesse auf der Basis einer „physikalischen“ Rechnerkopplung. Dem Benutzer erscheint es, als habe er es mit einem kohärenten System zu tun.

Ist der Anwender in der Lage, steuernd auf das Prozesssystem einzugreifen, kann auch von einer offenen Prozesskopplung gesprochen werden. Bei Simulationsprogrammen geschieht das in der Regel durch interaktive Eingaben. Im Gegensatz dazu hat der Anwender bei einer geschlossenen Prozesskopplung nur noch die Möglichkeit, zu beobachten. Für den vorliegenden Fall ist eine offene Prozesskopplung daher zwingend notwendig.

Die Kommunikation von Prozessen über ein Netzwerk besteht grundsätzlich aus zwei Bausteinen. So müssen sowohl die Hardware als auch die Software gewissen Anforderungen entsprechen. In den folgenden Abschnitten wird ein kurzer Überblick über den Stand der Technik gegeben und ein Konzept zur Kopplung entfernter Prozesse mit Hilfe der Software–Kapselung vorgestellt. Ein besonderer Schwerpunkt der Arbeit liegt in der Übertragung flexibler Datenstrukturen in verteilten Systemen.

4.1 Stand der Technik

In der heutigen Zeit ist in vielen Bereichen offensichtlich, dass Computeranwendungen schon lange nicht mehr nur auf einen Rechner beschränkt sein müssen. Sehr leistungsfähige Netzwerke sind seit ARPANET (Advanced Research Projects Agency Network), dem Vorläufer des heutigen Internets, weltweit im Einsatz und ermöglichen heute schon zuverlässige Übermittlung enormer Datenmengen in kurzer Zeit. Es liegt also nahe, Rechner und Prozesse über diese Netzwerke zu koppeln, was ersichtliche Vorteile mit sich bringt.

In erster Linie dient die Prozesskopplung der Koordination von Prozessen und unterstützt dadurch die Zusammenarbeit zwischen Computern und Menschen. Zum anderen wird bei sehr rechenintensiven Prozessen eine enorme Leistungssteigerung erzielt durch Verteilung auf mehrere Prozessoren oder durch Teilen von speziellen Ressourcen. Die Begriffe *Parallelisierung* von Rechnern oder das *Outsourcing* von Teilprozessen sind in diesem Zusammenhang zu nennen. Durch den gezielten Einsatz spezieller Hard- und Software werden Teilaufgaben äußerst effektiv gelöst. Des weiteren wird eine bessere Verfügbarkeit von Kapazitäten gewährleistet durch das Umgehen von Engpässen. Eine Steigerung der Zuverlässigkeit und Minimierung der Fehlertoleranz sind weitere wichtige Schlagworte. Nicht zuletzt tritt eine Kostenersparnis auf, wenn kleinere Komponenten mit gutem Preis–Leistungs–Verhältnis miteinander zu günstigen aber leistungsfähigen Einheiten verknüpft werden.

Damit wurde eine neue Entwicklungsrichtung für Betriebssysteme und Anwendungen für verteilte Systeme notwendig. Gerade in Disziplinen, die besonders hohe Anforderungen an die Leistungsfähigkeit ihrer Software stellen, ist dies heute schon Stand der Technik. So sind alle gängigen Betriebssysteme auf ihren Einsatz in Netzwerken konzipiert.

Durch die so genannte Interprozesskommunikation (IPC), also einen Austausch von Informationen zwischen kooperierenden parallelen Prozessen, sind Software–Entwickler nicht mehr auf einen Rechnertyp, ein Betriebssystem oder eine Programmiersprache beschränkt. Dies ermöglicht die Verwendung bereits ausgereifter Komponenten, um diese zu neuen Programmsystemen zu verknüpfen. Damit lassen sich innerhalb kürzester Zeit maßgeschneiderte Programmcodes entwickeln, welche ganz neue Aufgaben bewältigen können.

Der zeitliche Versatz zwischen Forschung und Anwendung in der Praxis wird dadurch deutlich reduziert. Forscher könnten auf diese Weise ihren Code in kommerzielle Software über ein Computernetz integrieren, ohne dass der Quellcode ausgetauscht oder umfangreich angepasst werden muss. Die einzige Grundvoraussetzung ist eine Schnittstelle, welche die Programmsteuerung und Datenkommunikation übernimmt.

Im Bauingenieurwesen finden sich hauptsächlich in der Fluid-Struktur-Interaktion (FSI) Lösungsansätze dafür. Hier wird gerne auf das Mesh–based parallel Code Coupling Interface (MpCCI) zurückgegriffen, um nur ein Beispiel zu nennen. Es stellt einen kommerziellen Aufsatz auf dem freien Message Passing Interface (MPI) dar. Zusätzlich zur Datenübermittlung wird durch MpCCI auch das *Mapping* der Daten zwischen zwei unterschiedlichen FE–Netzen vorgenommen.

So existiert eine Vielzahl freier und kommerzieller Softwarepakete, welche zugeschnittene Problemlösungen bieten, jedoch mit den Nachteilen hoher Komplexität

und großem Umfang. Dabei sind die zugrunde liegenden Prinzipien die selben und schon seit der Entwicklung des Netzwerks bekannt.

Im Sinne möglichst hoher Transparenz, Variabilität und Einfachheit wird ein eigenes Kapselungs- und Kopplungsmodul vorgestellt, welches in der folgenden Problemstellung seine Anwendung findet. Integriert in ein Workflow–Management–System ist damit dem Entwickler ein sehr flexibles Softwarekonzept bereitgestellt. Durch seinen modularen Aufbau sind dessen Komponenten schnell austausch- und erweiterbar. Auch über die Rechnergrenzen hinweg können Module plattformunabhängig angeknüpft werden. Durch die Kapselungstechnik sind unterschiedlichste Programmiersprachen integrierbar.

4.2 Aufgabenstellung

Für Ingenieure ist es besonders wichtig, das strukturelle Verhalten ihrer Konstruktionen zu verstehen. In einfachen und bekannten Fällen ist dies nicht mehr schwierig, jedoch kommen immer neue Herausforderungen auf den Ingenieur zu. Gerade in der sich sehr stark entwickelnden 3D–Berechnung ist die Vorstellungskraft vom Tragverhalten komplexer Zustände sehr eingeschränkt. Hier ist die Erweiterung moderner Visualisierungstechniken von großer praktischer Bedeutung. Deren Entwicklung muss schnell und adaptiv auf die Anforderungen des Anwenders eingehen können. Eine Herausforderung dahingehend ist es, ein Konzept zu entwickeln, um neuartige Ideen, egal welcher Art, schnellst möglich umzusetzen.

Besteht die Aufgabe eines Software–Ingenieurs darin, ein komplexes Problem zu lösen, stehen ihm unterschiedliche Strategien zur Verfügung. Klassischer Weise wird er ein Programm suchen, welches die Problemstellung am besten abdeckt. Die fehlenden Teile müssen dann aufwendig im bestehenden Quellcode umgestaltet und ergänzt werden. Zwar sind Algorithmen zu vielen Problstellungen vorhanden, jedoch müssen sie oft umständlich angepasst und integriert werden. Eine weitere Schwierigkeit liegt in der mangelhaften Dokumentation oder in der Verfügbarkeit von Quellcodes. Kommerzielle Software–Hersteller stellen nur selten ihren Quellcode anderen Entwicklern zur Verfügung. Eine weitere Strategie besteht darin, ein der Problemstellung angepasstes Programm völlig neu zu entwickeln. Vorteil dabei ist, dass auf besondere Wünsche eingegangen werden kann und die Performance durch die hohe Spezialisierung enorm ist. Der Nachteil liegt in der längeren Entwicklungszeit und der eingeschränkten Anwendungsmöglichkeit und Erweiterbarkeit durch den hohen Spezialisierungsgrad.

Für eine individuelle und gleichzeitig effektive Lösung sollten beide Strategien kombiniert werden. Komplexe Problemstellungen lassen sich meist in Teilaufgaben zerlegen, für die in den meisten Fällen bereits ausgereifte Lösungen existieren.

Diese müssen dann lediglich miteinander verknüpft werden. Fehlende Teile sind dann sehr schnell ergänzt.

Das Prinzip kann am Beispiel der Fluid–Struktur–Interaktion verdeutlicht werden. Die Aufgabe besteht in der Berechnung einer umströmten Struktur unter Einbeziehung der Wechselwirkung von Fluid und Struktur. Zunächst sorgt ein effektives FE–Programm für die Berechnung der Struktur. Deren Verformungen dienen einem Fluidcode als Randbedingung für eine Strömungssimulation. Die Ergebnisse daraus in Form von Drücken, werden wiederum dem FE–Code als Belastung übergeben. Eine iterative Berechnung in der ständig Daten zwischen den Modulen ausgetauscht werden, ergibt die erwünschte Wechselwirkung. Die Aufgabe ist am schnellsten durch eine Kopplungssoftware umzusetzen, welche die betreffenden Datenströme zwischen den beiden bestehenden Programmen ermöglicht.

Die Lösung besteht darin, ein systemunabhängiges Kopplungsmodul zu generieren, welches die genannten Programmmodule via Netzwerk miteinander verknüpfen kann. Dabei soll auf Fileschnittstellen verzichtet werden. Dadurch ist eine interaktive Kommunikation möglich und der Datenstrom kann auf ein Minimum beschränkt werden. Das Modul soll einfach und leicht erweiterbar sein und auf jede erdenkbare Problemstellung angepasst werden können. Des weiteren sollte der Eingriff in die bestehende Software minimal ausfallen. Somit können unterschiedliche Programmierer ihren Quellcode leicht anpassen, ohne ihn für andere zur Verfügung stellen zu müssen. Damit ist das Problem gelöst, auch über große Entfernungen hinweg komplexe Programmsysteme zu schaffen.

Am Beispiel der Visualisierung von Berechnunsergebnissen eines Finite–Elemente–Programms (CARAT) soll die Umsetzung des Konzepts getestet werden. Das auf FORTRAN basierende Programm ist auf LINUX- und UNIX–Systemen portiert. Zur Darstellung der Ergebnisse wird eine kommerzielle Visualisierungssoftware von der Firma Advanced Visual System Inc. (AVS) verwendet, die für alle gängigen Betriebssysteme Implementierungen besitzt. Eine Anbindung von eigenem Quellcode in FORTRAN, C und C++ wird ermöglicht. Eine weitere Aufgabe ist die Visualisierung des Tragverhaltens scheibenartiger Tragwerke mit Hilfe von Stabwerkmodellen. Ein eigens dafür entwickeltes Modul in C++ (ALFIT) ist für die gängigen Betriebssysteme umgesetzt. Eine anschließende Berechnung des Stabwerkes zur Abschätzung der Kräfteverteilung wird durch ein eigenes objektorientiertes Fachwerkprogramm in C++ (TRUSS) durchgeführt.

Der Umgang mit unterschiedlichen Programmiersprachen, das so genannte Mixed Language Programming (MLP), ist eine besondere Herausforderung bei der Verwendung bestehender Software (siehe dazu Kapitel 2.4 und Anhang B). Durch die Aufgabenstellung entsteht zusätzlich eine Konfigurationen mit Verwendung unterschiedlicher Rechnerarchitekturen.

Zur Umsetzung ist teils eine unidirektionale teils eine bidirektionale Kommunikation notwendig. In Kapitel 4.6 wird dafür ein Konzept vorgestellt, welches auf einem simplen Client–Server–Modell basiert und über TCP/IP kommuniziert. Damit werden alle Anforderungen einer flexiblen IPC Software (Inter–Process Communication) erfüllt.

4.3 Rechnernetze

Vor der softwaretechnischen Umsetzung der Kommunikation in einem verteilten System muss die dafür notwendige Peripherie geschaffen werden. Dafür werden in den folgenden Abschnitten grundlegende Begrifflichkeiten geklärt. Ein detaillierter Überblick findet sich in Tanenbaum [Tan03].

4.3.1 Begriffe

Ein Rechnernetz (Computer Network) ist ein Zusammenschluss von verschiedenen technischen, primär selbstständigen elektronischen Systemen, insbesondere Computern, aber auch Sensoren, Aktoren, funktechnologischen Komponenten usw., welches die Kommunikation der einzelnen Systeme untereinander ermöglicht.

Ein verteiltes System besteht aus mehreren Prozessoren (CPU's) und Speichermodulen. Dabei ist es nicht erforderlich, dass jede Einheit mit jeder anderen verbunden ist. Sie müssen lediglich zusammenhängend verknüpft sein. Diese physikalische Verbindung zwischen den funktionalen Einheiten bildet das Rechnernetzwerk.

Die unterschiedliche Anordnung und Zugriffsmöglichkeit der CPU's auf den Speicher teilt die Hardwarekonfiguration in zwei Gruppen. Multiprozessoren bestehen aus mehreren CPU's, die auf einen gemeinsamen Speicher zugreifen. Systeme, die keinen gemeinsamen Speicher besitzen, werden als Multicomputer bezeichnet. Sind sie im Wesentlichen über ein gleich geartetes Verbindungsnetz miteinander verbunden und greifen zugleich auf ähnliche CPU–Leistung und Speicherkapazitäten zurück, handelt es sich um homogene Multicomputer. Sie werden immer häufiger als parallele Systeme eingesetzt, ähnlich den Multiprozessoren.

Da eine derartige Konfiguration eher selten anzutreffen ist, wird im weiteren Verlauf ein heterogenes Multicomputersystem vorausgesetzt. Es verbindet verschiedene unabhängige Computer über unterschiedliche Netzwerke miteinander.

4.3.2 Netzwerk–Topologien

Die Topologie bezeichnet bei einem Rechnernetz die Struktur der Verbindungen der Geräte zueinander. Die Kenntnis der Topologie eines Netzwerkes ist nützlich zur Bewertung seiner Performance und Ausfallsicherheit, aber auch für die Investitionskosten und Auswahl geeigneter Hardware für das Netzwerk. Abbildung 4.1 zeigt derzeit gebräuchliche physikalische Topologien.

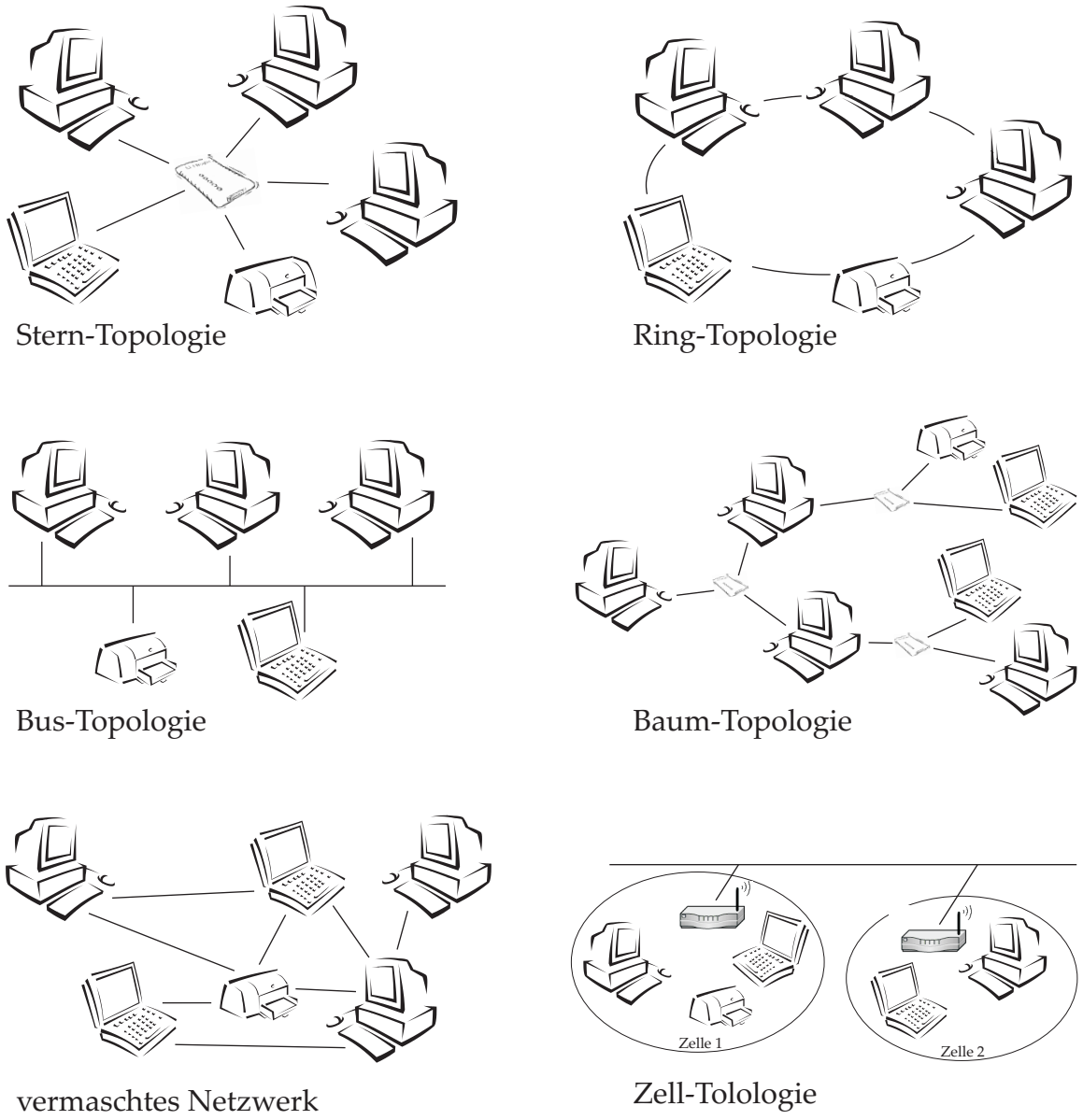


Abbildung 4.1: Gebräuchliche physikalische Netzwerktopologien

Es wird zwischen physischer und logischer Topologie unterschieden. Die physische Topologie beschreibt den Aufbau der Netzwerkverkabelung, während die logische Topologie den Datenfluss zwischen den Endgeräten darstellt. Weitere Erläuterungen dazu finden sich z.B. in Tanenbaum [Tan02] und [Tan03].

4.3.3 Zur organisatorischen Abdeckung von Netzwerken

Rechnernetzwerke können nach ihrer organisatorischen Abdeckung in lokale und nicht-lokale (globale) Netze eingeordnet werden. Zur ersten Kategorie gehören das *Personal Area Network* (PAN) sowie das *Local Area Network* (LAN). Beide werden auch als kabellose Varianten (Wireless) WPAN und WLAN eingesetzt. Zu den nicht-lokalen Netzwerken zählen das *Metropolitan Area Network* (MAN), das *Wide Area Network* (WAN), das *Global Area Network* (GAN) und das *Virtual Private Network* (VPN).

4.3.4 Anforderungen an Netzwerke

Es gibt unterschiedliche Betrachtungsweisen über die Anforderungen an Netzwerke, die bei deren Planung berücksichtigt werden müssen. Dabei ist es meist nicht möglich, allen Ansprüchen gleichzeitig gerecht zu werden. Um so wichtiger ist es, einen geeigneten Kompromiss zu finden.

Für den Anwender bzw. Programmierer müssen Netzwerke komfortabel und ansprechend bedien- und programmierbar sein, eine hohe Leistungsfähigkeit und Zuverlässigkeit besitzen und kostengünstig sein. Aus der Sicht des Netzwerk Designers sind hohe Flexibilität, Erweiterbarkeit und ebenfalls möglichst geringe Kosten des Netzwerks gewünscht. Netzwerk Anbieter hingegen fordern geringe Komplexität, einfache Handhabung, Überwachung und Bilanzierung ihres Netzwerks.

4.3.5 Beispiele für Netzwerke

In fast allen Ländern werden spezielle Hochleistungsnetze für Wissenschaft, Forschung und Lehre bereitgestellt. In folgender Tabelle findet sich eine kleine Auswahl von verschiedenen Ländern.

Australien	AARNET	Italien	GARR
Canada	Canarie Inc.	Japan	SINET
China	CERNET	Niederlande	SURFnet
Deutschland	DFN	Österreich	ACOnet
Frankreich	RENATER	Schweiz	SWITCH
Griechenland	GRNET	Spanien	RedIRIS
Großbritannien	UKERNA	USA	UCAID

Tabelle 4.1: Internationale Netzwerke

Das Deutsche Forschungsnetz (DFN) verbindet Hochschulen und Forschungseinrichtungen miteinander und unterstützt die Entwicklung und Erprobung neuer Anwendungen. Der nationale Backbone des DFN ist das Gigabit–Wissenschaftsnetz G–WiN (vgl. Abbildung 4.2).

Abbildung 4.2: Das Gigabit–Wissenschaftsnetz des Deutschen Forschungsnetzes

Über den europäischen Backbone GÉANT ist das G–WiN mit dem weltweiten Verbund der Forschungs- und Wissenschaftsnetze direkt verbunden. Das schafft die notwendige, leistungsfähige Infrastruktur für global verteilte Anwendungen. Damit ist das in Abschnitt 4.6 beschriebene Konzept nicht mehr auf ein lokales Netzwerk beschränkt.

4.4 Grundlagen einer Prozesskopplung

Um die auftretenden Schwierigkeiten der Prozesskopplung zu verstehen, ist die Kenntnis grundlegender Begrifflichkeiten hilfreich. An dieser Stelle wird der Prozess und die einfachen Prinzipien gekoppelter Prozesse etwas näher beleuchtet (vgl. hierzu [Dij65] und [Han73]).

4.4.1 Der Prozess

In der Informatik wird unter einem Prozess ein dynamisches Objekt verstanden. Er besteht aus einer Sequenz von Anweisungen oder Operationen, durch die eine in sich abgeschlossene Aufgabe bearbeitet wird. Dabei besitzt der Prozess einen systemweit eindeutigen Namen sowie einen definierten Entstehungs- und Terminierungszeitpunkt (nach [Neh85]).

Der Begriff *Operation* lässt sich am besten anhand seiner mathematischen Eigenschaft erklären. Eine Operation ist demnach eine Vorschrift, die aus Eingabedaten zeitunabhängige reproduzierbare Ausgabedaten erzeugt. Wegen der Reproduzierbarkeit schließen sich Operationen, die auf die selben Daten zugreifen und manipulieren, einander zeitlich aus.

4.4.2 Konkurrierende Prozesse

Bei der nichtsequenziellen Programmverarbeitung bestehen Systeme im Allgemeinen aus einer Vielzahl koexistenter Prozesse. Sie sind voneinander vollständig entkoppelt, wenn die Datenbereiche der beteiligten Prozesse disjunkt sind (vgl. Abbildung 4.3 links). Gewöhnlich werden umfangreiche Aufgaben in einem Datenverarbeitungssystem durch mehrere Prozessen gemeinsam erledigt. Dabei werden Informationen zwischen den beteiligten Prozessen ausgetauscht, sie sind somit gekoppelt. Daraus ergibt sich die Problematik der Prozesssynchronisation.

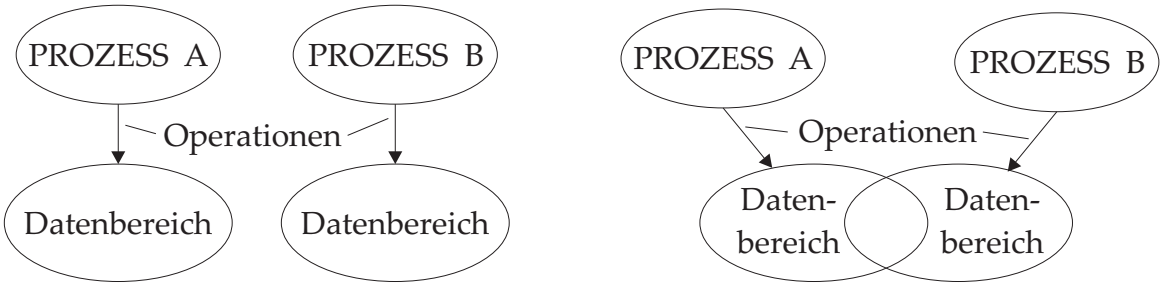


Abbildung 4.3: Koexistente Prozesse (links) und konkurrierende Prozesse (rechts)

Das Problem der Ablaufintegrität bringt in einem gekoppelten Prozesssystem die Notwendigkeit mit sich, die Vorgehensweise der beteiligten Prozesse zu koordinieren. In diesem Fall spricht man von kooperierenden Prozessen. Ablaufintegritätsbedingungen regeln das geordnete Zusammenwirken gekoppelter Prozesssysteme, damit ein funktionales, d.h. zeitunabhängiges Verhalten gewährleistet ist.

Prozesse haben prinzipiell zwei Kommunikationsmöglichkeiten. Sie können zum einen Nachrichten über einen gemeinsamen Datenbereich austauschen, wie in Abbildung 4.3 rechts zu sehen ist. Zum anderen können Daten oder Nachrichten untereinander verschickt werden.

Wenn das Zusammenwirken von Prozessen in einer verteilten Anwendung auf einen Austausch von Informationen zurückgeführt wird, kann dies insbesondere durch eine verteilte Datenbank geschehen, die zugleich den gemeinsamen Zugriff der Prozesse steuert. Ein derartig verteiltes Datenverarbeitungssystem kann aus herkömmlichen autonomen sequenziellen Programmen bestehen, die gemeinsam

auf die verteilte Datenbank zugreifen und nichts voneinander wissen. Für den Zugriff ist es ausreichend, den gemeinsamen Datenbereich für den Zeitraum der Nutzung exklusiv zu belegen und damit gegen Zugriffe weiterer Prozesse zu schützen. Die klassische Synchronisation dieser Zugriffe geschieht mit *Semaphore* oder mit *Monitore* (s. Dijkstra [Dij65] oder Brinch und Hansen [Han73]).

Bei Fehlern in der Synchronisation können zeitabhängige Diskrepanzen auftreten, die entweder zu einer Verletzung der Ablaufkonsistenz oder zu Verklemmungen führen können. Das ist ein Zustand, in dem Prozesse wechselseitig auf eine Freigabe warten und somit auf Dauer blockiert bleiben.

Die Kommunikation über einen gemeinsamen Adressbereich ist in verteilten Systemen in der Regel nicht praktikabel. Deswegen wird die zweite Variante, das so genannte *Message Passing* für die beschriebene Interprozesskommunikation (IPC) verwendet (Botschaften–gekoppeltes Prozesssystem). Dafür müssen vom System Aufgaben wie Senden und Empfangen übernommen werden. In verteilten Systemen sind dafür folgende Kommunikationsmodelle möglich. Für

1-to-1–Kommunikation:

Bei diesem Modell schickt ein Sender genau einem Empfänger eine Nachricht. Meist als Client–Server–Kommunikation bezeichnet, spielt dieses Kommunikationsmodell besonders in verteilten Systemen eine zentrale Rolle. Der *Remote Procedure Call* (RPC) stellt ebenfalls ein derartiges Modell dar und wurde speziell für verteilte Systeme entwickelt. Für die Prozesskopplung wird im Folgenden das Client–Server–Modell verwendet, da jeweils zwei bekannte Kommunikationspartner Daten untereinander austauschen.

1-to-n–Kommunikation:

Eine Nachricht wird von einem Sender an n Empfängern verschickt. Dieses Modell wird auch als *Multicast* bezeichnet. Dafür müssen alle Empfänger im System explizit bekannt sein.

1-to-all–Kommunikation:

Eine nicht speziell an definierte Empfänger gerichtete Botschaft wird *Broadcast* genannt. Ein Sender schickt allen Empfängern eines Systems eine Nachricht. Dieses Modell wird dann verwendet, wenn in einem verteilten System die Empfänger nicht explizit bekannt sind.

4.5 Netzwerk Software

Während sich Abschnitt 4.3 mit der physikalischen Kopplung von Rechnern beschäftigt, wird im Folgenden der softwaretechnische Aspekt bei der Interprozesskommunikation betrachtet. Dabei spielt das Betriebssystem eine wesentliche Rolle. Das im Rahmen dieser Arbeit verwendete heterogene Multicomputersystem bedingt den Einsatz von Netzwerkbetriebssystemen. Zusammen mit geeigneter Anwendungssoftware sind sie in der Lage, mit den vernetzten Rechnern und deren Prozessen Nachrichten und Daten auszutauschen.

Die so genannte Netzwerkprogrammierung beschäftigt sich mit der Erstellung von Programmen, die über ein Computernetzwerk miteinander kommunizieren. Der Übersicht halber wird der grundlegende Aufbau von Computernetzen am weit verbreiteten ISO/OSI–Schichtenmodell (s. Abschnitt 4.5.3) erläutert. Die Kommunikation erfolgt über verschiedene Protokolle, die durch das Schichtenmodell klassifiziert werden können.

Im Folgenden werden Sockets als ein leistungsfähiges, universelles Kommunikationsmittel vorgestellt, das insbesondere für verteilte Systeme geeignet ist. Anschließend wird deren Realisierung im Internet unter Verwendung des TCP/IP beschrieben. Darauf greift das später beschriebene Kopplungsmodul bei der Kommunikation zurück. Als Alternativlösung bilden zwei spezialisierte Ansätze zur verteilten Programmierung den Abschluss, der Remote Procedure Call (RPC) und das objektorientierte Pendant dazu, die Remote Method Invocation (RMI).

4.5.1 Ein Client–Server–Modell

Die Kommunikation zwischen zwei Prozessen über ein Netzwerk basiert in der Regel auf einer Client–Server–Beziehung. Ein Server ist ein Prozess, der einen bestimmten Dienst zur Verfügung stellt. Der Client wiederum ist ein Prozess, der vom Server diesen Dienst anfordert und auf die Antwort wartet. Die Anfrage geht immer vom Client aus, weswegen ihm alle alle Daten zum Aufbau einer Verbindung bekannt gegeben werden müssen. Bekannte Beispiele für Client–Server–Beziehungen sind Webbrowser und Webserver, FTP–Client und FTP–Server oder Telnet–Client und Telnet–Server.

Im 1-1–Kommunikationsmodell ist eine derartige Verbindung, bezogen auf die Aufgabenstellung, am sinnvollsten (vgl. Abbildung 4.4 links), da jeweils nur zwei Prozesse Daten untereinander austauschen müssen. Eine Erweiterung auf ein 1-n–Modell im Bild rechts zu sehen ist jedoch ohne weiteres möglich.

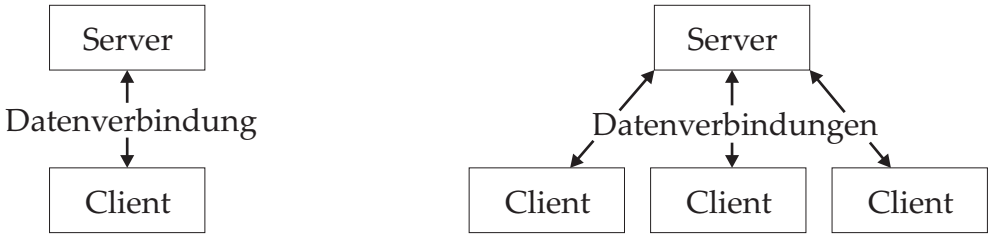


Abbildung 4.4: Client–Server Beziehungen

4.5.2 Nachrichtenorientierte Kommunikation mit Sockets

Die nachrichtenorientierte Kommunikation stellt die allgemeinste Form der Kommunikation in verteilten Systemen dar. Als Teil von Middleware–Lösungen wird im weiteren Verlauf die Datenübergabe über Sockets auf der Transportebene betrachtet. Dabei handelt es sich um Endpunkte einer nachrichtenorientierten Kommunikationsverbindung mit jeweils einem eindeutig zugeordneten Prozess. Dafür stehen eine einfache Menge standardisierter, elementarer Funktionen zur Verfügung, womit eine Standardschnittstelle für unterschiedlichste Systeme erzeugt werden kann.

Vom Konzept her ist ein Socket ein Kommunikationsendpunkt, an dem eine Applikation Daten schreiben kann, die über das zugrunde liegende Netzwerk versendet werden und von dem eingehende Daten gelesen werden können.

Mit Einführung der hochleistungsfähigen Multicomputer werden darüber hinausgehende Funktionen notwendig, welche die Entwicklung von effizienteren Applikationen ermöglichen. Dafür existiert ein von der Hardware unabhängiger Standard für die Nachrichtenübergabe innerhalb eines Hochgeschwindigkeits–Verbindungsnetzwerkes, auch als Message–Passing Interface (MPI) bekannt.

Für die einfache Kommunikation über verschiedene Netzwerke werden im Folgenden die Berkley–Sockets verwendet.

4.5.3 Das ISO/OSI–Referenzmodell

Da sich der Markt aus einer Vielzahl unterschiedlicher Systeme zusammensetzt, ist die Kommunikation untereinander nicht ganz einfach. Um die Komplexität dieser Problematik zu verringern, sind die meisten Netze in Schichten aufgebaut. Sie haben die Aufgabe, den jeweils höheren Schichten bestimmte Dienste zur Verfügung zu stellen, ohne sie mit Einzelheiten zu überladen. Der Anwender kann den komplexen Vorgang der Kommunikation durch diesen hohen Abstraktionsgrad sehr einfach steuern. Die *International Standards Organization* (ISO) hat sich mit einem Modell beschäftigt, dem sogenannten *open system interconnection* (OSI), zur Verknüpfung beliebiger Rechner mit unterschiedlichen Architekturen und Betriebssystemen.

Im ISO–OSI–Referenzmodell (vgl. Abbildung 4.5) wird das Kommunikationssystem in sieben Schichten unterteilt.

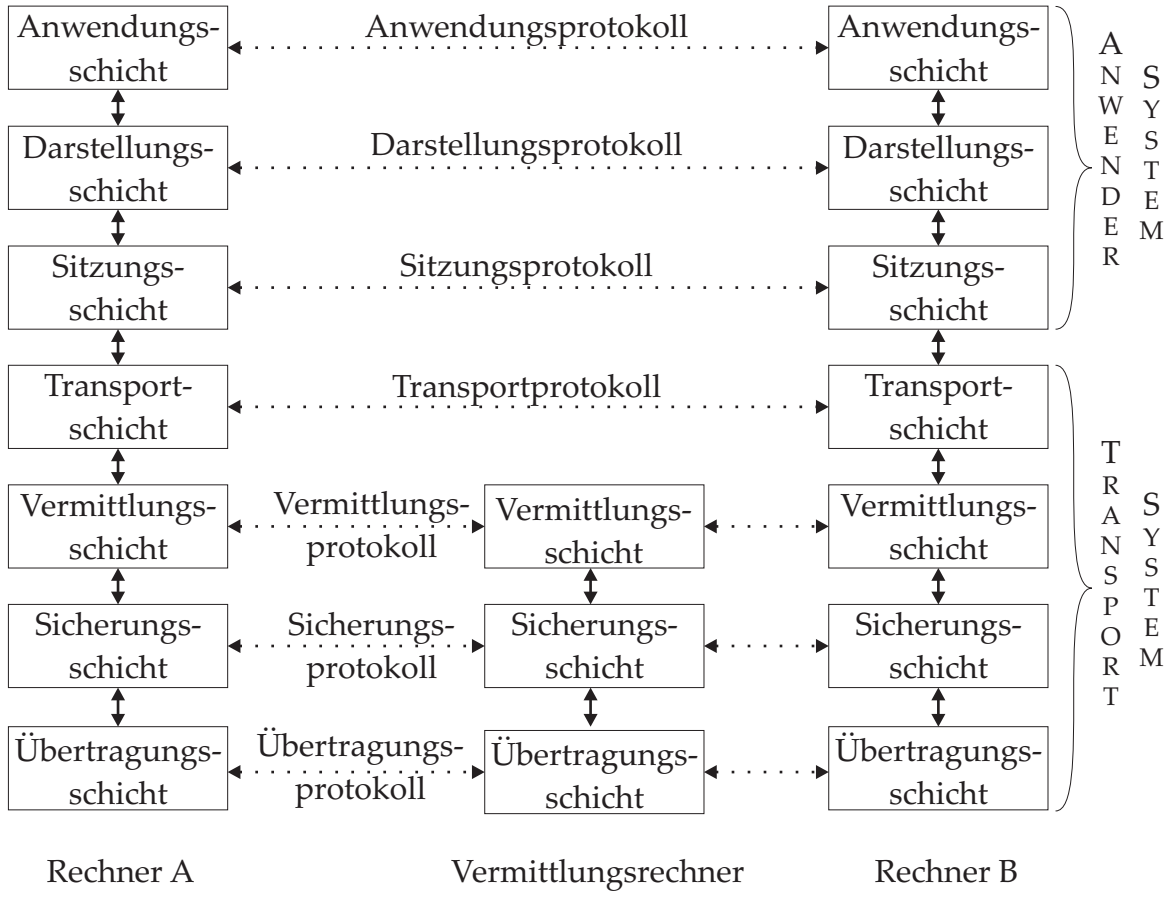


Abbildung 4.5: Netzwerkarchitektur des ISO-Referenzmodells nach der OSI

Die Anwendungsschicht enthält eine Vielzahl vom Benutzer benötigter Protokolle wie das *Hypertext Transfer Protocol* (HTTP) oder das *File Transfer Protocol* (FTP). Zudem stellt sie die unterschiedlichsten Kommunikationsdienste zur Verfügung, beispielsweise E–Mail, Terminal–Emulationen und Webbrowser. Damit Computer mit unterschiedlichen Datendarstellungen kommunizieren können, müssen die ausgetauschten Datenstrukturen von der Darstellungsschicht in ein allgemein lesbares Format kodiert werden. Die Sitzungsschicht ermöglicht eine strukturierte Kommunikation zwischen Prozessen mit Hilfe von Diensten zur Dialogsteuerung Synchronisation und zur Token–Verwaltung. Die Transportschicht hat die Aufgabe, Daten von der Sitzungsschicht evtl. in kleinere Einheiten zu unterteilen und dafür zu sorgen, dass alle Teile möglichst effizient beim richtigen Empfänger ankommen. Die Vermittlungsschicht steuert den Betrieb des Verbindungsnetzes mit Wahl der Paketrouten. Die Hauptaufgabe der Sicherungsschicht schließlich ist es, den Bitstrom richtig zu interpretieren, der von der Übertragungsschicht über einen Kommunikationskanal, das physikalische Übertragungsmedium, empfangen wird.

Das Anwendersystem bestehend aus den obersten drei Schichten ist für die Datenverschlüsselung und -komprimierung zuständig. Die untersten vier Schichten zusammengefasst bilden das so genannte Transportsystem und sind für die eigentliche Übertragung der Daten zuständig.

Die meisten Implementierungen eines derartigen Systems halten sich jedoch nicht strikt an diese Unterteilung und fassen meist mehrere Schichten zusammen. Die untersten beiden werden oft sogar Hardware–technisch auf den Netzwerkkarten realisiert. Die Verknüpfung zwischen Hard- und Software durch die Vermittlungsschicht erfolgt dann über so genannte *Device Driver*. Die obersten vier Schichten sind in der Regel in Funktionsbibliotheken zusammengefasst, auf die die eigentliche Anwendung zugreift, wie in Abschnitt 4.5.6 noch deutlich wird.

Werden Datenpakete zwischen Anwendungen auf unterschiedlichen Rechnern auf weitere Strecken ausgetauscht, so geschieht dies in der Regel über Vermittlungsrechner (siehe Bild 4.5). Sie haben lediglich die Aufgabe, Datenformate bei der Kommunikation in unterschiedlichen Netzen aufeinander abzubilden.

4.5.4 Netzwerkprotokolle

Eine allgemeine Prozesskopplung muss nicht auf ein lokales Rechnernetz beschränkt sein. Es liegt also nahe, die derzeit größte Vernetzung von Rechnern weltweit, das Internet, als Übertragungsmedium zu nutzen. Bei der Kommunikation bedienen sich sowohl Server- als auch Client–Anwendungen den sogenannten Netzwerkprotokollen. Es gibt zwei wichtige Transportprotokolle, das *Transmission Control Protocol* (TCP) und das *User Datagram Protocol* (UDP), die beide auf das IP–Protokoll aufsetzen und sich auf der Transport–Ebene des ISO/OSI–Schichtenmodells befinden.

User Datagram Protocol (UDP)

Das UDP ist ein einfaches, aber unzuverlässiges Transportschichtprotokoll, welches nicht sicherstellt, dass Daten korrekt übertragen werden, oder überhaupt ihr Ziel erreichen. Zwischen Client und Server besteht keine längerfristige Beziehung (verbindungsloser Dienst), weswegen auf der selben Verbindung unmittelbar hintereinander Datagramme an unterschiedliche Server verschickt werden können. Ein UDP–Server kann folglich auf einem Socket nacheinander mehrere Datagramme von unterschiedlichen UDP–Clients erhalten. Für datensatzorientierte Applikationen mit der Forderung nach hoher Übertragungsgeschwindigkeit eignet sich das UDP besonders gut, wie zum Beispiel für Video- oder Audio–Echtzeitübertragungen. Für Anwendungen, bei denen Übertragungsfehler weitreichende Folgen hätte, wie bei FE–Software–Systemen, ist das UDP–Protokoll nicht geeignet.

Transmission Control Protocol (TCP)

Das TCP bietet Zuverlässigkeit beim Datentransfer durch Quittungen. Fehlen diese Bestätigungen werden weitere Übertragungsversuche unternommen und der Vorgang erst abgebrochen, nachdem mehrere Versuchen erfolglos bleiben. Das TCP enthält dafür einen Algorithmus zur dynamischen Bestimmung der Round–Trip–Time (RTT) zwischen Client und Server, so dass das Protokoll weiss, wie lange es auf eine Quittung warten soll. Dies ist abhängig vom Netzwerk (LAN, WAN) und von der Netzwerkbelastung.

Die Verbindung mit TCP ist vollduplex, das heißt eine Anwendung kann jederzeit gleichzeitig Daten senden und empfangen. Dabei werden vom TCP laufend Statusinformationen wie Sequenznummern und die Fenstergröße für jede Richtung des Datenflusses verfolgt. Durch die Flusssteuerung wird sichergestellt, dass der Empfängerbuffer nicht vom Sender zum Überlaufen gebracht werden kann.

Jedem verschickten Byte wird zusätzlich eine Sequenznummer mitgegeben. Kommen Segmente ungeordnet oder mehrfach beim Empfänger an, werden diese vom TCP automatisch sortiert und die doppelt vorhandenen gelöscht.

Aufgrund der hohen Sicherheit bei der Datenübertragung wird im Folgenden TCP verwendet.

4.5.5 Das TCP/IP–Modell

Historisch gesehen sind die Protokolle der Internet–Protokoll–Familie nach dem ISO–OSI–Referenzmodell aufgebaut. Wegen dessen Komplexität wurde daraus jedoch ein vereinfachtes TCP/IP–Referenzmodell geschaffen. Es findet in vielen Bereichen Anwendung wie beispielsweise in UNIX- oder LINUX–Netzwerken und im genannten globalen Internet.

Der Datentransfer wird mittels TCP–Protokoll abgewickelt, das sich wiederum dem Internet Protokoll (IP) bedient, welches direkt auf die Datenübertragungsschicht zugreift. Im gezeigten Beispiel in Abbildung 4.6 geht der tatsächliche Datenfluss zwischen der Client–Anwendung und der Server–Anwendung über den Protokoll–Stack und das Netzwerk, hier das Ethernet.

Die Vermittlungsschicht ist für die Datenübertragung über weite Strecken verantwortlich und stellt sicher, dass die Daten dem richtigen Empfänger zugestellt werden können und dann auch ankommen. Die Netzwerkschicht wird von den IP–Protokollen bedient. Die Transportschicht (TCP) ist für die Daten der Applikation verantwortlich und sorgt dafür, dass sie in der richtigen Reihenfolge ankommen und nichts verloren geht. Die Anwendungsschicht selbst befasst sich mit der Datenverarbeitung.

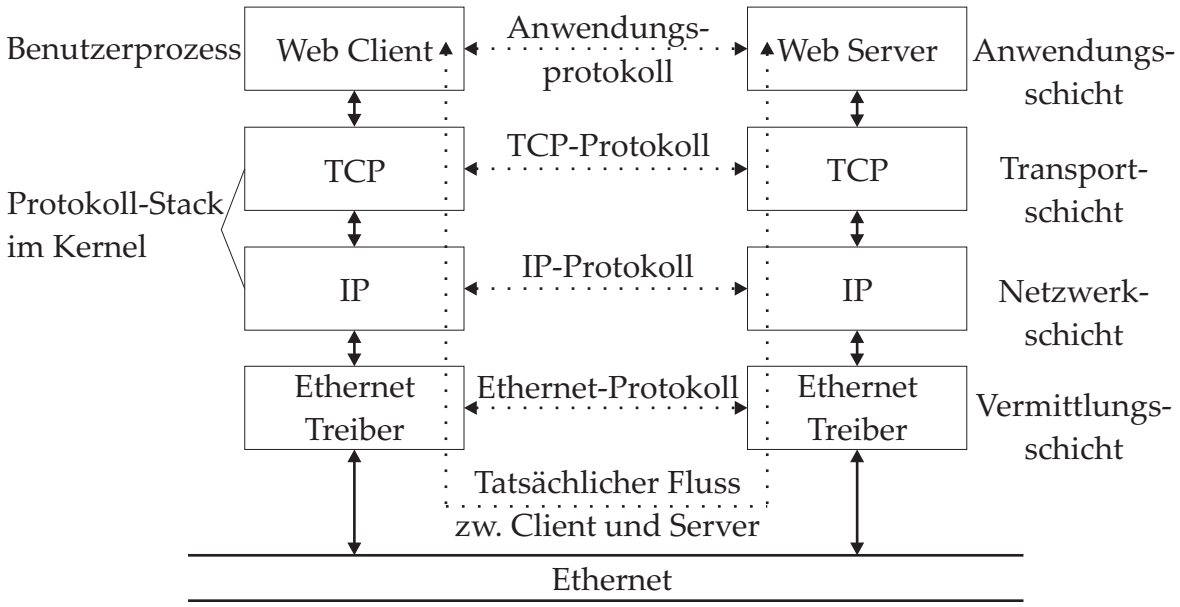


Abbildung 4.6: Datenfluss zwischen Client und Server im selben Netz

Eine Netzwerkverbindung von Client und Server über ein WAN (Wide Area Network), welches zwei LAN's mittels so genannter Router miteinander verbindet, ist eine typische Struktur für das Internet (vgl. Bild 4.7).

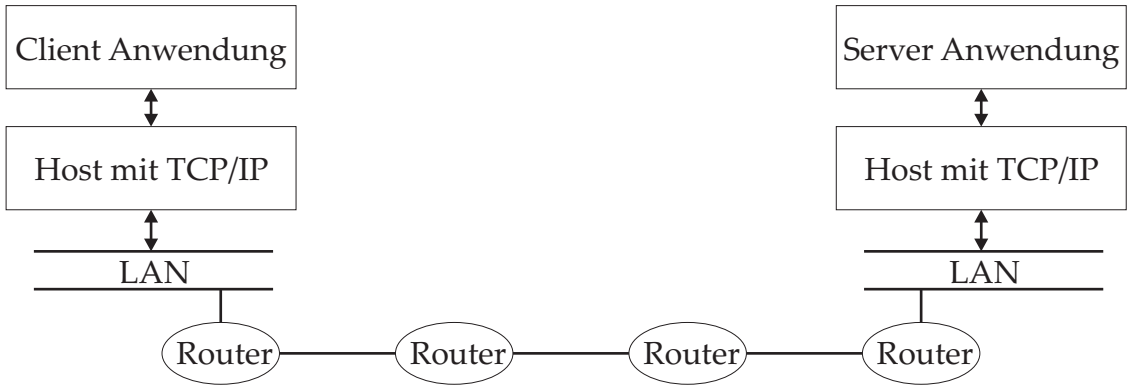


Abbildung 4.7: Datenfluss zwischen Client und Server in unterschiedlichen Netzen

Außer dem Ethernet gibt es noch andere Vermittlungsschichten, die auch als Sicherungsschicht bekannt sind, wie FDDI und ISDN. Übertragen werden die Daten letztendlich über Glasfaserkabel oder herkömmliche Kabel.

4.5.6 Die TCP/IP–Verbindung

Für die Kommunikation zwischen Client und Server über eine TCP/IP–Verbindung müssen beide Prozesse eindeutig im Netz bestimmt werden. Erst durch die gegenseitige Identifizierung beider Prozesse kann eine Verbindung zueinander aufgebaut werden. Besteht diese, erfolgt die Kommunikation durch den Austausch von Datenpaketen. Sind alle Aufgaben erledigt, muss die Verbindung wieder getrennt werden.

Der Wunsch einer Verbindung geht immer vom Client aus. Der Server muss jedoch für eine Verbindung vorbereitet sein, bevor der Client den Aufbau der Verbindung injiziert. Das Senden und empfangen von Daten kann dann von beiden Seiten ausgehen. In der Regel fordert jedoch der Client Daten vom Server an. Das Ende einer Verbindung kann wieder von beiden Kommunikationspartnern ausgehen.

Genauere Erläuterungen zu den Bibliotheksfunktionen sowie eine vollständige Client–Server–Verbindung ist in Anhang A abgedruckt. Weitere Hinweise finden sich in Stevens [Ste00], Hunt [Hun03] sowie Fischer und Müller [Fis99].

Aufbau der Verbindung

Die eindeutige Identifikation von Prozessen im Internet ist durch *Sockets* möglich. Sie wurden ursprünglich für das BSD–Unix–Betriebssystem 1983 entwickelt, definieren jedoch mittlerweile einen De facto–Standard. Ein Socket setzt sich aus der IP–Adresse des Rechners und dem TCP–Port des Prozesses zusammen, und ist damit eindeutig. Er ist nichts anderes als ein *File–Deskriptor*, also ein Handle auf ein I/O–Device. Das heißt natürlich auch, dass ein Socket mit ganz normalen `read()` und `write()` Anweisungen angesprochen werden kann.

Ein *Socket–Paar*, bestehend aus den Sockets der beiden miteinander kommunizierenden Prozesse, definiert somit eine TCP/IP–Verbindung eindeutig. Durch Aufruf seines Deskriptors mit Angabe der Protokollfamilie und der gewünschten Übertragungsart wird der Socket vom Betriebssystem angefordert.

```
int socket(int family, int type, int protocol);
```

Im vorliegenden Fall soll eine sichere streamorientierte Verbindung für einen bidirektionalen Datenaustausch mit dem Standard–Protokoll TCP aufgebaut werden.

Sind für beide Prozesse eindeutige Sockets definiert, muss der Server in einen Wartezustand versetzt und auf eingehenden Verbindungswünsche von Clients vorbereitet werden. Dies geschieht im Wesentlichen durch die drei Routinen `bind()`, `listen()` und `accept()`.

Zunächst wird dem zuvor erzeugten Socket durch `bind()` eine Protokolladresse zugeordnet, also mitgeteilt, zu welchem Port er gehört. Nur wenn die Portnummern von Server und Client übereinstimmen, wird eine Verbindung zustande kommen.

```
int bind(int sockfd, struct sockaddr_in *my_addr, int addrlen);
```

Hier wird der Socket–Deskriptor und die protokollspezifische Adresse des Servers übergeben, welche die eigene IP–Adresse und Portnummer beinhaltet.

Die Funktion `listen()` dient dazu, den Socket des Servers in den passiven Modus zu versetzen, d.h. die Programmausführung zu unterbrechen und auf Verbindungswünsche von Clients zu warten.

```
int listen(int sockfd, int backlog);
```

Zusammen mit dem Socket–Deskriptor wird die Anzahl der Verbindungsanfragen definiert, die maximal in eine Warteschlange gestellt werden sollen.

Daraus entnimmt die `accept()`–Anweisung die erste Anfrage und erzeugt dafür einen neuen Deskriptor, der vom Kernel automatisch erzeugt wird.

```
int accept(int sockfd, void *addr, int *addrlen);
```

Dabei muss der horchende Socket, an dem der Verbindungsaufbauwunsch eingetroffen ist und der üblicherweise während der gesamten Lebenszeit des Servers existiert, angegeben werden. Der Rückgabewert von `accept()` ist ein verbundener Socket, der jedem Verbindungswunsch eines Clients neu zugeordnet wird. In der übergebenen Adressstruktur werden die Informationen über den Verbindungspartner, seine IP–Adresse und die Portnummer gespeichert.

Jetzt erst kann der Client mit der Funktion `connect()` eine Verbindung zu einem wartenden Server aufbauen, sofern dieser einen Socket besitzt.

```
int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

Durch die Anfrage wird der Socket–Deskriptor übertragen, über den auf die Verbindung zugegriffen werden soll, sowie die Adressstruktur des Servers. Das sind im Einzelnen die IP-Adresse des Partner–Rechners und die Portnummer des Server–Prozesses, der kontaktiert werden soll.

Im Fall einer TCP–Verbindung werden für den Aufbau der Verbindung drei Pakete benötigt, weswegen vom Drei–Wege–Handshake gesprochen wird. Abbildung 4.8 zeigt die passende Zeitschiene mit den vier dafür notwendigen Schritten. Dem eigentlichen Verbindungsaufbau vorgeschaltet ist die Definition der Sockets auf beiden Seiten.

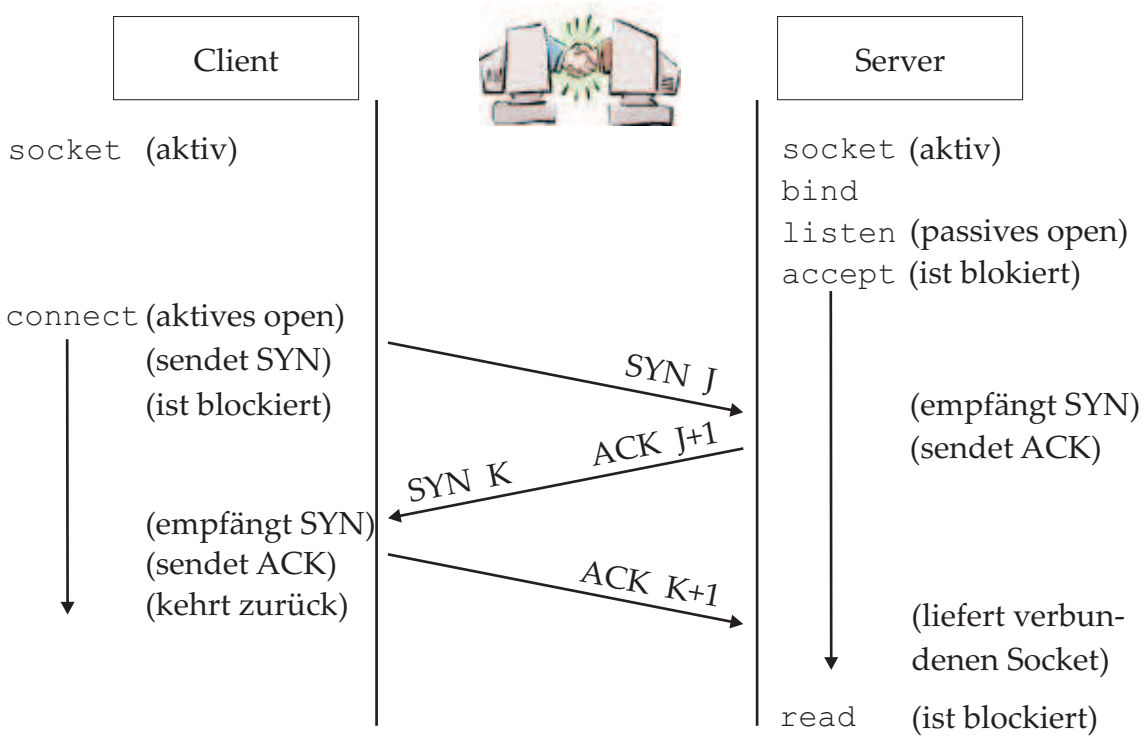


Abbildung 4.8: TCP–Verbindungsaufbau (Drei-Wege-Handshake)

Mit `connect()` wird der Client veranlasst, ein SYN–Segment zu senden (Synchronisation), um dem Server die Anfangssequenznummer des Clients für dessen Adresse zu übermitteln. Mit SYN werden ein IP-Header, ein TCP–Header und möglicherweise TCP–Optionen verschickt. Der Server muss daraufhin das SYN des Clients quittieren (ACK= acknowledge). Die Funktion `accept()` speichert dabei die IP–Adresse und die Portnummer des Clients und schickt im Gegenzug ein eigenes SYN zusammen mit der Anfangssequenznummer der Daten des Servers. Das SYN und ACK werden in einem einzigen Segment verschickt. Für jede Verbindungsanfrage eines Clients wird ein neuer verbundener Socket erzeugt, über den der aufrufende Prozess ab sofort die Datenübertragung durchführen kann. Der Client wiederum verifiziert den richtigen Empfänger durch quittieren des SYN des Servers mit einem ACK.

Datenübertragung

Die Übertragung von Daten über eine bestehende Verbindung kann sowohl vom Client als auch vom Server aus injiziert werden. In Abbildung 4.9 ist die Sequenz eines Datentransfers skizziert, wobei hier eine Anfrage des Clients vom Server bedient wird.

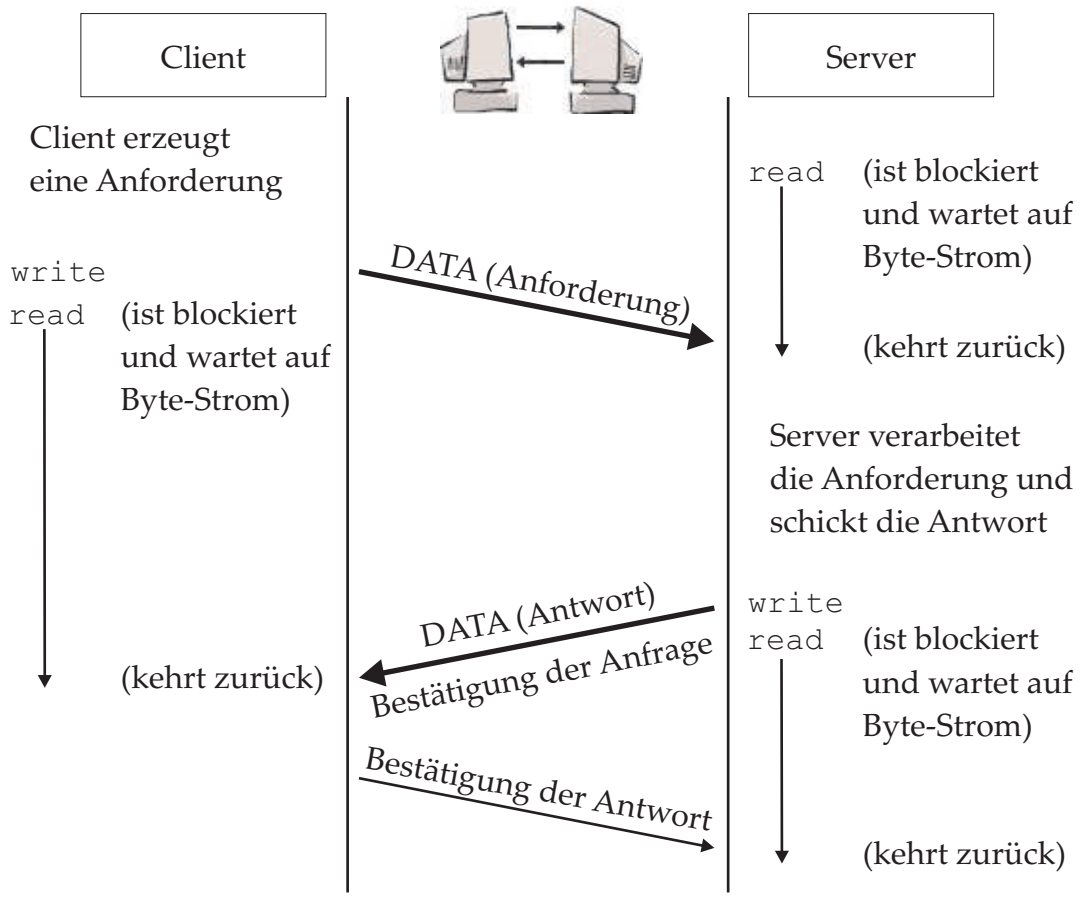


Abbildung 4.9: Datentransfer

Mit `write()` wird ein Byte–Stream verschickt, der vom Empfänger mit `read()` über den Socket empfangen werden muss.

```
int write(int sockfd, char * data, int datalen);
```

Dabei werden der Socket–Deskriptor sowie die Speicheradresse und Länge der Daten die übermittelt werden sollen übergeben. Zum Empfang der Daten wird die Funktion `read()` eingesetzt. Sie blockiert den Ablauf des Empfängerprozesses solange, bis ein Datenstrom eines berechtigten Senders eintrifft.

```
int read(int sockfd, char * buffer, int buflen);
```

Neben dem Socket–Deskriptor zur Identifikation des richtigen Byte–Streams wird die Adresse des Datenbuffers und dessen Länge angegeben, in welchem die übermittelten Daten zwischengespeichert werden. Bei Stream–Sockets, wie dem verwendeten TCP–Socket werden Daten nicht blockweise übertragen, so dass ein `read()` nicht unbedingt so viele Bytes liest, wie von `write()` versendet werden. Unter Umständen muss `read()` mehrfach aufgerufen werden, um alle Daten zu empfangen.

Das Beenden einer Verbindung

Sind alle Daten versendet, muss die TCP–Verbindung durch `close()` beendet werden, wodurch lediglich der zu terminierende Socket–Deskriptor übergeben wird.

```
int close(int sockfd);
```

Beide Kommunikationspartner können die Verbindung beenden. In Abbildung 4.10 geschieht dies beispielhaft durch die Client–Anwendung.

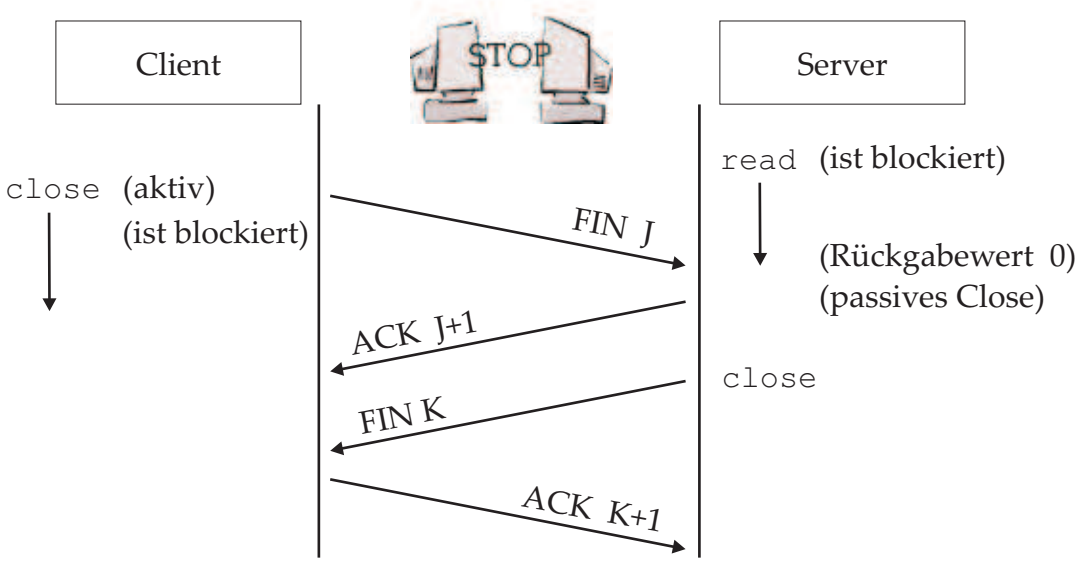


Abbildung 4.10: Schließen einer TCP-Verbindung

Für das Schließen einer TCP–Verbindung werden vier Segmente benötigt. Eine der beiden Anwendungen injiziert den Abbruch der Verbindung durch ein aktives `close()`. Ein FIN–Segment signalisiert das Ende der Verbindung. Der Empfang des FIN–Segmentes durch die `read()`–Funktion wird der anderen Anwendung bestätigt und angezeigt (EOF = end of file). Die Anwendung wird keine zusätzlichen Daten über diese Verbindung mehr empfangen, und der Socket etwas später automatisch durch ein `close()` geschlossen. Das veranlasst den Versand eines eigenen FIN–Segmentes. Das TCP der ersten Anwendung, welches diese endgültige FIN empfängt, bestätigt seinerseits dessen Ankunft, terminiert den Socket und schließt die Verbindung endgültig.

Weitere wichtige Funktionen des Application Programming Interface (API)

Allein die Funktionen zum Senden und Empfangen von Daten reichen zur Verwaltung einer Verbindung in der Regel nicht aus. Einige wichtige zusätzliche Funktionen des API sind im Anhang A aufgelistet.

4.5.7 Weitere Ansätze zur Kommunikation in verteilten Systemen

Natürlich existieren neben der nachrichtenorientierten Kommunikation auf Basis der Berkley–Sockets noch andere Techniken einer Interprozesskommunikation in verteilten Systemen.

Der Remote Procedure Call (RPC)

Birell und Nelson [Bir84] haben in einem Aufsatz 1984 eine Methode erarbeitet, entfernte Prozeduren von einem lokalen System aus aufzurufen. Die Idee hinter dem RPC beruht auf dem bekannten Client–Server–Modell. Es sollte die gemeinsame Nutzung von Programmfunktionen über Rechnergrenzen hinaus ermöglichen (vgl. Abbildung 4.11).

Der Programmcode zur Kommunikation der Prozesse (Client- und Server–Stubs) wird dabei automatisch generiert. Der Entwickler spezifiziert lediglich die durch den Client aufzurufenden Prozeduren, die im Server implementiert werden. Client und Server werden zusammen mit ihren Stubs jeweils auf ihren Rechnern übersetzt und gebunden.

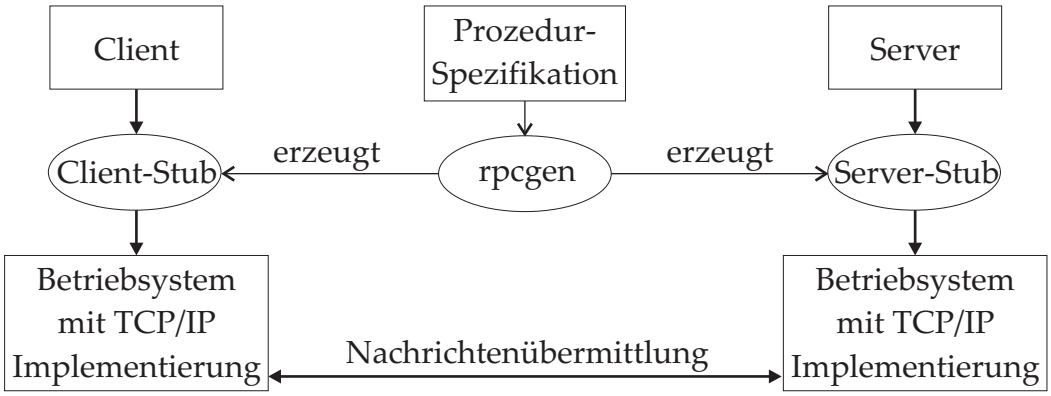


Abbildung 4.11: Client–Server–Modell mittels RPC

Nachdem beide Prozesse gestartet sind, kann der Client eine Prozedur des entfernten Servers aufrufen. Im Einzelnen laufen dabei folgende Schritte ab. Zunächst wird eine gleichlautende Prozedur des Client–Stubs auf dem selben Rechner aktiviert. Die Eingabeparameter werden konvertiert (*Marshalling*) und zusammen mit weiteren Angaben über die Prozedur als Nachricht an den Server–Stub über das Netz verschickt. Er empfängt die Botschaft, konvertiert die Übergabedaten (*Unmarshalling*) und ruft die betreffende Funktion des Servers auf. Ein RPC läuft dabei fast immer synchron ab, d.h. der aufrufende Client wartet solange mit der Ausführung des weiteren Programmcodes bis er eine Antwort der Prozedur vom Server erhalten hat. Die Antwort des Servers geschieht dabei umgekehrt analog.

Die automatisch generierten Stubs befinden sich zwischen der eigentlichen Anwendung und dem Betriebssystem, genauer auf der 5. oder 6. Schicht des ISO–OSI–Modells. Der RPC wird deswegen auch als Kommunikations–Middleware bezeichnet, der einer einfachen standardisierten Kommunikation von Komponenten in einem verteilten System dient.

Die Remote Method Invocation (RMI)

Der Methodenaufruf ist das Pendant des RPC in der objektorientierten Welt. Objekte, die sich auf entfernten Rechnern befinden, werden dabei wie lokale Objekte aufgerufen.

Aus der Spezifikation einer Schnittstelle werden ein Client–Stub und ein Server–Skeleton generiert. Nach dem Start des Servers erzeugt dieser ein Objekt mit einer speziellen Schnittstelle für die Fernnutzung und registriert dieses unter einem bestimmten Namen. Der Client sucht in der Registrierung nach dem Objekt und erhält ein Stub–Objekt, mit der selben Schnittstelle wie das Server–Objekt. Wenn der Client nun eine Methode auf dem Stub–Objekt aufruft, werden die Übergabeparameter konvertiert und eine Nachricht an das Server–Skeleton geschickt. Dort werden die Parameter in das lokale Format umgewandelt und die Methode des entsprechenden Objekts aufgerufen. Wie beim RPC geschieht die Antwort vise versa.

Mit der *Object Management Group*, einem internationales Konsortium, welches Standards für die objektorientierte Programmierung entwickelt, gibt es schon seit längerem Bestrebungen, die Kommunikation zwischen objektorientierten Programmsystemen zu standardisieren. Mit CORBA ist ein Standard definiert worden, der eine Technologie festlegt, mit der objektorientierte Systeme Objekte miteinander kommunizieren lassen können.

Weitere bekannte Vertreter für entfernte Methodenaufrufe sind Java RMI und DCOM. Diese Kommunikationstechnik findet weite Verbreitung, ist jedoch für einfache Anwendungen zu komplex und überladen. Zudem sollten dem alternativem Konzept einer Kommunikationssoftware zur Prozesskopplung keine Einschränkungen seitens einer standardisierten Schnittstelle im Wege stehen.

Deswegen wird im weiteren Verlauf eine möglichst einfache Umsetzung der beschriebenen TCP/IP–Verbindung verwendet, die leicht verständlich ist und flexibel eingesetzt werden kann.

Trotzdem ist für die zukünftige Weiterentwicklung des Konzepts die Standardisierung, zum Beispiel durch Verwendung von CORBA, in Betracht zu ziehen.

Stream–orientierte Kommunikation

Die Kommunikation über Streams ist aufgrund der Wahrung der Synchronität und Güte der Übertragung meist schwer zu implementieren und bietet sich daher nicht für eine einfache Kommunikationsschnittstelle an. Detaillierte Informationen dazu finden sich in Halsall [Hal01].

4.5.8 Bekannte Anwendungen von Netzwerk Software

Wie anfangs erwähnt, ist Software zur Netzwerkkommunikation bereits in vielen Bereichen Stand der Technik. Hier sind einige bekannte Anwendungen aufgelistet, um deren enormen Nutzen zu zeigen. Für den Bereich des Ingenieurwesens, bei dem die Anwendung von unterschiedlichen Softwareapplikationen an der Tagesordnung steht, ist großes Potenzial für neue Konzepte vorhanden.

1. **Kommunikation und Information:**
   - E–mail, Audio/Video Mail, Nachrichten, Videokonferenzen, Bildtelefon
   - Daten- und Dokumentenaustausch
   - globale Informationssysteme: WWW, digitale Bibliotheken
2. **Zusammenarbeit unter Verwendung von remote services/data, bzw. remote control:**
   - Fernunterricht, CSCW virtuelle Konferenzen
   - remote login, remote backup, distributet file systems, Client/Server und peer–to–peer computing, cluster und grid computing
   - Fernüberwachung und -kontrolle z.B. von Fabriken
3. **E–Buisness und E–Government:**
   - Telebanking, E–Shopping, Audio/Video und Software Distribution
   - Logistik Management, Kundenbeziehungsmanagement
   - Aktienhandel, Kreditkarten Transaktionen
4. **Unterhaltungsindustrie:**
   - Spiele, video on demand, interactive video

4.6 Konzept zur Kopplung verteilter Anwendungen

Hier wird die Idee aufgegriffen, bereits bestehende Software–Komponenten auf Basis eines Client–Server–Modells zu einem größeren Programmsystem zu verknüpfen. Damit sollen neue und komplexere Aufgaben bewältigt werden. Um möglichst große Flexibilität zu gewähren, sollen die beteiligten Komponenten unter Umständen beliebig im Netz verteilt sein können.

Es ist oftmals nicht möglich und meist auch nicht sinnvoll, die dafür notwendigen Schnittstellen direkt in den Quellcode der bestehenden Software zu integrieren. Eine umständliche Anpassung in verschiedenen Programmiersprachen mit den jeweiligen Compilern und Linkern auf unterschiedlichen Rechnerarchitekturen wäre notwendig. Gelegentlich verhindern auch rechtliche Gründe den Zugriff auf benötigten Quellcode.

An dieser Stelle entsteht das Konzept einer allgemeinen Kommunikationsschnittstelle, die für sich betrachtet leicht auf die unterschiedlichsten Gegebenheiten angepasst werden kann. Trotzdem soll die Schnittstelle klein, logisch strukturiert und verständlich sein. Das kann durch ein vorgefertigtes Kommunikationsgerüst umgesetzt werden mit der Möglichkeit, beliebige Datenstrukturen zu verschicken. Damit kann sich der Anwender wie bei Verwendung von Bibliotheken auf die eigentliche Problematik konzentrieren. Overhead wird vermieden und die Performance bei der Entwicklung gesteigert. Auf dieser Basis müssen sich Programmentwickler dann nur noch auf die Datenstruktur der Schnittstelle einigen, diese dann anpassen und für sich betrachtet an ihre Programmpakete anbinden.

Die Kommunikation an sich erfolgt durch die in Kapitel 4.5 beschriebene Client–Server–Beziehung auf Basis von Sockets, wie Abbildung 4.12 nochmal verdeutlicht. Der markierte Bereich ist hier problemabhängig anzupassen.

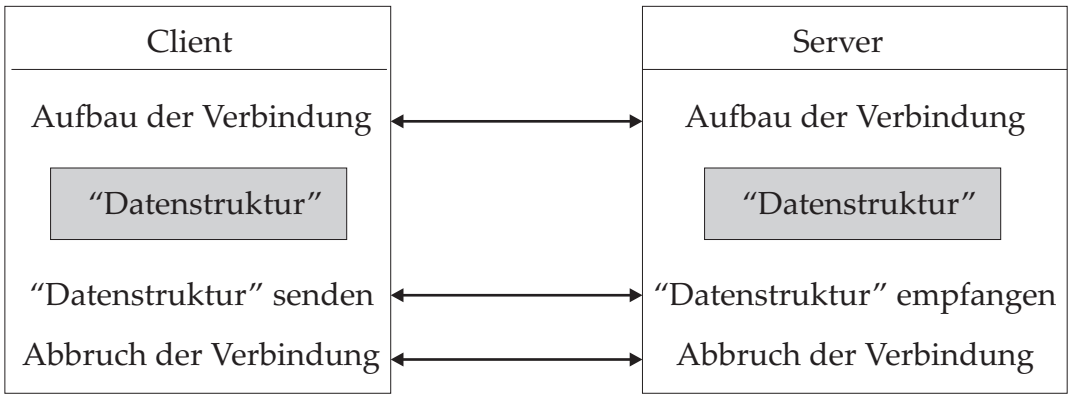


Abbildung 4.12: Grundgerüst einer Client–Server–Beziehung

Der komplette Ablauf einer Datenübertragung zwischen Client und Server ist in Abbildung 4.13 in Form von Pseudocode dargestellt.

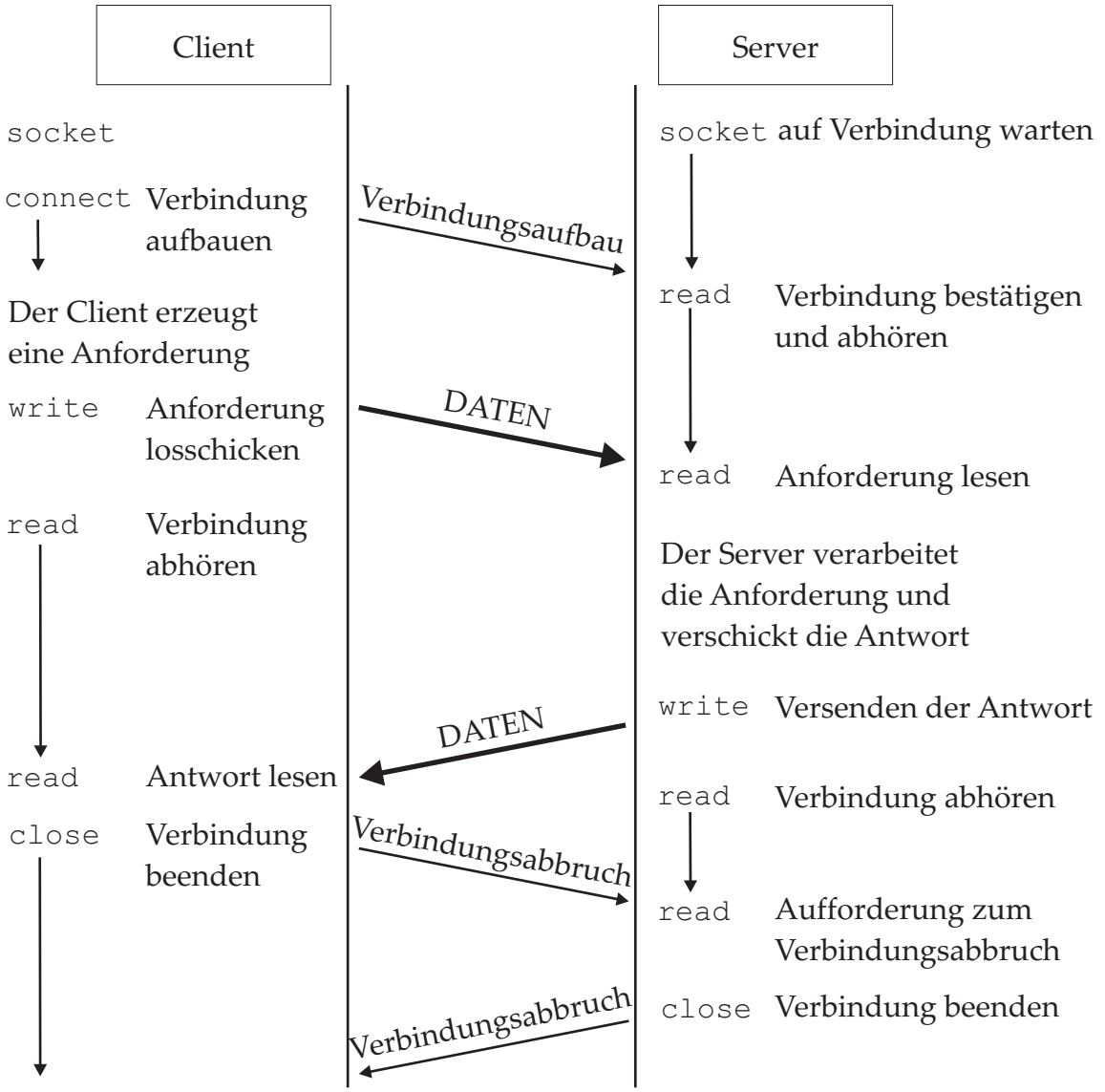


Abbildung 4.13: Ablauf der Datenübertragung

Die Anbindung der Kommunikationsschnittstelle wird dann mit Hilfe der in Kapitel 2.5 benannten Methode der Prozesskapselung umgesetzt. Dabei wird die eigentliche Anwendung mit einer klar definierten Schnittstelle versehen und zusammen mit dem Kommunikationsmodul nach außen hin abgekapselt, wie Abbildung 4.14 verdeutlicht. Die Kapsel stellt hier die Verbindungsschicht zwischen dem Kommunikationssystem und dem gekapselten Prozess dar, und übernimmt wesentliche Aufgaben. Durch sie wird der eigentliche Prozess aufgerufen, Daten werden übernommen und für den Versand im Bedarfsfall aufbereitet. Zu dem beinhaltet die Kapsel das Kommunikationsmodul in Form eines Servers bzw. eines Clients.

Die Kapselung der Prozesse an sich bedarf wenig Anstrengung und kann von den jeweiligen Programmierern in deren Entwicklungsumgebung selbständig und unabhängig unternommen werden.

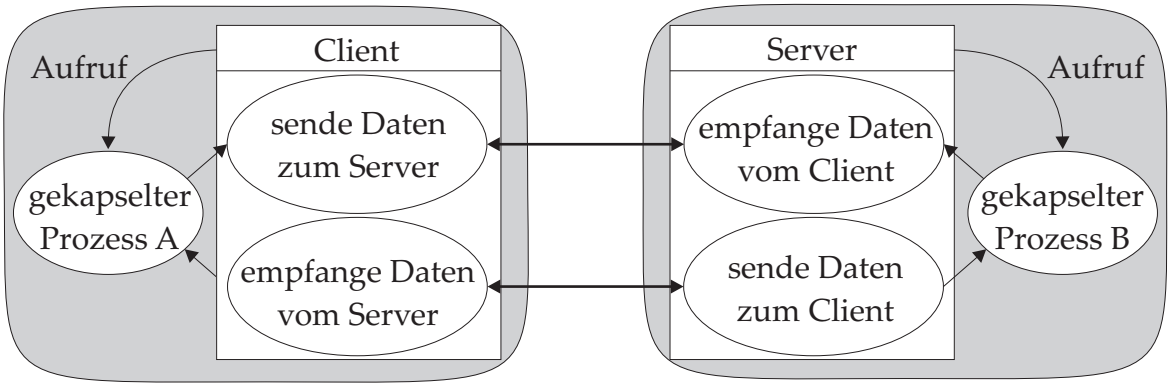


Abbildung 4.14: Client–Server Kommunikationsmodell mit gekapselten Prozessen

Im Klartext bedeutet das, dass die Kommunikationsschnittstelle von der Anwendung als Unterroutine aufgerufen wird. In manchen Fällen agiert sie selbst als Hauptprogramm und ruft die Anwendung als Unterroutine auf. Dabei ist unter Umständen die Herausforderung zu bewältigen, unterschiedliche Programmiersprachen miteinander zu verknüpfen. Mehr über das Gebiet des sogenannten *Mixed Language Programming* (MLP) findet sich in Kapitel 2.4 und im Anhang B.

Bei der Kopplung koexistenter Prozesse ist die Ablaufintegrität durch eine sinnvolle Steuerung der beteiligten Prozess zu gewährleisten (s. Kapitel 4.4.2). Bei interaktiv kooperierenden Prozessen müssen an klar definierten Stellen im Programmablauf Daten und/oder Nachrichten versendet und empfangen werden. Für den Anwender ist es praktikabel, den Ablauf von einer zentralen Steuereinheit überwachen und gegebenenfalls steuern zu können - am besten graphisch interaktiv.

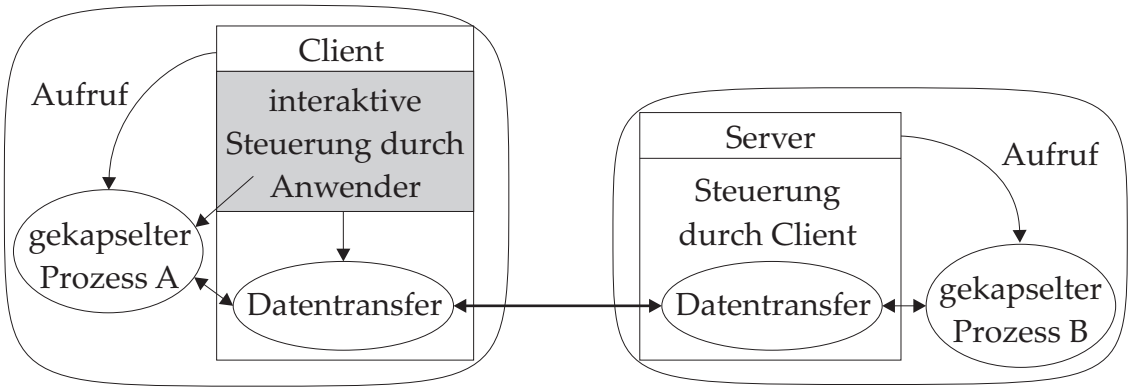


Abbildung 4.15: Steuerung der kooperierenden Prozesse durch den Client

Dafür gibt es unterschiedliche Ansätze. Wenn lediglich zwei Prozesse beteiligt sind, ist die Steuerung vom Client oder vom Server aus denkbar und sinnvoll. In Abbildung 4.15 übernimmt dies der Client. Gleichermaßen kann die Steuerung auch von einem außen stehenden Prozess mit eigener graphischer Oberfläche geregelt werden, wie Abbildung 4.16 darstellt. Dies bietet sich vor allem bei mehr als zwei beteiligten Modulen an, oder wenn keine der beteiligten Anwendungen eine geeignete, graphisch unterstützte Steuerung bietet.

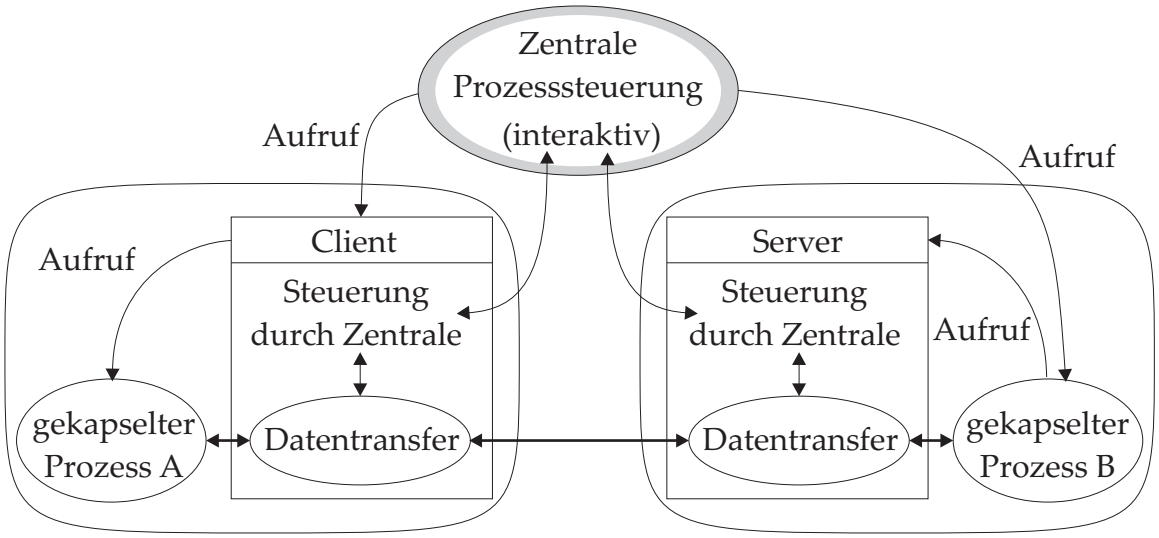


Abbildung 4.16: Steuerung der kooperierenden Prozesse durch eine zentrale Stelle

In obiger Abbildung ist links der gekapselte Prozess als Unterroutinen des Kommunikationsmoduls dargestellt und wird von diesem zu gegebener Zeit aufgerufen. Der rechte Prozess wird direkt von der zentralen Steuereinheit aufgerufen. Hier ist das Kommunikationsmodul eine Unterroutine der eigentlichen Anwendung.

Mit einem derartigen Modell ist die Vorgabe erfüllt, für eine spezielle Aufgabe möglichst effizient ein neues Softwaresystem zu erzeugen. Durch die Kapselung von bereits bestehenden Modulen oder Prozessen ist eine kurze Entwicklungszeit garantiert. Zudem trägt die Verwendung von ausgereiften Komponenten zur Qualität der Gesamtlösung bei. Durch den objektorientierten Aufbau ist gleichzeitig eine hohe Modularität gewährleistet. Komponenten können jederzeit ergänzt oder ersetzt werden. Dies bedarf jedoch immer einen Eingriff in das Programmsystem mit den dafür nötigen Programmierkenntnissen.

4.7 Das Workflow–Management–System

Die Aufgabe eines Workflow–Management–Systems besteht in der aktiven Steuerung arbeitsteiliger Prozesse, d.h. es dient der Koordination, *wer* (Prozess) *was* (Berechnung) *wann* (Ablauf) und *wie* (Methoden) bearbeitet. Es unterstützt insbesondere strukturierte Aufgaben. Ziel hierbei ist die Automatisierung der Ausführung.

Arbeitsablaufmodelle sollen helfen, die optimale Einbindung verschiedenster Anwendungen in die jeweiligen Arbeitsabläufe sicherzustellen. Damit können komplexere Prozesse vereinheitlicht und deren Qualität verbessert werden. Die Bearbeitungszeit wird durch einen kontrollierten Fluss der essentiellen Daten zwischen Prozessen verringert. Als weiteres wird die Flexibilität von Prozessen erhöht, womit Ressourcen besser genutzt werden können.

Der Begriff Workflow–Management ist eine praktische Anwendung der Netzplantechnik und wird im eigentlichen Sinn für rechnergestützte Gruppen- oder Zusammenarbeit benötigt. Das Workflow–Management umfasst alle Aufgaben, die bei der Modellierung, Spezifikation, Simulation sowie bei der Ausführung und Steuerung von Arbeitsabläufen (engl. workflows) erfüllt werden müssen.

Im übertragenen Sinn kann ein Computerprogramm, durch welches eine individuelle Zusammenstellung von Prozessen und Modulen geplant und gesteuert werden kann, als Workflow–Management–System bezeichnet werden.

Das Ziel weiterer Überlegungen ist ein System bereitzustellen, welches einzelne Objekte in Form von Softwaremodulen dynamisch miteinander verbinden zu können, ohne in den Programmcode eingreifen zu müssen. Der Anwender soll nicht mit programmiertechnischen Details konfrontiert werden, wenn er sich sein Softwarepaket individuell zusammen stellt. Dafür müssen jedoch geeignete Module mit passenden Schnittstellen bereitgestellt werden, die aufeinander abgestimmt sind. Sie dürfen zudem nicht mehr fest in einem Softwaresystem miteinander verknüpft sein.

Werden die einzelnen Module nach dem Konzept von Abschnitt 4.6 miteinander gekoppelt, eröffnet das Workflow–Management–System beinahe uneingeschränkte Möglichkeiten.

- Module sind jeder Zeit schnell und ohne Aufwand durch andere ersetzbar.
- Es besteht die Möglichkeit auf Module entfernter Rechner zuzugreifen. Der große Vorteil liegt darin, die einzelnen Komponenten auf beliebige aber dafür speziell vorgesehene Rechner oder Rechnersysteme zu verteilen.
- Andererseits können bestehende Module, die auf spezielle Rechnersysteme hin optimiert sind, miteinander verknüpft werden, ohne sie umständlich und zeitaufwendig portieren zu müssen.

- Berechnungsingenieure könnten von unterschiedlichen Orten aus gleichzeitig auf Rechenergebnisse zugreifen, die ein Beteiligter gerade erzeugt.
- Wenn lediglich die Module geladen werden, die zur Lösung benötigt werden, kann bei großen Programmsystemen ein gewaltiger Overhead vermieden werden.

In Abbildung 4.17 ist das Modell eines derartigen Workflow–Management–Systems dargestellt. Durch „Drag & Drop“ werden verfügbare Prozesse und Module in einem *Workflow* arrangiert und damit der zeitliche Ablauf geregelt. Verbindungslinien geben die Beziehungen und Datenströme untereinander wieder. Ein Prozess bezeichnet hier eine Softwareapplikation auf einem beliebigen Rechner. Unter dem Begriff Modul wird hier eine Zusammenstellung von Methoden des Workflow–Management–Systems verstanden (vgl. Abbildung 4.18).

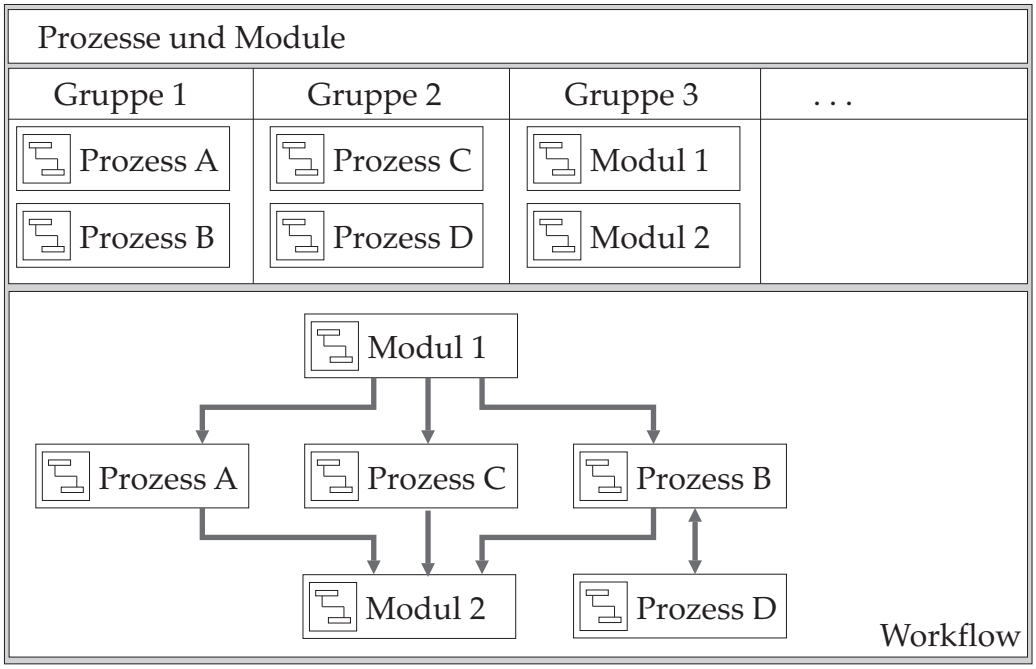


Abbildung 4.17: Schematische Darstellung eines Workflow–Management–Systems

Beispielhaft ist hier die Verknüpfung von vier *Prozessen* auf zwei unterschiedlichen Rechnern dargestellt. Das Workflow–Management–System auf dem dritten Rechner dient dem Anwender zur Planung, welche Prozesse wie miteinander verknüpft werden (hier über *Modul 2*) und der späteren Ablaufsteuerung (*Modul 1*). Dem Endanwender präsentiert sich der erstellte Workflow als ein zusammenhängendes Gesamtpaket, dessen Bedienung er von einer zentralen Steuerung aus übernehmen kann.

Unter Einhaltung zu definierender Regeln können beliebige Prozesse und Module ergänzt oder ausgetauscht werden. An dieser Stelle sind Entwickler in der Lage

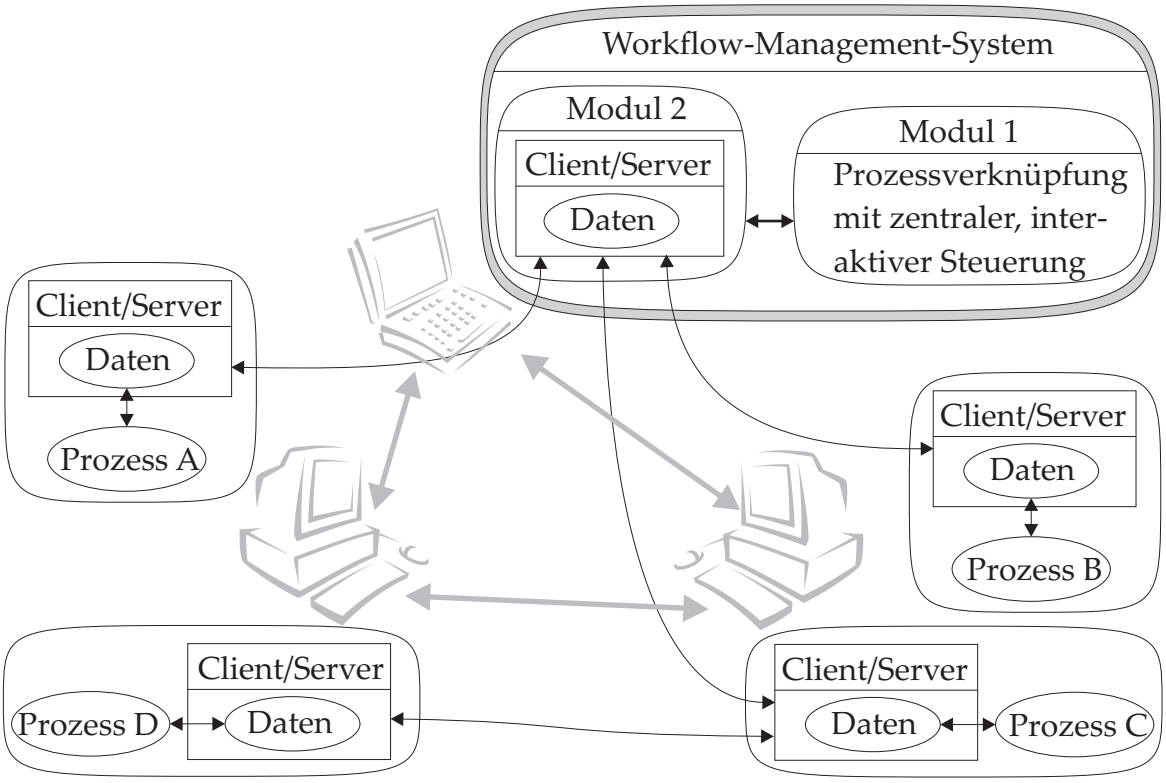


Abbildung 4.18: Tatsächliche Beziehungen der Anwendung

schneller auf die Bedürfnisse der Anwender reagieren zu können, da sie meist „nur“ ein bestehendes Modul anpassen und hinzufügen müssen.

Abbildung 4.18 verdeutlicht die tatsächlichen Beziehungen der Prozesse und Module der im Workflow aus Bild 4.17 gezeigten Anwendung untereinander und die Verteilung auf den drei unterschiedlichen Rechnern.

Kapitel 5

Prozesssteuerung und Visualisierung der Stabwerksermittlung

Bei der Entwicklung verteilter Anwendungen hat die Synchronisierung und Steuerung der einzelnen Prozesse höchste Priorität. Wird ein Workflow–Management–System zur Prozesssteuerung eingesetzt, ist eine graphische Unterstützung für den Anwender unverzichtbar. Für den Entwickler ist eine graphisch–interaktive Entwicklungsumgebung für die Programmierung und Erweiterung des Workflow–Management–Systems von Vorteil und damit ein wichtiges Argument bei der Wahl der Software–Umgebung.

Bei der Verwendung von Simulationssoftware ist für den Anwender die Auswertung von Berechnungsergebnissen komplexer Systeme besonders wichtig. Die schnelle und richtige Deutung immenser Datenmengen ist meist nur durch deren graphischen Darstellung möglich.

In kommerziellen Anwendungen ist dieser Aspekt schon lange zu einem hohen Standard gereift. In der Forschung und Entwicklung entsteht jedoch immer wieder die Herausforderung, Neuentwicklungen schnellstmöglich in Programme zu integrieren und mit geringstem Aufwand zu visualisieren. Das bedeutet im Allgemeinen, abstrakte Daten in eine angebrachte, verständliche Form zu bringen. Dabei müssen Datensätze derart gefiltert werden, dass sie trotzdem korrekt interpretierbar bleiben.

So war im Zusammenhang mit dieser Arbeit eine komplette Visualisierung für ein Finite–Elemente–Programm (CARAT) zu generieren. Durch die Portierung des Programmcodes auf ein LINUX Betriebsysstem konnten die verwendeten Graphik–Bibliotheken von HP–UX nicht weiter genutzt werden.

Um den Entwicklungsaufwand möglichst gering zu halten und damit den Fokus auf das eigentliche Kernproblem der wissenschaftlichen Arbeit verlagern zu können, werden von Software–Herstellern mächtige Programmpakete zur Verfügung gestellt.

5.1 Beschreibung von AVS/Express

AVS/Express von der Firma Advanced Visual Systems Inc. stellt eine Lösung zur High–End–Visualisierung von wissenschaftlichen und technischen Daten dar. Es ist ein Entwicklungs- und Anwendungssystem zur Visualisierung großer, komplex strukturierter Datenmengen, und stellt eine Vielzahl vorgefertigter Module zu deren Verarbeitung, Aufbereitung und Darstellung zur Verfügung. Zudem werden in zahlreichen Anwenderpools ständig neue Entwicklungen angeboten. Klassische Einsatzbereiche sind die Gebiete der Struktur- und Fluidmechanik. Darin wird ganz besonders die Darstellung dynamischer Prozesse unterstützt. Aber auch andere Disziplinen, wie die Medizin, die Geowissenschaften usw. profitieren vom großen Potential der Visualisierungsmöglichkeiten.

Dabei ist AVS/Express eher als eine Programmierumgebung mit dem Schwerpunkt der Aufarbeitung und Visualisierung von Daten zu sehen. Durch die visuelle, objektorientierte Entwicklungsumgebung können auf einfache Weise flexible interaktive Grafik- und Visualisierungsanwendungen geschaffen werden. Die visuelle und auf Komponenten basierende Programmierung bietet eine Reihe von Vorteilen. Programmierfehler werden weitgehend ausgeschlossen, da nur kompatible Objekte miteinander verknüpft werden können und sofort aktiv sind. Der Zeitaufwand gegenüber einer herkömmlichen Programmierung wird deutlich reduziert, da bereits viele vorgefertigte Module existieren, auf die zugegriffen werden kann. Durch eine gut gegliederte, hierarchische und visuelle Entwicklung kann die gedankliche Vorstellung des Programmierers von der Programmstruktur repräsentiert werden, was die Programmgestaltung deutlich erleichtert.

Der modulare Aufbau erleichtert die Generierung so genannter *Netzwerke* (Anwendungen), die aus der Verknüpfung von Modulen und Makros entstehen. Ein Modul ist als Objekt zu verstehen, welches Daten empfängt, verarbeitet und zur weiteren Verarbeitung zur Verfügung stellt. Durch die Verwendung von Makros, einer sinnvollen Gruppierung von Modulen, lässt sich die Anwendung in einer hierarchischen Netzwerkstruktur aufbauen (vgl. Abbildung 5.1).

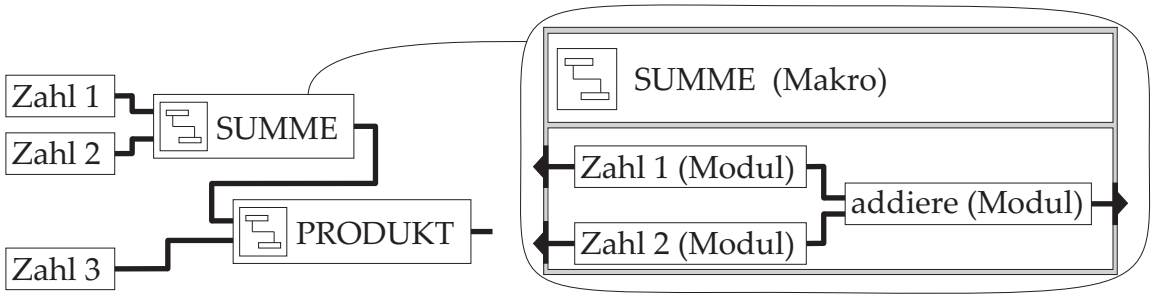


Abbildung 5.1: Gruppierung von Modulen zu Makros

Zudem kann die vorhandene Komponentenbibliothek durch Einbindung eigenen Quellcodes (Fortran, C, C++) individuell erweitert werden. Durch die direkte Zugriffsmöglichkeit auf die Datenhaltung steht einer Prozesskopplung und der Einbindung in ein objektorientiertes Programmsystem nichts mehr im Weg. Mit der Anbindungsmöglichkeit ist ein interaktiver Datenaustausch mit anderen Prozessen in einem verteilten System während ihrer Laufzeit möglich. Gerade bei der Fluid–Struktur–Interaktion oder in der Optimierung entfällt damit das lästige Schreiben, Versenden und Lesen riesiger Datenmengen über einen Festplattenspeicher. An deren Stelle tritt der gezielte Austausch der essentiellen Daten direkt von einem Arbeitsspeicher in den anderen. In verteilten Systemen wird zudem die Netzbelastung deutlich verringert.

Der Entwickler kann ein Sammelsurium an Komponenten für den Anwender zusammenstellen, die beliebig erweiterbar oder austauschbar sind. Nach bestimmten Vorschriften können diese zu spezialisierten Anwendungen verbunden werden und damit individuelle Probleme lösen (siehe Abbildung 5.2).

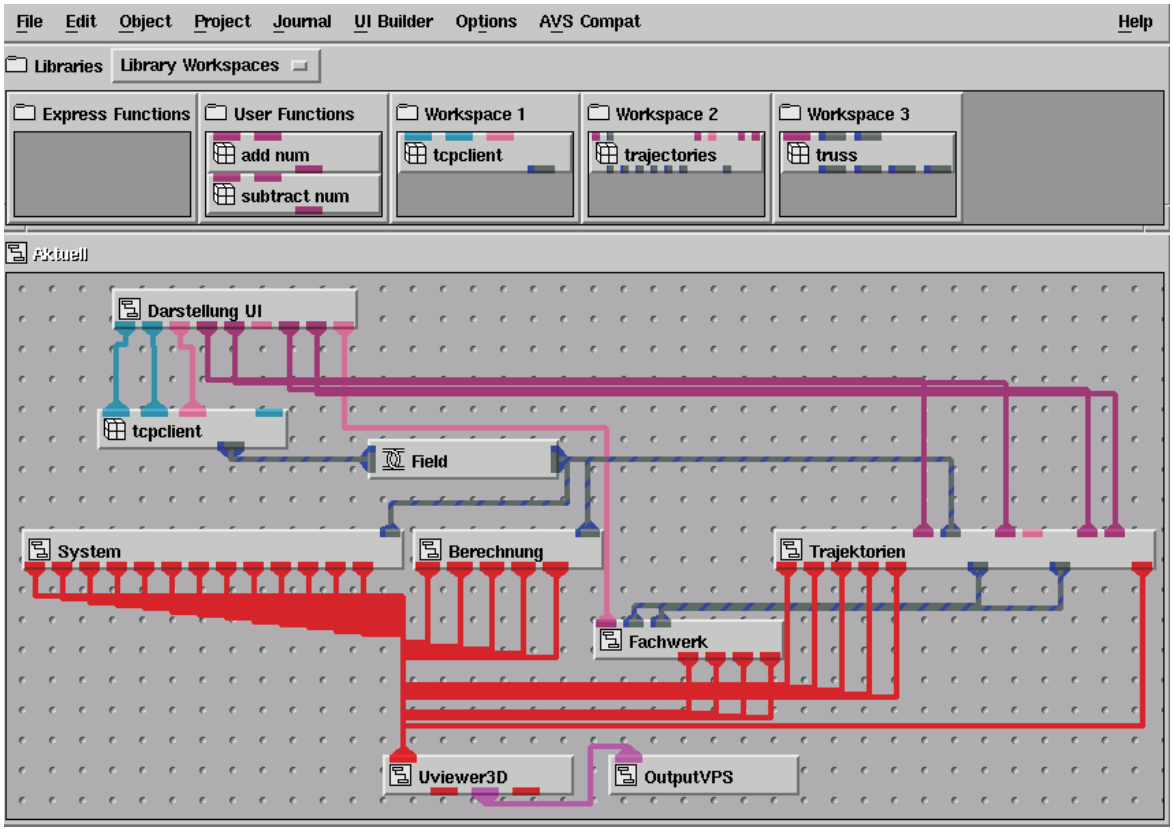


Abbildung 5.2: AVS/Express als Workflow–Manager (Netzwerk)

Das Bild zeigt die Umsetzung der in Abschnitt 3.4 beschriebenen Generierung optimierter Stabwerkmodelle auf der Basis von Kraftflusspfaden. Die Rechtecke symbolisieren Makros, Module und ganze Prozesse, während die Verbindungslinien die

Datenströme darstellen. Beispielhaft ist hier lediglich die oberste Hierarchieebene der Anwendung bzw. des Netzwerks (in AVS/Express) abgebildet.

Das damit erstellte Darstellungsfenster und grafische Benutzerinterface ist in Abbildung 5.3 zu sehen.

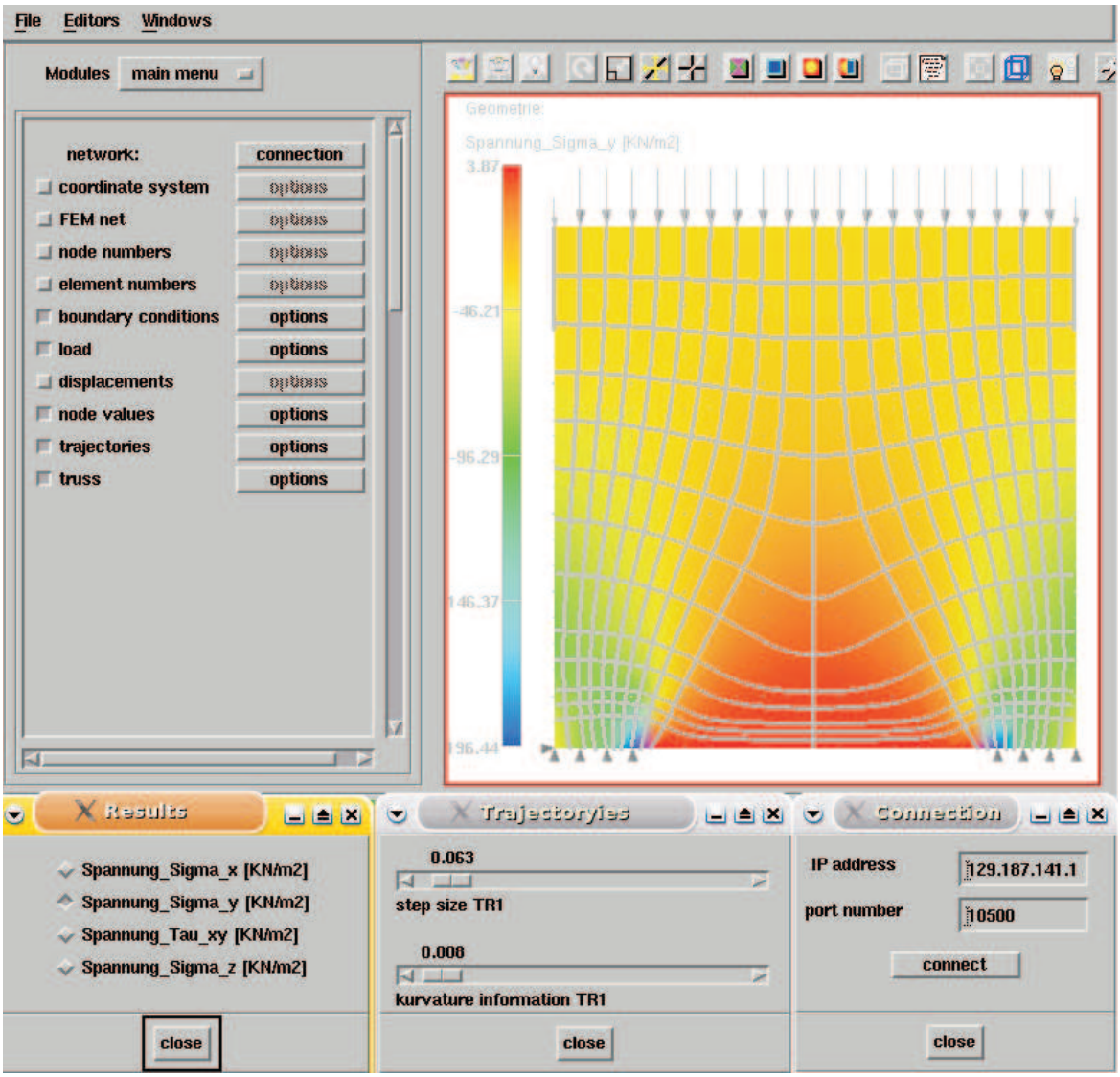


Abbildung 5.3: Darstellungsfenster mit Optionen

Im sogenannten Darstellungsfenster sind die bereits aufgearbeiteten Daten zu sehen. Am Beispiel einer Scheibenberechnung sind das im Einzelnen die Geometrie und Randbedingungen der Problemstellung, sowie die Ergebnisse der Finite–Elemente–Berechnung. Während die Verformungen lediglich zur Kontrolle der Eingabe dienen, werden die Spannungsinformationen zur weiteren Ermittlung des Kraftflusses benötigt.

Das Benutzerinterface stellt dem Anwender standardmäßig zahlreiche Möglichkeiten der Datenmanipulation, wie drehen, skalieren, etc. zum besseren Verständnis

zur Verfügung. Vielseitige weitere Optionen, welche den Ablauf des Programmsystems interaktiv steuern, sind in der linken Spalte und am unteren Bildrand dargestellt. Mit Dreh- und Schiebereglern, Schaltknöpfen, Eingabefeldern usw. ist die gesamte Palette für ein interaktives Arbeiten vorhanden. Hier werden die wichtige Parameter zur Erzeugung von Trajektorien und Verbindungsdaten zu den beteiligten Prozessen abgefragt.

Für eine detailliertere Beschreibung von AVS/Express und weiterführende Hilfen sei auf die Online–Dokumentation oder folgende Web–Adressen hingewiesen: `http://www.avs.com`, `http://www.iavsc.org` und `http://www.uni-ulm.de/gaug`.

5.2 Die Anwendung zur Erzeugung optimierter Stabwerkmodelle

Für die Ermittlung und Visualisierung von Kraftflusspfaden sind unterschiedliche Applikationen notwendig. Im Wesentlichen sind das ein FE–Programm (CARAT), ein Algorithmus zur Erzeugung von Kraftflusspfaden (ALFIT), ein Fachwerkprogramm (TRUSS) und das Visualisierungstool (AVS/Express). Das in Abschnitt 4.6 vorgestellte Konzept zur Interprozesskommunikation wird dafür verwendet, die einzelnen Prozesse zu Kapseln und über ein Netzwerk miteinander zu einem kohärenten Programmsystem zu Verknüpfen.

Für den Anwender von zentraler Bedeutung ist die visuelle Darstellung des eigentlichen Problems und dessen Lösung. Gerade wegen der Bedeutung von AVS/Express als visuelle Schnittstelle zwischen Anwender und den Rechenroutinen, aber auch aufgrund der vorher genannten Fähigkeiten dient AVS/Express zusätzlich als zentrale Steuereinheit für alle beteiligten Prozesse.

Als solche muss die Anwendung nicht von anderen aufgerufen werden, weswegen auch keine Kapselung notwendig ist. In Abbildung 5.4 werden der Aufbau der Anwendung in AVS/Express und die Funktionen der einzelnen Module ersichtlich. Hier sind vier Bereiche zu unterscheiden.

Als zentrale Einheit übernimmt das Visualisierungstool die **Prozesssteuerung**. Dies geschieht zwar in einer vorgegebenen Reihenfolge, jedoch nicht im Bachmodus. Durch Initiieren von Prozessen in der richtigen Abfolge obliegt dem Anwender die interaktive Steuerung.

Die **Kommunikation** an sich mit dem Datenaustausch zwischen der lokalen und der entfernten Datenhaltung übernehmen die entfernten Prozesse CARAT, ALFIT und TRUSS im verteilten System selbst, ganz im Sinne eines objektorientierten Programmsystems. Speziell an die jeweiligen Aufgaben angepasste AVS–Clients verbinden sich auf Anfrage des Anwenders mit dem jeweiligen Server. Das ist bei-

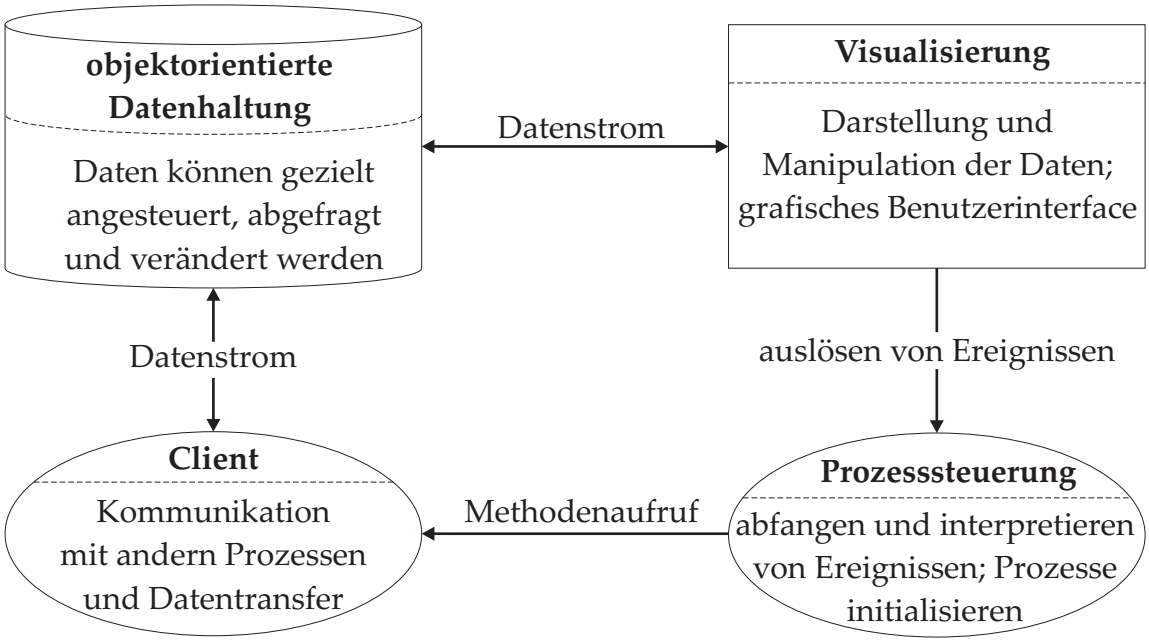


Abbildung 5.4: Die Gliederung der Anwendung

spielsweise der CARAT–Server, der mit dem Berechnungsmodul (CARAT) zusammen gekapselt ist. Durch den AVS–Client werden hierbei alle erforderlichen Geometriedaten und Berechnungsergebnisse vom CARAT–Server angefordert, der diese dann versendet.

Der AVS–Client empfängt die Daten und leitet sie an die objektorientierte **Datenhaltung** von AVS/Express weiter, wo sie für die Weiterverarbeitung zur Verfügung gestellt werden. Eine anschließende Filterung und Aufarbeitung der Rohdaten, ermöglicht die **Darstellung** am Bildschirm. Hier befindet sich gleichzeitig das Benutzerinterface (vgl. Abbildung 5.2).

Wie bei einem objektorientierten Programmsystem ist der Programmablauf durch eine Anzahl von miteinander interagierenden Objekte charakterisiert, die sich Nachrichten zusenden. Die Empfänger einer Nachricht reagieren durch die Ausführung von Methoden.

Abbildung 5.5 zeigt die Einbindung von AVS/Express im Kontext des Gesamtkonzepts. Alle anderen beteiligten Komponenten werden als gekapselte Objekte im Sinne eines objektorientierten Programmsystems mit eingebunden.

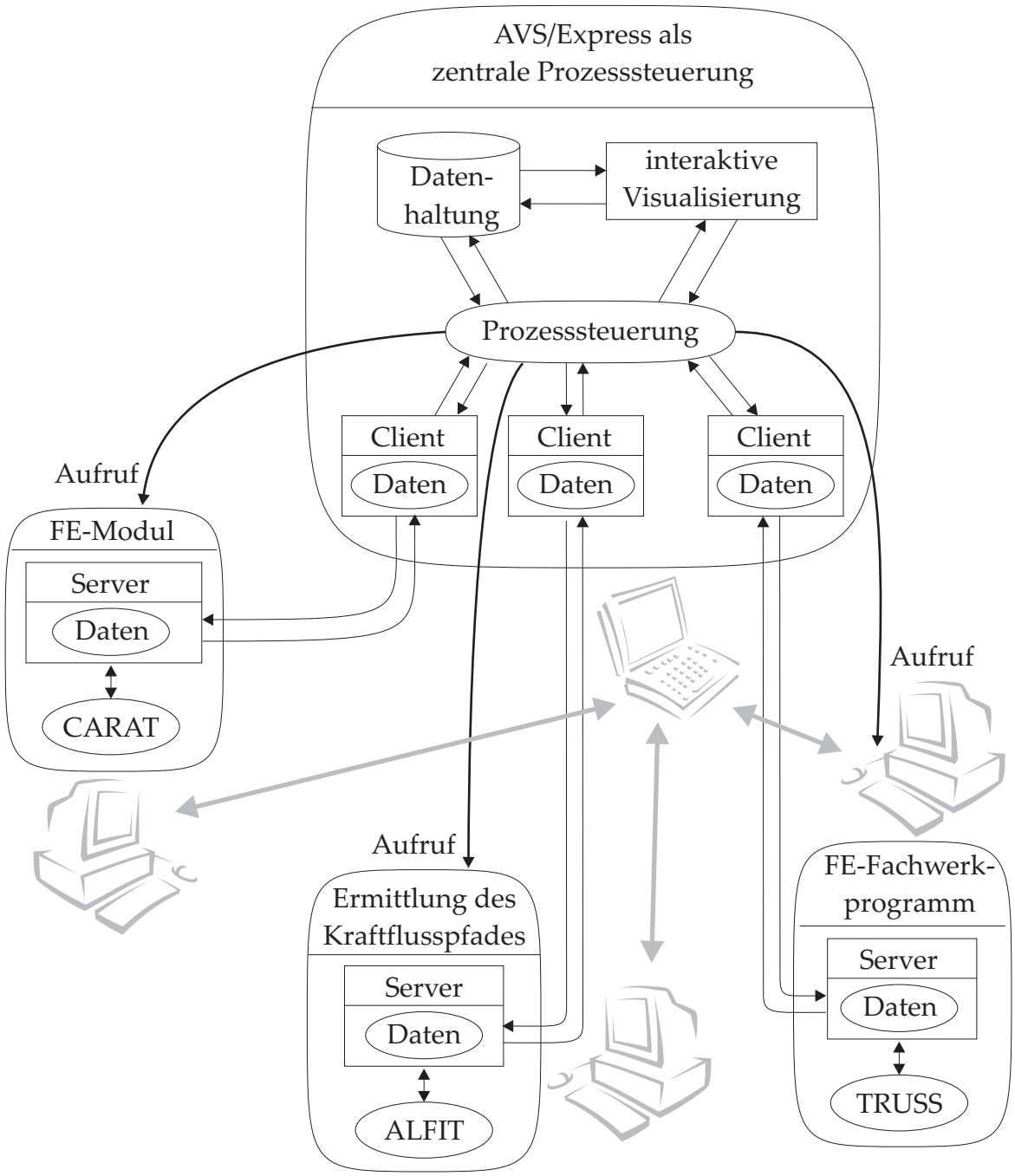


Abbildung 5.5: Die Eingliederung der Anwendung in das Gesamtkonzept

5.3 Ablauf und Steuerung einer Kraftflussberechnung

Zunächst wird die Finite–Elemente–Berechnung (CARAT) des betreffenden Scheibensystems gestartet und die Ergebnisse vom CARAT–Server dem AVS–Client zur Verfügung gestellt. Dieser bedient sich der notwendigen Daten für die Darstellung der Geometrie sowie der relevanten Ergebnisse (vgl. Bild 5.6).

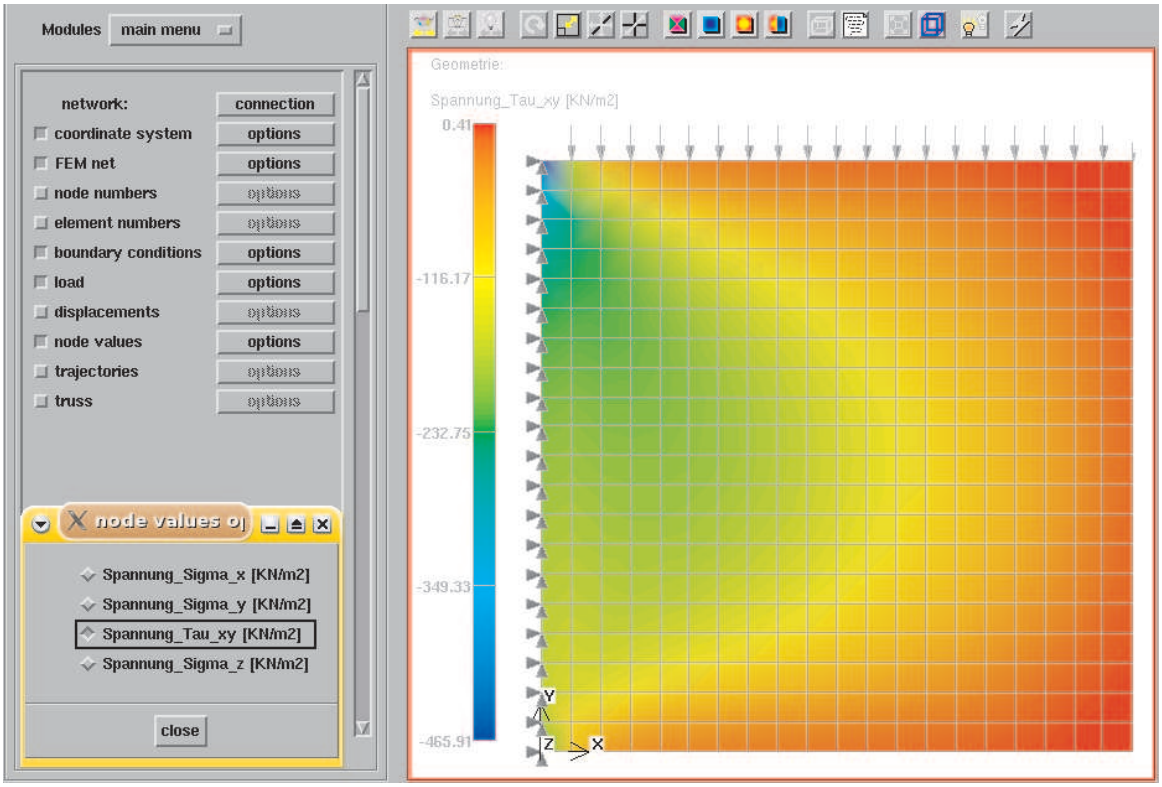


Abbildung 5.6: Darstellung der Spannungsauswertung von CARAT

An dieser Stelle hat der Anwender die Möglichkeit die Berechnung zu kontrollieren und gegebenenfalls die Modellierung abzuändern.
Dafür könne die notwendigen Informationen abgerufen und dargestellt werden. In Abbildung 5.7 ist beispielsweise das FE–Netz in unverformter und deformierter Lage zu sehen. Knoten und Elementnummern sowie Randbedingungen und Lasten können abgerufen werden. Rechts im Bild ist ein Ausschnitt mit Trajektorien zu sehen.

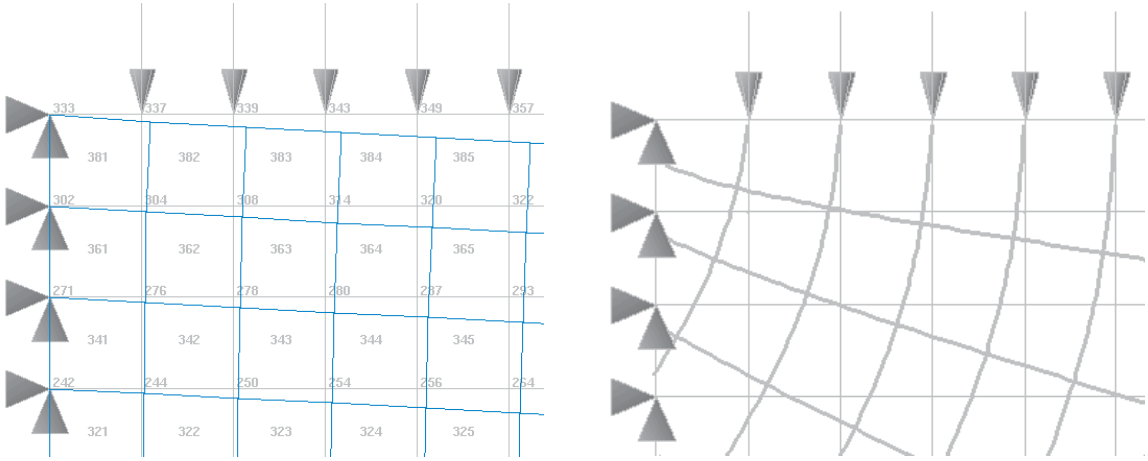


Abbildung 5.7: Netzinformationen mit Verformung (links) und Trajektorien (rechts)

Im nächsten Schritt wird der Algorithmus zur Bestimmung von Kraftflusspfaden (ALFIT) angestoßen. Dessen Server bekommt vom AVS–Client die dafür notwendigen interaktiven Steuerparameter, Spannungswerte und weitere Daten übermittelt. Die Ergebnisse werden vom ALFIT–Server zurückgesendet und ad hoc von AVS/Express dargestellt. Wieder ist ein steuernder Eingriff des Anwenders möglich, indem er die Parameter für den Algorithmus anpasst, um eine alternative Diskretisierung der sekundären Kraftflusspfade zu initialisieren. In Abbildung 5.8 ist der Vorgang der Ermittlung von Kraftflusspfaden (s. Abschnitt 3.4) bis hin zum Stabwerk nochmal bildhaft aufgezeigt.

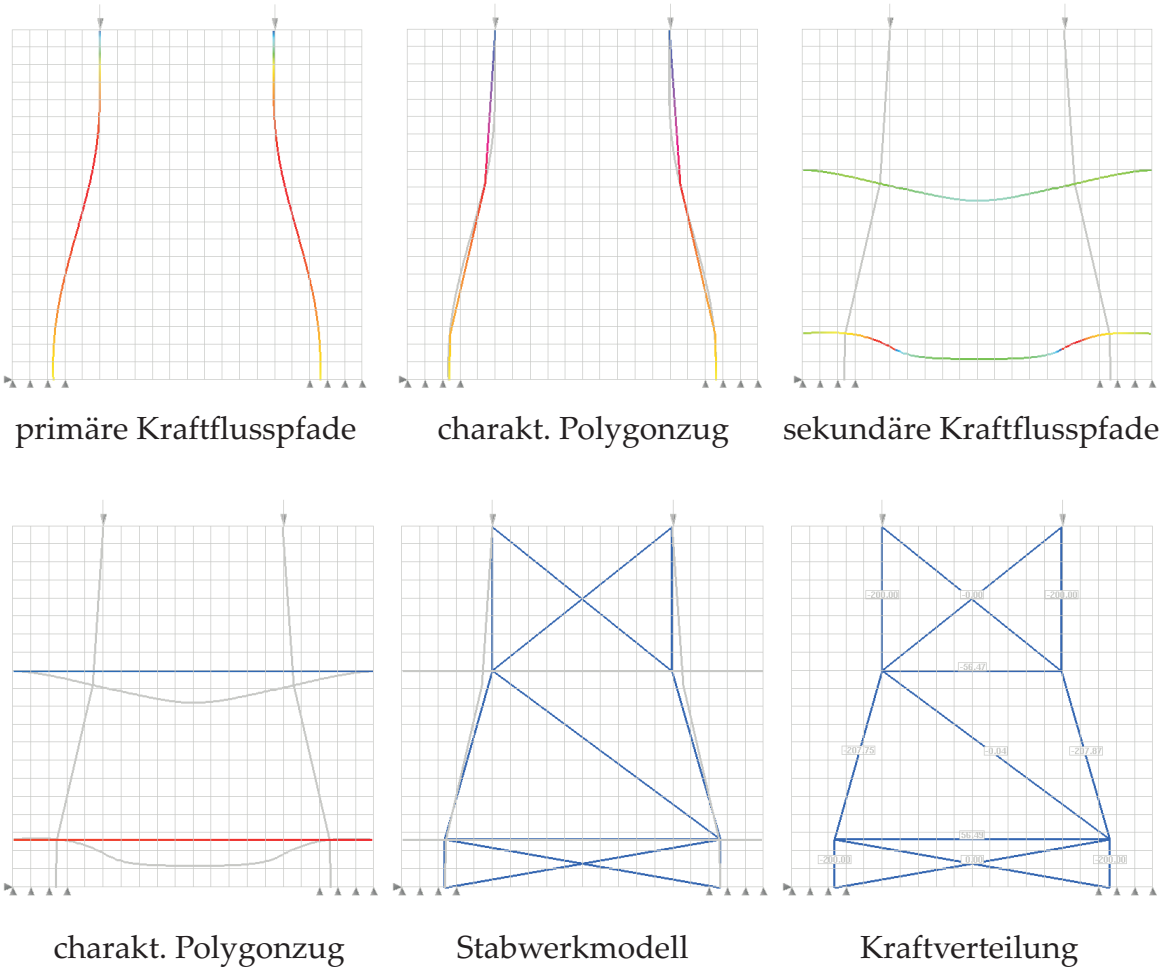


Abbildung 5.8: Ablauf bei der Ermittlung der Kraftflusspfade und des Stabwerkmodells

Ist die Modellierung der sekundären Kraftflusspfade zufriedenstellend, wird daraus ein kinematisches Stabwerk auf der Basis der in Kapitel 3 beschriebenen Methode generiert (ALFIT). Intern werden stabilisierenden Stäben mit geeignet angepassten Steifigkeitsverhältnissen hinzugefügt. Das derart entstandene Fachwerk kann von jedem beliebigem Fachwerkprogramm berechnet werden.

Für die behandelten numerischen Beispiele ergibt sich hieraus kein Einfluss auf die Tragwirkung des Stabwerkes. Das führt auf zum Schluss, dass sich das Stabwerk

durch die Lage seiner Knotenpunkte und die Randbedingungen bereits nahezu im Gleichgewicht befindet (siehe dazu Kapitel 3.4 und 6). Das Hinzufügen aussteifenden Stäben dient lediglich der numerischen Stabilisierung des Gleichungssystems bei der anschließenden Berechnung.

Durch die Anpassung der Steifigkeiten mit der Forderung nach Kompatibilität, ist ein Minimierungsproblem zu lösen (s. Abschnitt 3.4.10), welches idealerweise in das Fachwerkprogramm zu integrieren ist. Dafür wird ein eigener objektorientierte Code (TRUSS) verwendet.

Als Vorstufe zur Bemessung und zum Verständnis des Kraftflusses erfolgt anschließend eine Berechnung der Stabkräfte. Wie bei den anderen Komponenten muss auch hier der gekapselte Prozess zunächst angestoßen werden. Danach können die notwendigen Daten durch eine Client–Server–Beziehung ausgetauscht werden. Wieder werden die Ergebnisse durch AVS/Express dargestellt (s. Bild 5.9).

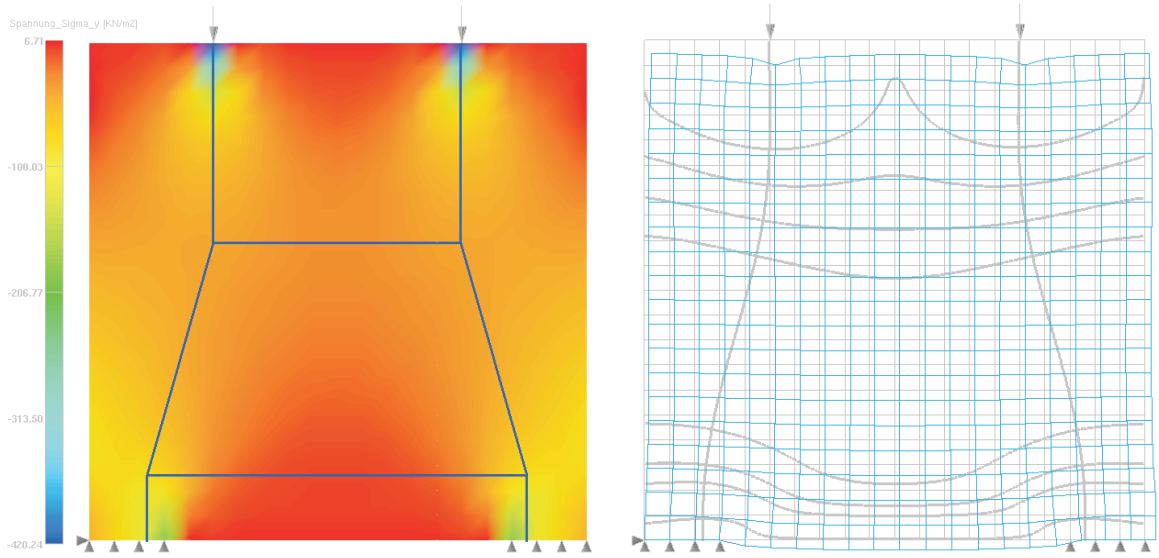


Abbildung 5.9: Darstellung von Stabwerksmodell, Verlauf der vertikalen Spannungen und Deformation

Mit Hilfe der schon erwähnten Darstellungsmöglichkeiten ist eine plausible Begründung der Ergebnisse möglich.

Für das Beispiel der streng strukturierten Kraftflussberechnung lässt sich ein Arbeitsablauf durch das einfaches Flussdiagramm in Bild 5.10 darstellen. So kann für die meisten Aufgaben in Ingenieurdisziplinen ein streng geregelter Ablauf bestimmt werden.

Bis vor kurzem haben sich Software–Hersteller wie Forschungseinrichtungen das Konzept fest definierter Abläufe zum Paradigma ihrer Simulationssoftware gemacht. Die enorme Weiterentwicklung der Computer und Netzwerke eröffnen dem Anwender jedoch einen viel größeren Anwendungsbereich, als bis vor wenigen Jahren denkbar war.

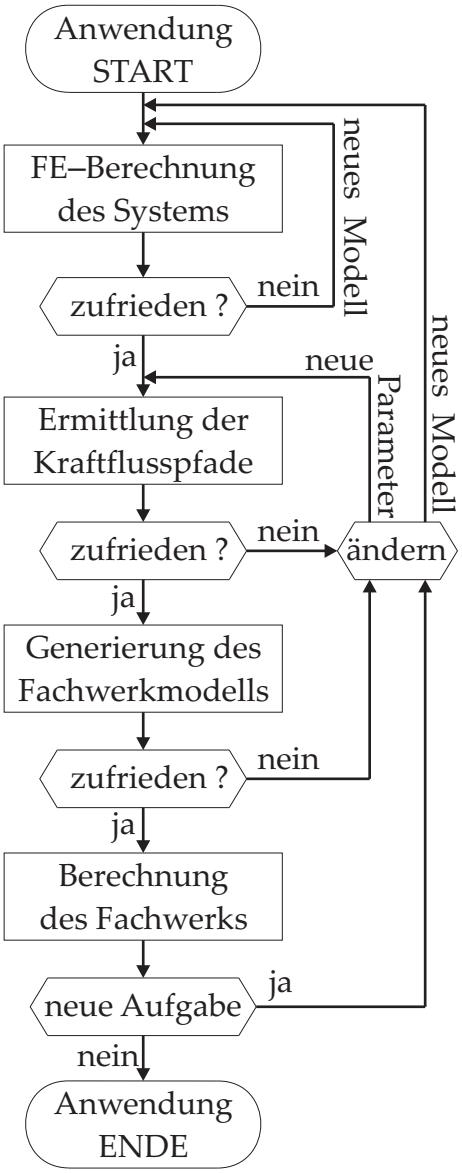


Abbildung 5.10: Workflow der Anwendung

5.4 AVS/Express als Workflow–Management–System

Mit den alten Programmen können neuartige Problemstellungen meist nicht ohne weiteres gelöst werden. Umständliche und zeitraubende Speziallösungen müssen geschaffen werden. Gerade für die vielfältigen Herausforderungen im Bereich der Computersimulation werden völlig neue Abläufe notwendig, mit wechselnden Kombinationen von spezieller Software.

Es macht also durchaus Sinn, den Ablauf einer Computerberechnung nicht mehr zwingend nach einem festen Schema vorzuschreiben, sondern dem Anwender gewisse Freiheiten zu gewähren. Je nach Anforderung sollten unterschiedliche Pakete geladen und miteinander verknüpft werden können.

Spezielle Problemstellungen wie dynamische Rechnungen, die Interaktion von Systemen (z.B. FSI) oder die Optimierung sollten durch vordefinierte und kombinierbare Strukturen ermöglicht werden. Ein objektorientierter Aufbau macht ein derartiges Programmsystem nicht nur für Anwender, sondern insbesondere für Entwickler interessant. Forschungseinrichtungen und Software–Hersteller sind damit in der Lage, einzelne Module zu ersetzen oder hinzuzufügen.

Zusammen mit der Interprozesskopplung führt der letzte Aspekt unweigerlich auf die vollständig entkoppelte und verteilte Entwicklung. In einem größeren Softwareprojekt sind diverse Entwicklergruppen nicht mehr an bestimmte Systemumgebungen oder Programmiersprachen gebunden. Der Quellcode braucht nicht mehr für andere Gruppen zur Verfügung gestellt werden, denn die jeweiligen Prozesse können ausschließlich lokal *programmiert*, *compiliert* und *gelinkt* werden. Lediglich eine gemeinsame Kommunikationsschnittstelle ist zu definieren.

AVS/Express als Entwicklungsumgebung mit der Möglichkeit der visuellen Programmierung ist gut geeignet, die Idee eines Workflow–Management–Systems umzusetzen. Die implementierte Lösung zur Steuerung von Workflows wird als Workflow–Management–Anwendung bezeichnet.

Ein Workflow ist in diesem Zusammenhang ein Arbeitsablauf, der aus einzelnen Aktivitäten aufgebaut ist, die sich auf Teile eines organisatorischen Vorganges beziehen. Der Workflow hat einen definierten Anfang, einen organisierten Ablauf und ein definiertes Ende. Die anfallenden Tätigkeiten werden vom Anwender bzw. Software–System koordiniert.

Bei der eben erwähnten verteilten Entwicklung dient ein Workflow–Management–System jedem autorisierten Teilnehmer als Zugriffsmöglichkeit auf alle freigegebenen Teilprozesse. Jedem Einzelnen zeigt sich dabei das selbe kohärente Softwaresystem, nachdem die Prozesse unter Einhaltung gewisser Regeln miteinander verbunden sind.

Das in Abschnitt 3.4 beschriebene Verfahren zur Visualisierung des Kraftflusses dient hierbei als Anwendungsbeispiel. Dessen Umsetzung kann in seiner komplexen Struktur vereinfacht und in drei wesentliche Teile zerlegt werden. Abbildung 5.11 zeigt in der oberen Leiste diese vorgefertigten Module, die der Anwender im unteren Abschnitt zu einem Workflow zusammenstellen kann.

Hierbei handelt es sich um die bereits beschriebenen Schritte der Kraftflussberechnung. Mit CARAT ist ein Modul gegeben, welches das gleichnamige Finite–Elemente–Programm beinhaltet. Weiters besteht eine interaktive Benutzeroberfläche und einem Modul zur Filterung und Aufarbeitung der Ergebnissdaten.

Das selbst generierte Makro ALFIT ist in der Lage, aus den Datensätzen einer FE-Berechnung Kraftflusspfade und ein Fachwerk zu generieren. Auch hier ist eine

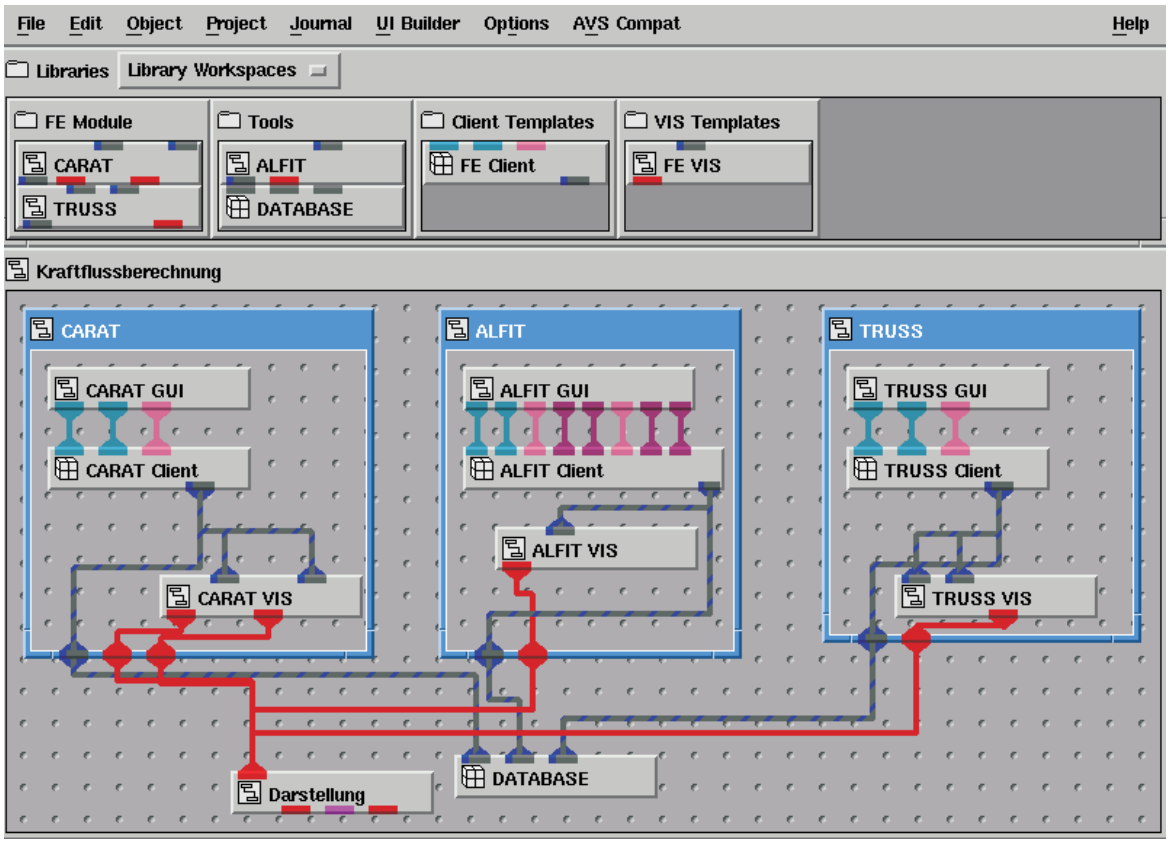


Abbildung 5.11: Die Gliederung der Anwendung

graphische Benutzerschnittstelle und ein Datenfilter implementiert. Bei der Erweiterung der Modul–Bibliothek ist es denkbar und sinnvoll, daraus zwei getrennte Module zu erzeugen.

Ein drittes Modul TRUSS dient der Berechnung von Fachwerken und ist ebenfalls mit dem notwendigen Benutzerinterface ausgestattet. Die von AVS/Express benötigten Daten zur Visualisierung werden von allen Prozessen in einer gemeinsamen Datenbank gesammelt.

Damit ist für den Endanwender ein leicht zu bedienender Workflow–Manager geschaffen, in dem durch so genanntes „Drag & Drop“ zunächst die genannten drei Module zur eigentlichen Anwendungen verknüpft werden können. Werden weitere kompatible Module zur Modul–Bibliothek hinzugefügt, sind individuelle Programmsysteme ohne Eingriff in Quellcode zur Laufzeit generierbar. Damit ist jeder in der Lage, ein Werkzeug zu schaffen, welches auf eine individuelle Problemstellung angepasst ist.

5.5 Andere Visualisierungstools

An Universitäten und anderen Forschungseinrichtungen wird rasant an der Weiterentwicklung von Berechnungsmethoden gearbeitet. Die Visualisierung der Ergebnisdaten ist dabei ein wesentlicher Bestandteil der Entwicklung. Das eben beschriebene, einfache Tool, dient als plattformunabhängige, flexible und erweiterbare Softwareschnittstelle, um die Kommunikation und den Datentransfer zwischen Prozessen in einem verteilten System zu ermöglichen. Im Beispiel der Visualisierung können damit die Ergebnissdaten an ein beliebiges Darstellungsmodul in einem verteilten System weitergeleitet werden, ohne auf so genannte File–Schnittstellen zurückgreifen zu müssen.

Wie angesprochen existieren eine Vielzahl von Softwarepaketen zur Visualisierung von Daten. Hier sei GID als ein Beispiel herausgegriffen. Es ist im Speziellen auf die Darstellung von FE–Berechnungsergebnissen zugeschnitten. Dabei kann es sowohl als Pre- als auch als Postprozessor eingesetzt werden. Unterschiedliche Hilfsmittel bieten auch hier große Freiheiten in der grafisch interaktiven Arbeit.

Jedoch ist es in GID nicht möglich direkt während der Laufzeit des Prozesses auf die Datenhaltung zuzugreifen. Somit ist es wie auch bei vielen anderen Visualisierungstools nicht möglich interaktiv mit anderen Prozessen zu agieren. Daten müssen in Files geschrieben, übertragen und wieder interpretiert werden. Dies ist zeit- und speicherintensiv, und bietet dabei nicht die Flexibilität der direkten Datenübertragung von Arbeitsspeicher zu Arbeitsspeicher.

Nachfolgend ist eine unvollständige Liste gängiger Visualisierungstools mit kurzer Beschreibung angegeben. Verweise auf weitere Projekte bietet das Internet unter den Suchbegriffen *visualization software* in Verbindung mit *finite element analysis*, *computational fluid dynamics* oder *mesh/grid generation*.

ParaView	Postprozessor für sehr große Datenmengen; open source
OpenDX	Visualisierungspaket für wissenschaftliche Anwendungen; UNIX; open source
MayaVi	Paket für wissenschaftliche Visualisierungen; UNIX, LINUX, Windows; open source
HIGHEND	Interaktive Visualisierung für hierarchische Datensätze; frei erhältlich
VIGIE	eigener Quellcode kann integriert werden; unterstützt X11; frei erhältlich;
pV3, Visual3	entwickelt für CFD; frei erhältlich
CAF3D, FEMFlow	Strömungsvisualisierung; frei erhältlich
CEI	Visualisierung für FEA, CAE und CFD; kommerziell

Kapitel 6

Numerische Beispiele

Die folgenden Beispiele geben lediglich einen Teil des große Spektrums an Scheibentragwerken im Bauingenieurwesen wieder. Sie dienen der Untersuchung und Bewertung des vorgeschlagenen Algorithmus zur Generierung von Stabwerkmodellen aus Kraftflusspfaden. Das Vorgehen wird nochmals bildhaft dargestellt und Hinweise zur richtigen Modellierung gegeben.

Im Einzelnen werden die Einfeld- und die Zweifeldscheibe unter Gleichlast, die Kragscheibe unter Gleich- bzw. Einzellast und die beiderseits eingespannte Scheibe unter Einzellasten untersucht, wie in Abbildung 6.1 dargestellt. Diese und weitere Beispiele sowie Vergleichsdaten finden sich unter anderem in Mörsch [Mör12], Leonhardt [Leo77], im Betonkalender 1993 Teil II [Sch93] und dem DAfStb Heft 240 [Gra91].

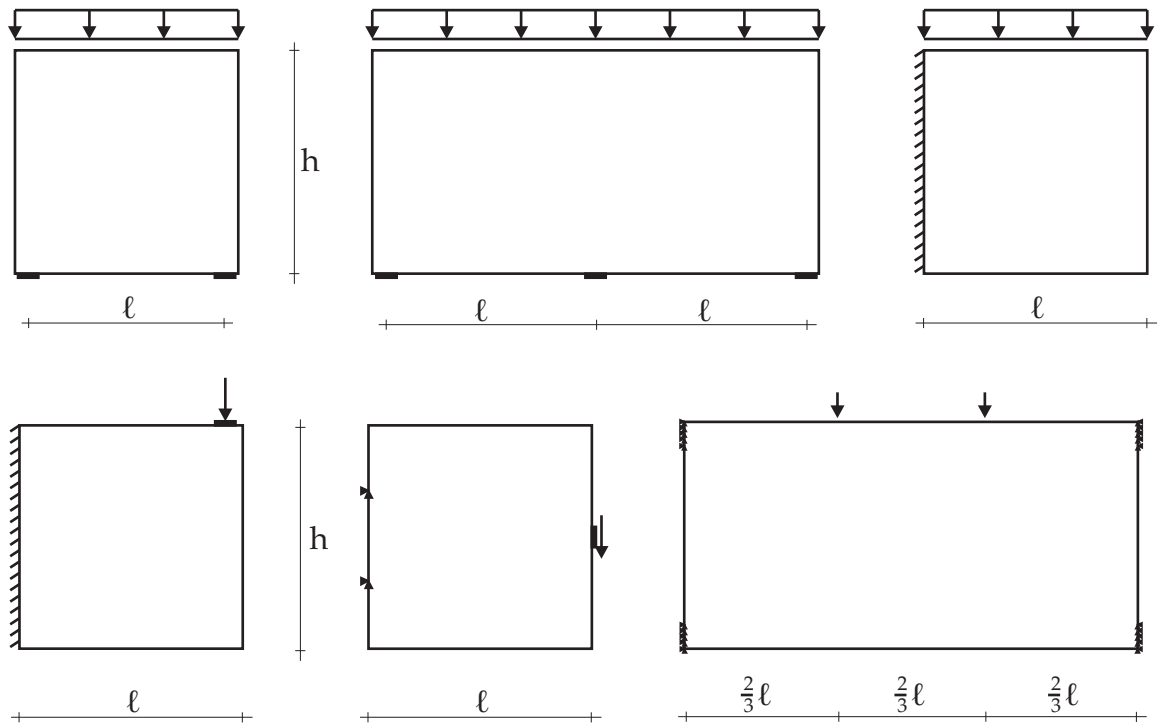


Abbildung 6.1: Geometrie und Belastung der behandelten Beispiele

6.1 Hinweise zur Modellierung

Die Modellierung beschäftigt sich mit der idealisierten Beschreibung eines realen Tragwerks. Sie spielt eine wesentliche Rolle, schließlich muss die Wirklichkeit hinreichend genau abgebildet werden. Die Aspekte einer sinnvollen Modellierung von Systemen für die erfolgreiche Anwendung der Finite–Elemente–Methode sind geeigneter Literatur zu entnehmen, z.B. [Coo95], [Mül00], [Har02], [Wer95].

Da die Kraftflussberechnung den Spannungsverlauf als Grundlage verwendet, ist dieser mit Hilfe richtiger Modellierung möglichst gut abzubilden. Das bedeutet, dass an Orten mit Diskontinuitäten das Modell fein genug diskretisiert werden muss. Dies gilt insbesondere an Stellen mit konzentrierter Lasteinleitung oder punktuellen Lagerungen. Die folgenden Abschnitte beleuchten deswegen wesentliche Aspekte bei der Modellierung der Belastung und der Auflager.

Modellierung der Last

Bei der Finite–Elemente–Methode werden Lasten in der Regel durch Knotenkräfte dargestellt. Kontinuierlich verteilte Lasten werden durch konsistente Kräfte auf allen Knoten in deren Einflussbereich ersetzt. Eine Abweichung davon ist unter Umständen notwendig und sinnvoll, bedarf jedoch zusätzlicher Überlegungen.

Die Auswirkung unterschiedlicher Modellierung einer verteilten Last auf die Spannungen im System zeigt Abbildung 6.2 exemplarisch an der vertikalen Spannungskomponente σ_y. Dargestellt ist jeweils die Hälfte einer Scheibe auf zwei Lagern unter Gleichlast, die jeweils mit einer unterschiedlichen Anzahl an Knotenlasten modelliert ist. Zu Vergleichszwecken beträgt die vertikale Summe der Belastung jeweils 400 kN. Bei Ort und Betrag der Knotenlasten sind Symmetrieeigenschaften und Einflussbreiten berücksichtigt.

Gut zu erkennen ist der homogene Spannungszustand des Bildes links oben, wie er sich einstellt, wenn die Gleichlast durch konsistente Einzellasten an jedem Knoten modelliert wird. Das entspricht sehr gut dem tatsächlichen Verhalten eines Systems. Je gröber die Last modelliert wird, desto mehr prägt sich ein diskontinuierlicher Bereich der vertikalen Spannung σ_y aus. Im Bild rechts oben wird nur jeder zweite Knoten belastet, während in den beiden unteren Bilder die konstante Last nur mehr durch vier bzw. zwei Einzellasten ersetzt wird.

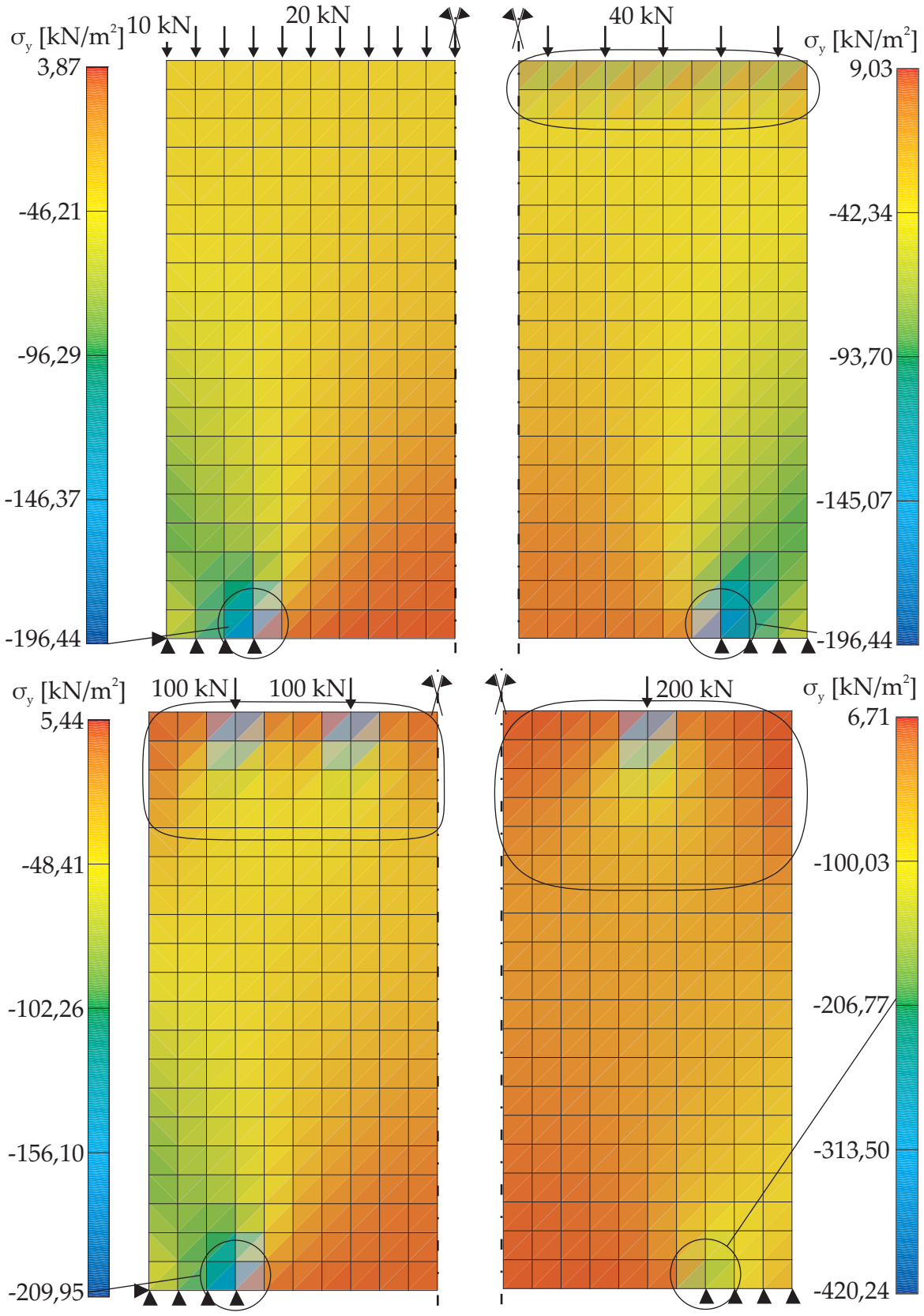


Abbildung 6.2: Vertikale Spannung σ_y bei unterschiedlicher Lastmodellierung

Der Abstand der Einzellasten zueinander bzw. zum Rand spiegelt in etwa auch den Einflussbereich der Diskontinuität wieder, den so genannten St. Venant–Bereich. In diesem Gebiet wird der tatsächliche Spannungszustand nicht mehr richtig abgebildet. Wie durch Abbildung 6.2 deutlich wird, entstehen bei großen Einzellasten wie auch an punktuellen Auflagern so genannte Spannungsspitzen. Das hat unter Umständen Einfluss auf Methoden, die auf die Spannungswerte zurückgreifen.

Ist dennoch eine derartige Modellierung der Last gewünscht, muss eventuell das System auf geeignete Weise angepasst werden. Abbildung 6.3 links zeigt, wie die Wirkung einer Gleichlast auf das Tragwerk auch durch Einzellasten erreicht werden kann. In diesem Fall werden zwei Einzellasten auf einer Reihe sehr steifer Elemente aufgebracht, die eine gleichmäßige vertikale Pressung der darunter liegenden Elemente hervorruft. Ein homogener Spannungszustand, wie unter einer Gleichlast, stellt sich ein. Auf der rechten Seite des Bildes ist das System um den St. Venant–Bereich nach oben hin erweitert worden, so dass sich in Höhe der ursprünglichen Systemkante der gewünschte homogene Spannungszustand einstellt.

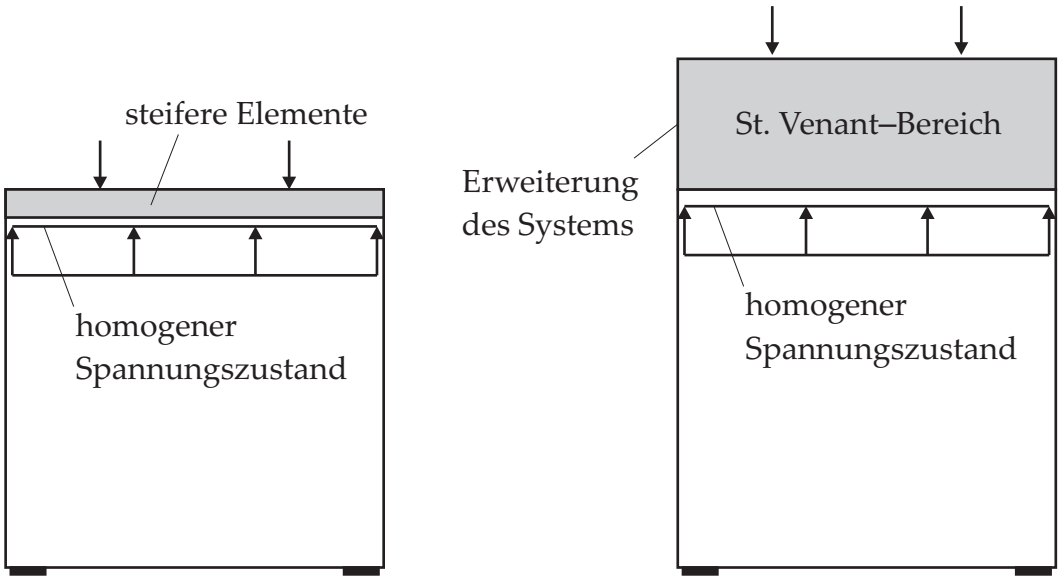


Abbildung 6.3: Homogener Spannungszustand trotz Einzellasten durch Systemanpassung

Durch die in Kapitel 3.4 beschriebene Methode, wird für jede Lasteinleitungsstelle ein Kraftflusspfad injiziert. Das dadurch erzeugte Stabwerkmodell steht im Gleichgewicht mit der modellierten Belastung und zeigt die Tragwirkung des Systems durch die Diskretisierung des Spannungszustandes. Je feiner also die Modellierung der Last ausfällt, desto weiter bewegt man sich in Richtung kontinuierlicher Lösung. Das generierte Stabwerkmodell ist durch die Ausrichtung an den Hauptspannungstrajektorien in gewisser Maßen optimiert. Ein Hinweis darauf findet sich in Abschnitt 3.3.6 beim Vergleich mit den Minimalstrukturen von Michell, die an den Hauptverzerrungstrajektorien ausgerichtet sind. Diese sind identisch mit den Hauptspannungstrajektorien bei isotropem Werkstoffverhalten.

In Abbildung 6.4 wird am Beispiel der einfachen Scheibe auf zwei Stützen unter Gleichlast die Auswirkung unterschiedlicher Lastmodellierungen deutlich.

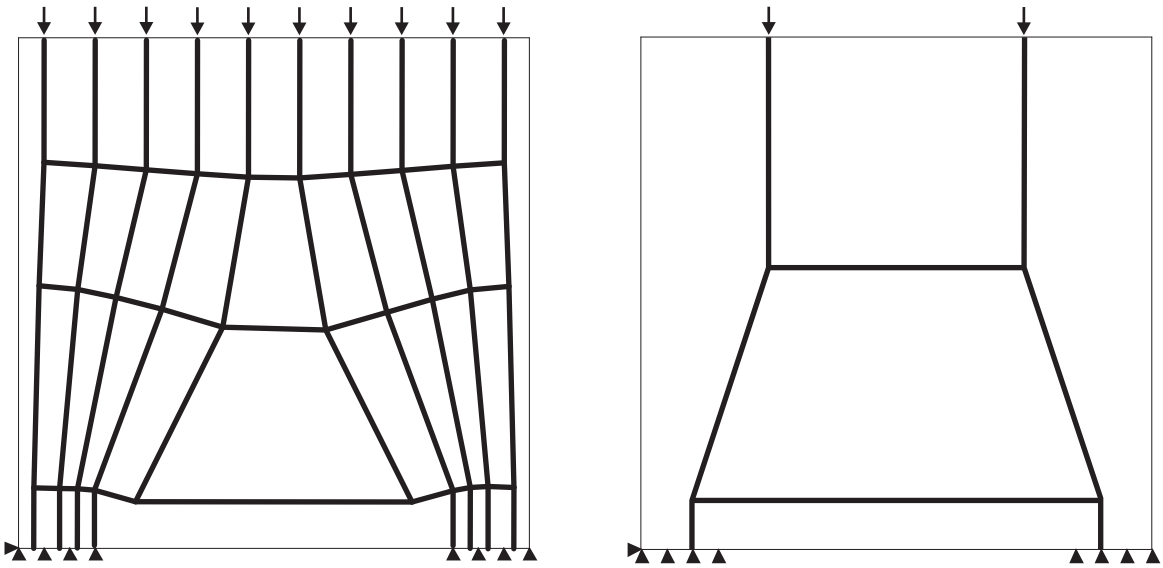


Abbildung 6.4: Stabwerkmodelle infolge unterschiedlicher Modellierung einer Gleichlast

Zusammenfassend ist zu sagen, dass durch die Modellierung der Belastung auch die Komplexität des Stabwerkes gesteuert werden kann. In manchen Fällen hilft ein einfaches Stabwerk, welches aus einer groben Diskretisierung der Last entsteht, das Tragverhalten eines Systems besser zu verstehen (vgl. hierzu auch Abschnitt 3.2).

Modellierung der Auflager

Bei der Modellierung der Auflager gelten ebenfalls die allgemein bekannten Regeln für die Anwendung der Finite–Elemente–Methode. Auch hier gilt es den tatsächlichen Spannungszustand durch die Modellierung möglichst gut abzubilden. Geometrische Randbedingungen sollten dabei auf einen endlichen Bereich anstelle auf einzelne Knoten aufgebracht werden. Gegebenenfalls ist dieser Bereich feiner zu diskretisieren. In Anlehnung an das DAfStb Heft 240 sind die Lager aller Beispiele starr modelliert, um einen sinnvollen Vergleich durchführen zu können.

Da die spätere Ausrichtung der Stäbe des Stabwerkmodells an den Lagerungsbedingungen angepasst wird, sind hier zusätzliche Überlegungen anzustellen. Verschiebliche Auflager können nur Kräfte in der festgehaltenen Richtung aufnehmen, deswegen werden Stäbe senkrecht zu ihrer Verschieblichkeit ausgerichtet. Trifft der Stab auf ein unverschiebliches Lager, behält er seine Richtung. Eingespannte Ränder werden durch unverschiebliche Knoten modelliert. Da die verwendeten verschiebungsformulierten Elemente keine Rotationsfreiheitsgrade besitzen, können diese auch nicht festgehalten werden.

Durch die beschriebene Methode ist nicht sichergestellt, dass die Kraftflusspfade am Systemrand in einem Auflagerknoten münden. Deswegen wird dem Schnittpunkt von Kraftflusspfad und Systembegrenzung die Lagerungsbedingung des nächst gelegenen Knotens übertragen.

Bei symmetrischen Systemen mit symmetrischen Lastfällen wird auch ein symmetrisches Verhalten des Tragwerks erwartet. Das gleiche gilt für das zugehörige Stabwerkmodell. Abbildung 6.5 zeigt ein Beispiel, bei dem dies aufgrund ungünstige Lagerungsmodellierung und zu grober Diskretisierung der Kraftflusspfade nicht der Fall ist. Auch diesem Umstand ist durch geeignete Modellierung Rechnung zu tragen.

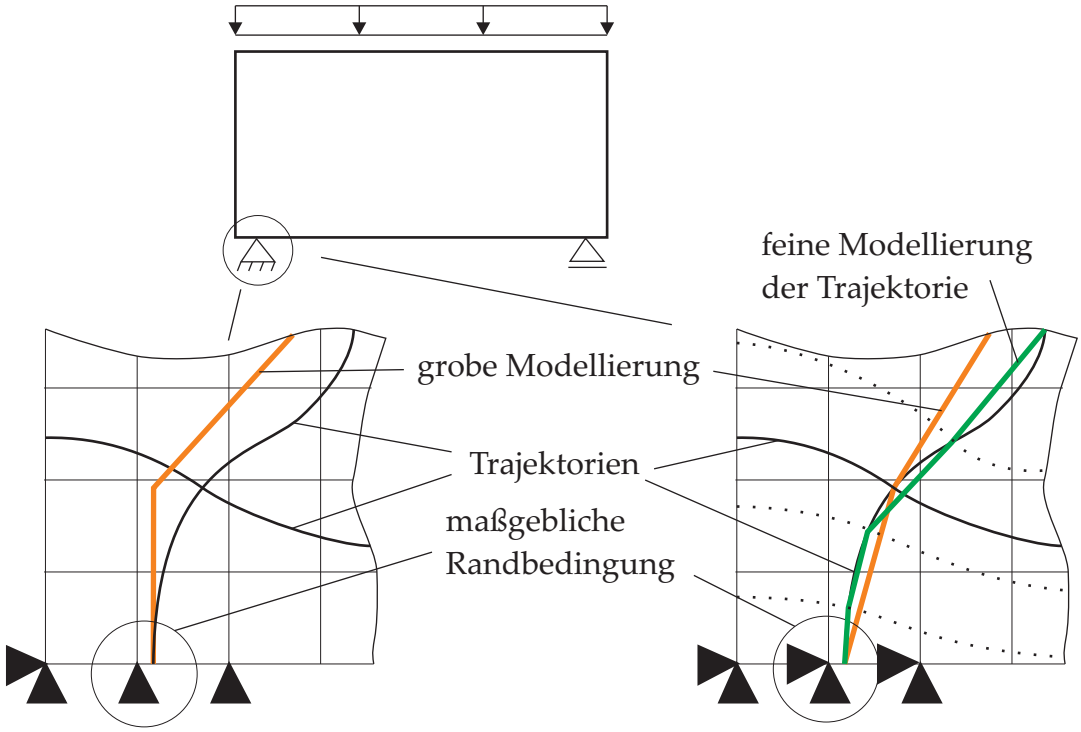


Abbildung 6.5: Anpassung des Fachwerkstabes an die Lagerungsbedingung

Im Bild links wird das unverschiebliche Lager nur am äußersten Punkt auch horizontal fixiert, um die Verschieblichkeit des gesamten Systems auszuschalten. Zur Einhaltung der Symmetriebedingungen sind alle anderen Knoten wie am rechten Auflager horizontal verschieblich modelliert (hier nicht abgebildet). In Folge dessen ergibt sich erwartungsgemäß ein symmetrisches Stabwerkmodell. Im rechten Bild sind hingegen alle Knoten fixiert, so dass die Normalkraft des Fachwerkstabs in beliebiger Richtung in das Auflager abgetragen werden kann. Eine Richtungsanpassung wird in diesem Fall nicht vorgenommen, woraus ein nicht symmetrisches Stabwerkmodell entsteht. Für den Spannungsverlauf des gesamten Tragwerks sind beide Varianten nahezu identisch, jedoch beschreibt nur das Stabwerkmodell der ersten Variante das zu erwartende symmetrische Tragverhalten. Bei ausreichend

feiner Modellierung ergibt sich diese Problematik nicht, da die Stabausrichtung am Auflager korrekt abgebildet wird.

6.2 Scheibe auf zwei Stützen

Im erstes Beispiel wird eine quadratische Einfeldscheibe mit 10 m Kantenlänge unter Gleichlast der Größe 40 kN/m betrachtet und ihr Tragverhalten etwas detaillierter beleuchtet. Für die vorausgehende linear elastische Spannungsberechnung nach der Finite–Elemente–Methode wird ein strukturiertes Netz von 20x20 vierknotigen Elementen zu Grunde gelegt, wie Abbildung 6.6 zeigt.

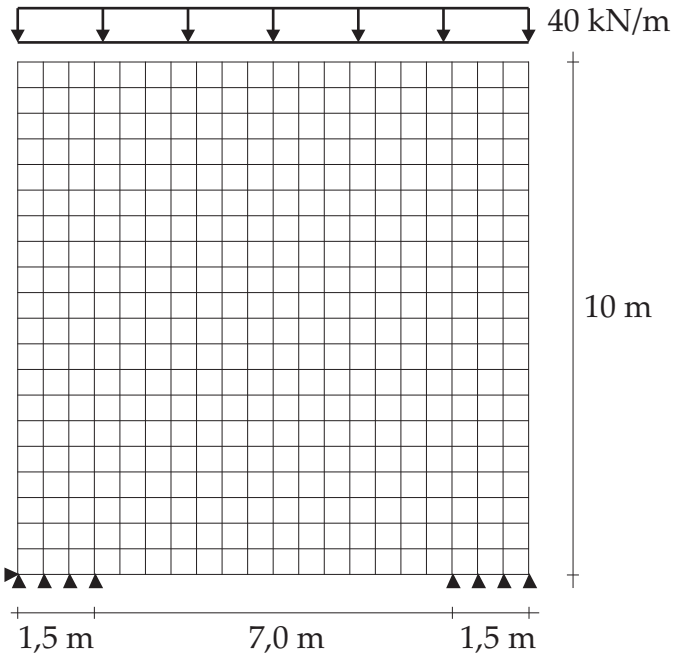


Abbildung 6.6: FE–Modell der Einfeldscheibe

Damit können sowohl die Auflagerbereiche, als auch die Belastung hinreichend genau modelliert werden, ohne ausgeprägte Spannungsspitzen erwarten zu müssen. Weiters gewährt die hohe Dichte an Spannungsinformationen gute Ergebnisse bei der Bestimmung der Hauptspannungstrajektorien.

Wie bereits erwähnt ist die Modellierung der Belastung bei der Anwendung der Finite–Elemente–Methode ein zentrales Thema. In den Abbildungen 6.7 und 6.8 sind insgesamt vier unterschiedliche Modellierungen der Gleichlast mit den jeweiligen Kraftflusspfaden und den daraus resultierenden Stabwerkmodellen einander gegenübergestellt.

Nur im ersten Fall ist das für die Berechnung der Stabkräfte zugrunde liegende statisch bestimmte Fachwerk in Abbildung 6.7 rechts oben dargestellt. Die aus-

steifenden Diagonalstäbe waren lediglich zur numerischen Stabilisierung des Gleichungssystems für die Finite–Elemente–Berechnung mit TRUSS notwendig. Wie in allen anderen behandelten Fällen zeigen sie auch hier keine nennenswerten Kräfte. Das ist ein Hinweis darauf, dass sich das gefundene kinematische Stabwerk in einer optimierten Gleichgewichtslage befindet und keine aussteifende Diagonalstäbe notwendig sind. Aufgrund der Übersichtlichkeit wird für alle weiteren Beispiele auf die Darstellung der Aussteifungen verzichtet. Auch die ermittelten Stabkräfte werden deswegen nur in den ersten beiden Modellierungen gezeigt.

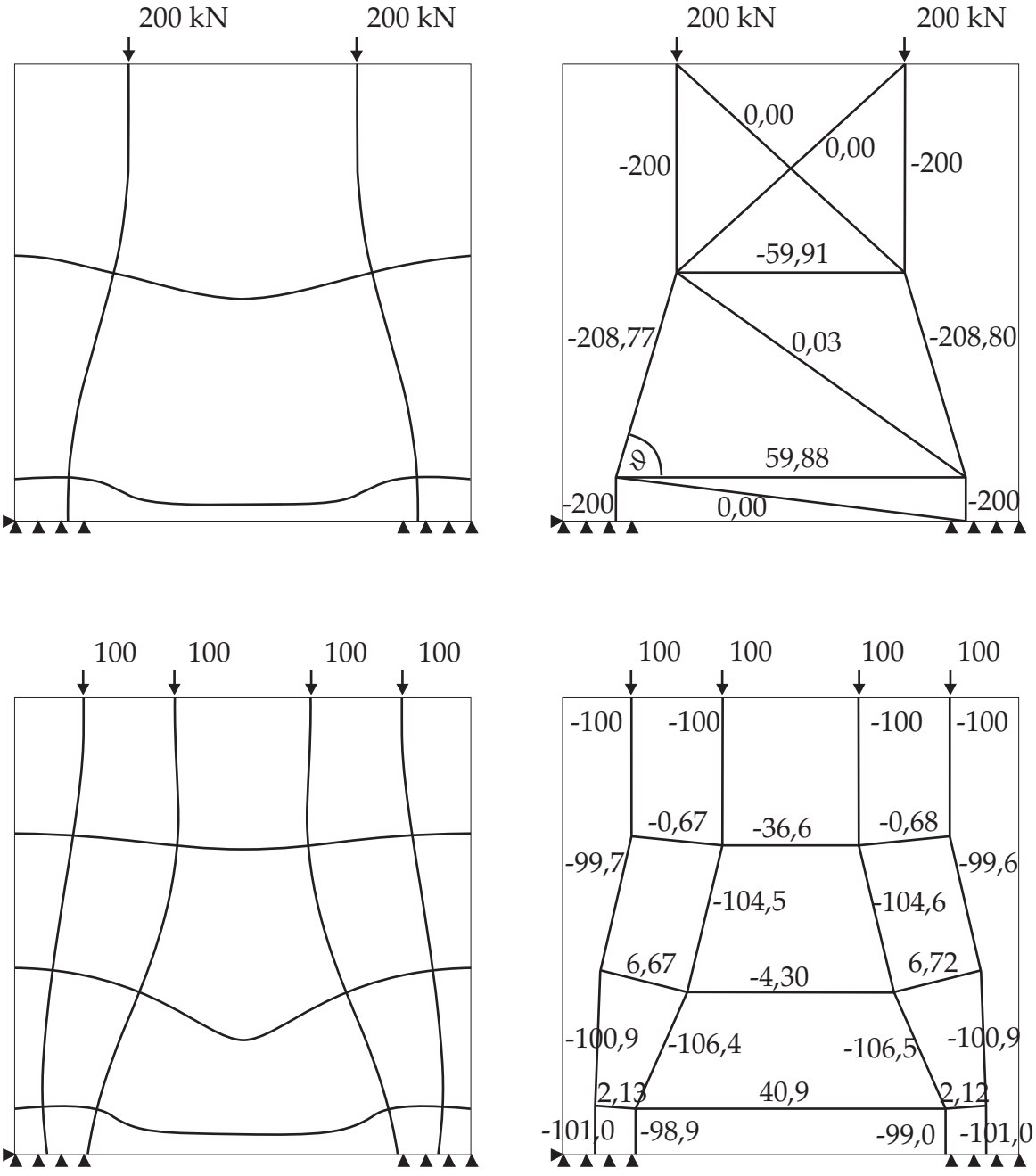


Abbildung 6.7: Kraftflusspfade und Stabwerkmodelle bei zwei und vier Einzellasten

Durch die vorab beschriebene Methode wird aus der gegebenen Laststellung zunächst ausgewiesene Kraftflusspfade erzeugt, die in den Bildern jeweils links zu sehen sind. Deren Schnittpunkte markieren optimalen Orte für Knoten eines Stabwerkes, welches einen möglichen Gleichgewichtszustand des modellierten Systems darstellt. Die Stäbe orientieren sich dabei an den Kraftflusspfaden, was bei einer feineren Diskretisierung deutlicher in Erscheinung tritt.

Im Bild 6.8 ist durch die fein aufgelöste Gleichlast der Übergang der Lastabtragung durch diskrete Stabwerke hin zum Kontinuum erkennbar.

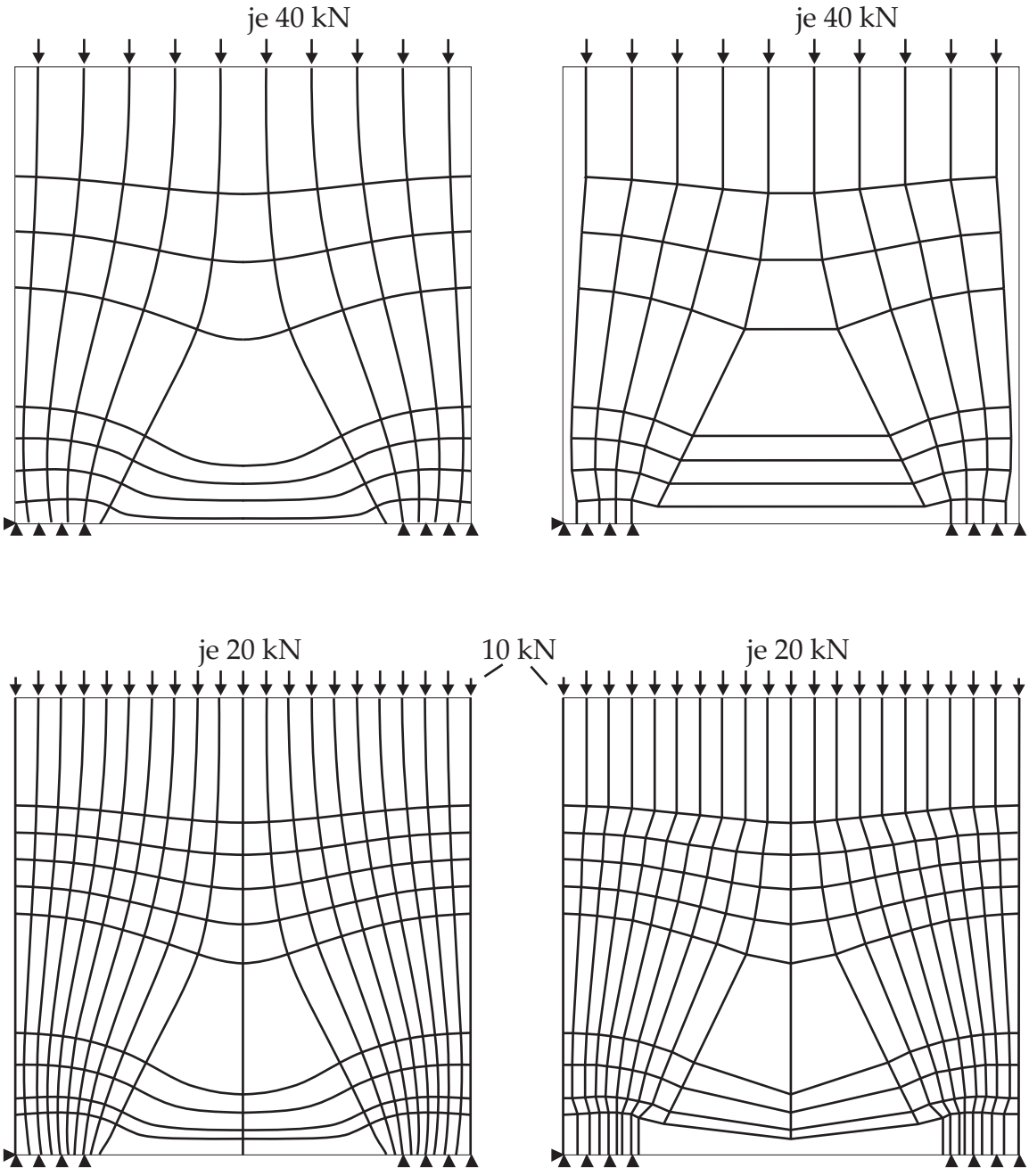


Abbildung 6.8: Kraftflusspfade und Stabwekmodell bei zehn und zwanzig Einzellasten

Bei allen Modellen zeigt sich, obgleich unterschiedlicher Informationsdichte, die selbe Charakteristik. Die Belastung wird durch vertikale Trajektorien nach unten hin zu den Auflagern abgeleitet. Die dafür notwendige Umlenkung der Kraftflusspfade wird im oberen Bereich und am unteren Rand der Scheibe durch horizontale Umlenkkräfte ermöglicht – hier durch die horizontalen Kraftflusspfade angedeutet.

Aufgrund der statisch bestimmten Lagerung des Stabwerkes und der Tatsache, dass es innerlich nicht überbestimmt ist, können die Schnittgrößen unabhängig von den Verformungen oder Steifigkeiten alleine aus dem Gleichgewicht berechnet werden.

Die Normalkräfte der vertikalen Stäbe entspricht je nach Neigung etwa den Knotenlasten und ist deswegen abhängig vom Diskretisierungsgrad der Gleichlast. Je feiner die Last modelliert wird, desto feiner ist das Stabwerkmodell zur Beschreibung deren Abtragung. Die Unterschiede in den horizontalen Stäben resultieren hauptsächlich aus den unterschiedlichen Geometrien. Im Übergang zur kontinuierlichen Lastmodellierung werden durch das Stabwerkmodell die Hauptspannungen dargestellt.

In der gängigen Literatur, z.B. in DAStb Heft 240 [Gra91] oder im Betonkalender Teil II 1993 [Sch93], wird die Gleichlast meist durch zwei Einzellasten modelliert. Das zugehörige Stabwerkmodell weist alle wichtigen Aspekte der Lastabtragung auf und wird aufgrund seiner Einfachheit und Aussagekraft favorisiert (vgl. Abbildung 6.9).

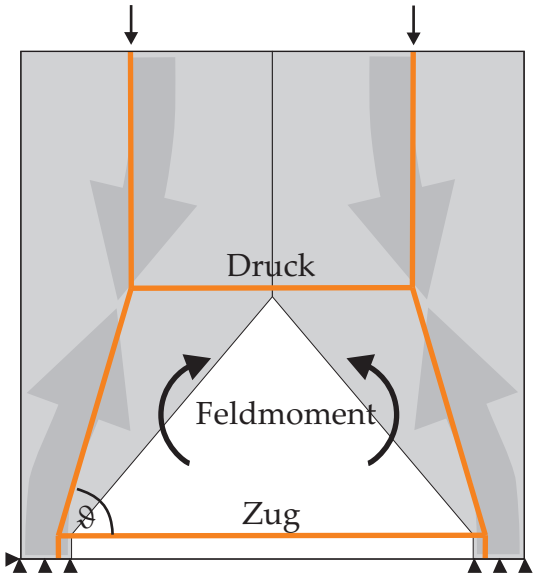


Abbildung 6.9: Schematische Darstellung der Lastabtragung einer Einfeldscheibe

Dabei genügen zwei vertikale Stabreihen, um die symmetrische Lastabtragung der vertikalen Last in die Auflager darzustellen (hell schattierter Bereich im Bild). Der untere horizontale Stab spiegelt den Zugbereich der Scheibe wieder und ermöglicht

die Umlenkung der vertikalen Kräfte direkt über den beiden Auflagern. Der obere horizontale Stab vervollständigt das sich einstellende Druckgewölbe in der Scheibe, und sorgt ebenfalls für Gleichgewicht der umgelenkten Lastpfade im Bereich der größten horizontalen Spannungen.

Die Lage der Stäbe ist insbesondere von der Geometrie und den Lagerungsbedingungen der zugrunde liegenden Scheibe abhängig. Der Druckstrebenwinkel ϑ hängt dabei vom Verhältnis der Höhe der Scheibe zu ihrer Stützweite ab. Durch das in Kapitel 3.4 beschriebene Verfahren orientiert sich das Stabwerk an den Kraftflusspfaden und definiert dadurch den Druckstrebenwinkel ϑ automatisch. Damit ist ein möglicher Gleichgewichtszustand dargestellt. Wird an keiner Stelle die Grenzen des Materials überschritten, ist damit eine untere Grenze für die Traglast gegeben.

Für einen Vergleich der Ergebnisse aus der Literatur mit dem ermittelten Stabwerk aus zwei Einzellasten wird die Lagerungsbedingung angepasst. Zur Anwendung der Tabelle in Bild 3.3-20 in [Sch93] ist ein Verhältnis der Auflagerbreite zur Stützweite der Scheibe von $a/\ell = 0,1$ vorgegeben. Aufgrund der größeren Stützweite im Vergleich zum Beispiel im Bild 6.7 entstehen größere Kräfte in den beiden horizontalen Stäben.

Die Ermittlung der Normalkräfte der horizontalen Stäbe ist in Bild 6.10 links aufgeführt. Vergleiche hierzu auch Leonhardt [Leo77] oder DAStb Heft 240 [Gra91].

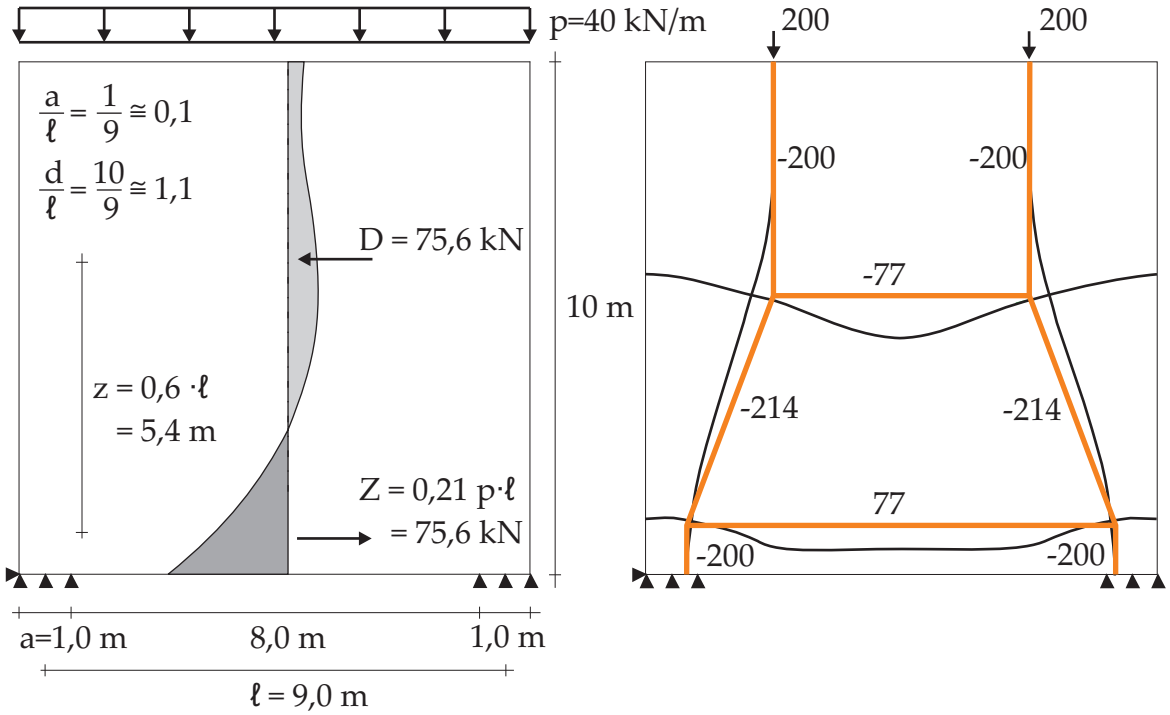


Abbildung 6.10: Trajektorien aus unterschiedlicher Lastmodellierung

Sowohl der Wert als auch die Lage der Resultierenden des Zugbereichs geben die charakteristischen Eigenschaften des Tragverhaltens wieder. Die Druckstrebe liegt jedoch in Höhe der maximalen horizontalen Spannung und nicht wie in der Literatur beschrieben in Höhe des Schwerpunktes der Druckspannungsfläche. Wird die Umlenkung der vertikalen Kräfte feiner aufgelöst, so prägen sich zwei klar definierte Bereiche aus, die sich in der Literatur in guter Näherung wiederfinden.

Im Bild 6.11 wird das Stabwerk aus Abbildung 6.10 wieder aufgegriffen. Links ist das Stabwerk gezeigt, wie es sich automatisch bei etwas feinerer Diskretisierung einstellt. Darin ist zu erkennen, dass sich in der oberen und der unteren Lage jeweils zwei Stäbe ausbilden. Im Vergleich zum Stabwerk aus Bild 6.10 ist deren Kräftesumme jedoch kleiner, was sich durch den in Summe größeren Hebelarm der horizontalen Stäbe erklären lässt. Bei weiterer Verfeinerung war keine wesentliche Änderung der horizontalen Kräftesumme mehr zu erkennen. Auf der rechten Seite im Bild ist der Übergang zur kontinuierlichen Lösung angedeutet.

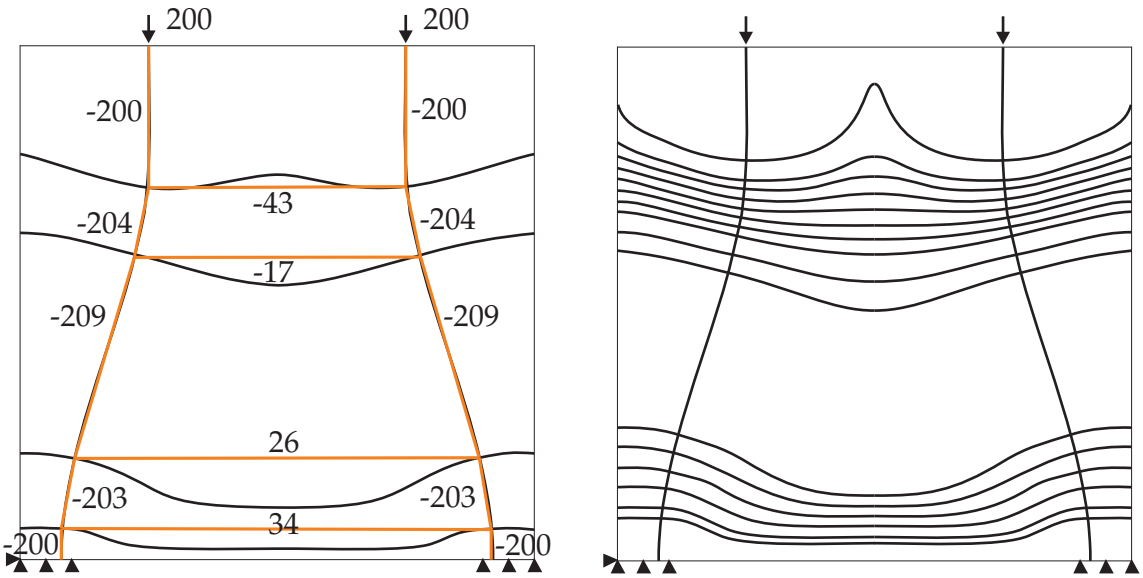


Abbildung 6.11: Variation der Anzahl der horizontalen Stäbe

An den oberen horizontalen Kraftflusspfaden ist das anfangs erwähnte Problem der Diskretisierung einer Gleichlast durch Einzellasten zu erkennen. Erst außerhalb des St. Venant–Bereiches von der Lasteinleitung aus gesehen, stellt sich der homogene Spannungszustand, wie er bei einer Gleichlast zu erwarten ist, ein.

Details zur Bemessung, sowie der Nachweis der Knoten und der Verankerungslängen von Zugbewehrungen sind der einschlägigen Literatur zu entnehmen. Beispielhaft sei hier Abschnitt B im Betonkalender 1993 [Sch93] oder das Vorlesungsmanuskript von Leonhardt [Leo77] genannt.

6.3 Scheibe auf drei Stützen

Eine weitere in der Praxis relevante Anwendung ist die Zweifeldscheibe unter Gleichstreckenlast. Aus den folgenden Betrachtungen lassen sich auch qualitative Rückschlüsse für andere Mehrfeldscheiben ziehen.

In Abbildung 6.12 ist die Geometrie einer 20 m breiten und 10 m hohen Scheibe und das für die Berechnung zu Grunde liegende Finite–Elemente–Netz mit $80x40$ vierknotigen Verschiebungselementen dargestellt. Der Elastizitätsmodul ist mit 10.000 kNm^2 und die Scheibendicke mit $t = 1\ m$ gegeben. Das feine Netz vermindert die Problematik von Spannungsspitzen an den Lasteinleitungsstellen und den Lagern und ermöglicht eine genaue Ermittlung der Kraftflusspfade.

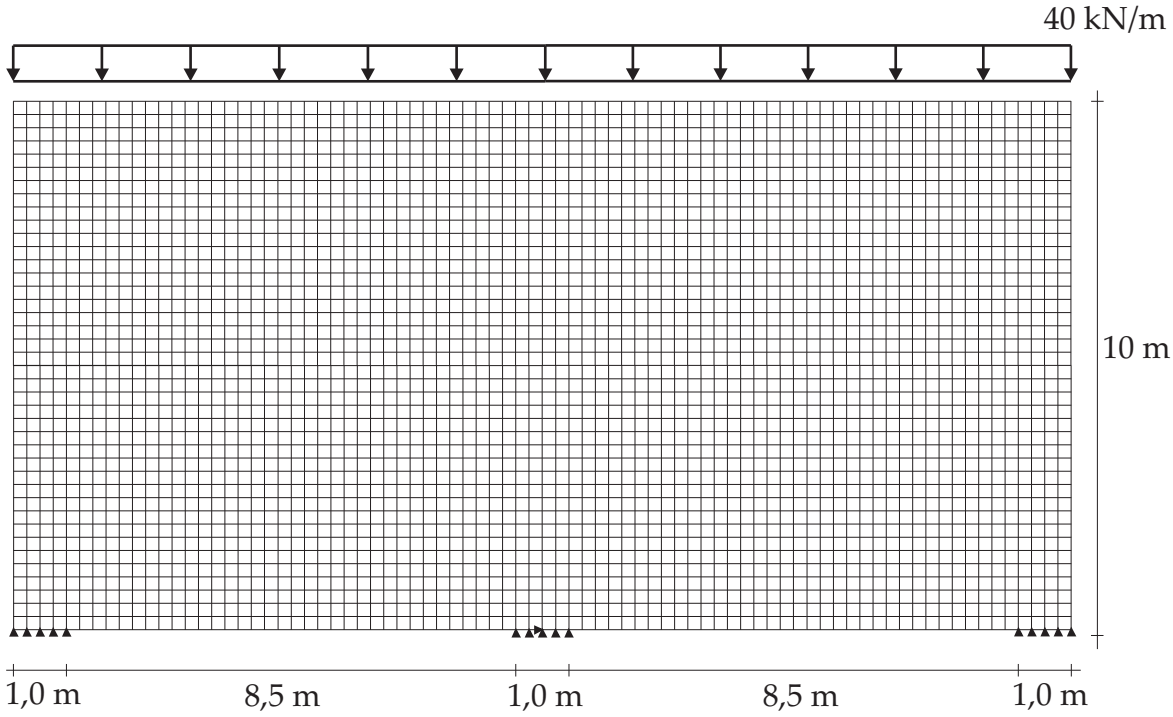


Abbildung 6.12: Zweifeldscheibe unter Gleichlast

Wie im vorigen Beispiel wird auch hier die Gleichlast von 40 kN/m gemäß der Finiten–Elemente–Methode durch Einzellasten modelliert. Soll die Tragwirkung unter Gleichlast visualisiert werden, so ist diese dementsprechend fein aufzulösen, was in Folge auf komplexe Stabwerke führt. In der Grenzbetrachtung wird der kontinuierliche Spannungszustand wiedergegeben.

Zur praktischen Anwendbarkeit der Stabwerksanalogie besteht die Aufgabe darin, die Gleichlast derart grob zu diskretisieren, dass der eigentliche Zustand des Systems noch gut genug abgebildet wird. Das unter dieser Voraussetzung ermittelte Stabwerk gibt dann einen Eindruck vom Tragverhalten des modellierten Systems wieder.

Unter Berücksichtigung der Symmetrieeigenschaften und in Anlehnung an vergleichbare Daten aus der Literatur ist in Abbildung 6.13 ein aussagekräftiger Ansatz mit vier konsistenten Knotenlasten gewählt worden. Das Bild zeigt die Lastpfade gemäß der angesetzten Belastung und das daraus resultierende Stabwerk.

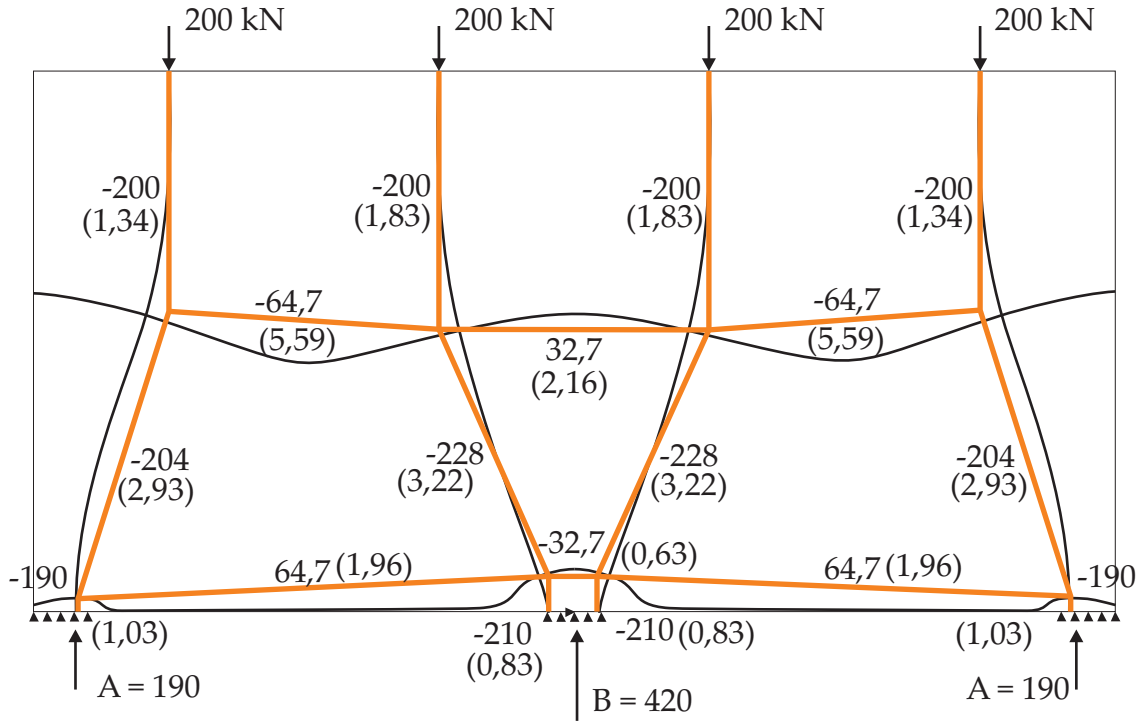


Abbildung 6.13: Lastpfade der Zweifeldscheibe mit Stabkräften und Stabbreiten

Die Stabkräfte sind wegen der statischen Unbestimmtheit des Problems von den Steifigkeitsverhältnissen abhängig. Unter Berücksichtigung der Kompatibilität der Verformungen des diskreten Stabwerkmodells und der kontinuierlichen Scheibe werden die äquvalenten Stabsteifigkeiten aus dem Minimum der Differenzarbeit ermittelt (s. Abschnitt 3.4.10). Für die Anschaulichkeit sind die tatsächlichen Stabbreiten angegeben (Klammerwerte in Bild 6.13 in [m])

Aufgrund der Unbestimmtheit dieses Problems existieren mehrere Lösungen für die Steifigkeits- und damit auch für die Kraftverteilung. Wählt man die Größe der Bewehrung für diesen Fall an einer Stelle (Stabsteifigkeit), so ergibt sich eine eindeutige Lösung für die Lastabtragung. Das bedeutet, dass die Tragwirkung des Systems von der Wahl der Bewehrung abhängig ist. In Abbildung 6.13 ist eine mögliche Steifigkeitskombination dargestellt, durch welche die Kompatibilität und das Gleichgewicht erfüllt sind.

In der allgemein anerkannten Literatur für die Bemessung von Stahlbetonscheiben (z.B. [Leo77], [Sch93] oder [Gra91]) finden sich aufgrund der statisch unbestimmten Lagerung keine eindeutigen Angaben über die Druckstrebenwinkel oder Stabkräfte eines zugehörigen Stabwerkmodells.

An Stelle dessen wird vorgeschlagen eine Bemessung mit Hilfe der Resultierenden der Hauptzugspannungen in Längsrichtung durchzuführen. Die Ermittlung der Spannungen kann aus einer linear–elastischen Berechnung nach der Elastizitätstheorie erfolgen, wobei Lagerungen unnachgiebig zu modellieren sind.

Damit lassen sich Stabkräfte ermitteln, die zum Beispiel dem DAfStb Heft 240 [Gra91] zu entnehmen sind. Unter Zuhilfenahme von Tafel 4.2 sind die Zugkräfte in den Feldern Z_F und über der Mittelstütze Z_S nach Gleichungen 6.1 zu ermitteln.

$$Z_F = 0,15 \cdot q \cdot \ell = 0,15 \cdot 40 \cdot 9,5 = 57,0\ kN \tag{6.1}$$

$$Z_S = 0,19 \cdot q \cdot \ell = 0,19 \cdot 40 \cdot 9,5 = 72,2\ kN \tag{6.2}$$

Der innere Hebelarm der Zugkräfte wird mit $z = 0,45 \cdot \ell = 4,275\ m$ angegeben und ist in Abbildung 6.14 dargestellt. Die Kraftflussberechnung ergibt im Vergleich dazu einen größeren Wert von $z = 0,52 \cdot \ell = 4,94\ m$

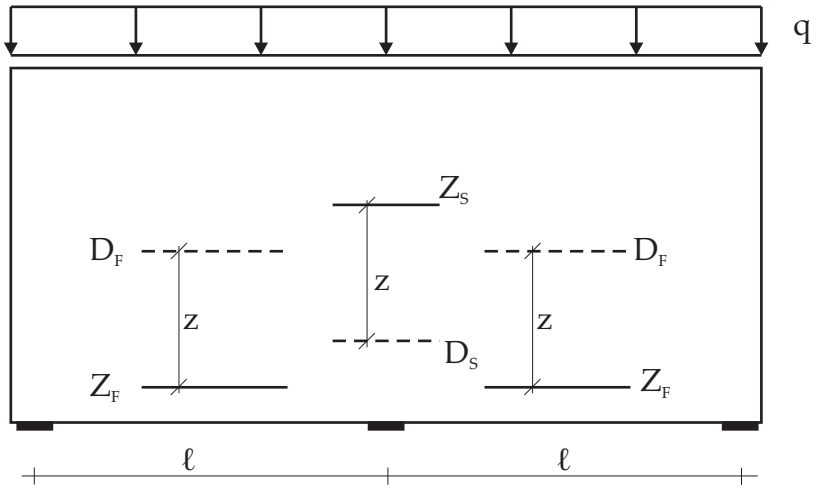


Abbildung 6.14: Lage der Zugkräfte in der Zweifeldscheibe nach DAfStb Heft 240

Ein quantitativer Vergleich ist hier aufgrund unterschiedlicher Annahmen mit Vorsicht zu genießen. Das Stabwerkmodell aus Abbildung 6.13 ist aus einer Belastung mit vier Einzellasten entstanden, was einer recht groben Diskretisierung der Gleichlast entspricht. Die Referenzwerte nach DAfStb Heft 240 [Gra91] orientieren sich an einer linear–elastischen Berechnung der Scheibe unter Gleichstreckenlast.

Eine alternative Herangehensweise zur Bestimmung der Zugkräfte auf Basis von Stabwerken ist im Betonkallender 1993 Teil 2 [Sch93] beschrieben. Hier wird ein Vorschlag bezüglich der qualitativen Lage eines Stabwerkmodells nach Abbildung 6.15 gemacht. Links liegen die Stäbe jeweils im Schwerpunkt der Hauptspannungstrajektorien. Ausreichende Tragreserven gestatten jedoch auch eine Bemessung mit dem vereinfachten Modell rechts.

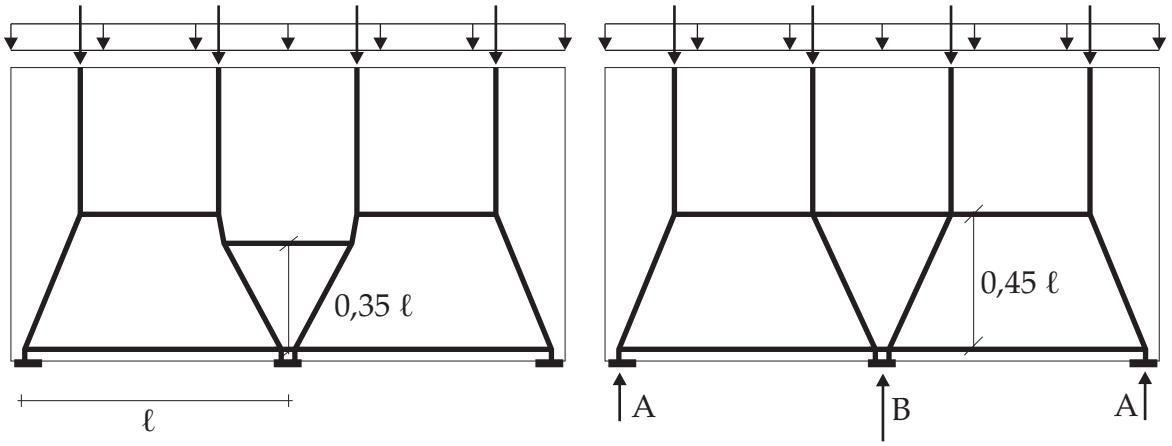


Abbildung 6.15: Vorschlag der qualitativen Lage der Stäbe nach Betonkallender 1993 Teil II

Eine Lösung für das Stabwerk ist nur für bekannte Auflagerkräfte möglich. Unter Annahme konstanter Gleichlast und starrer Stützung ist für das mittlere Auflager ein Wert von $B = 1,125 \cdot p \cdot \ell = 427,5\ kN$ vorgeschlagen. Die beiden äußeren Lager resultieren aus der Summe der vertikalen Kräfte zu $A = 1/2(\cdot 20 \cdot 40 - 427) = 186,25\ kN$.

Der Vergleich mit den Auflagerreaktionen aus Bild 6.13 zeigt eine gute Übereinstimmung mit dem erzeugten Stabwerkmodell. Mit den Auflagerkräften und der in Abbildung 6.15 vorgeschlagenen Geometrie des Stabwerkes können dessen Schnittgrößen bestimmt werden.

Dafür werden Ersatzmomente im Feld und über der Stütze aus den Auflagerreaktionen berechnet (s. Gleichungen 6.3 und 6.3). Die Zugkraft der Stäbe in den Feldern (s. Gleichung 6.5) ergibt sich dann durch Division des Momentes durch den inneren Hebelarm von Druck- und Zuggurt nach Abbildung 6.15. Die Zugkraft über der Mittelstütze berechnet sich analog aus Gleichung 6.6.

$$M_F = A \cdot \frac{1}{2}\ell - P \cdot \frac{1}{4}\ell = 410\ kN\ m \tag{6.3}$$

$$M_S = A \cdot \ell - 2\ P \cdot \frac{1}{2}\ell = -130,6\ kN\ m \tag{6.4}$$

$$Z_F = \frac{M_F}{0,45 \cdot \ell} = 95,8\ kN \tag{6.5}$$

$$Z_S = \frac{M_S}{0,45 \cdot \ell} = 30,6\ kN \tag{6.6}$$

Der Vergleich mit DAStb Heft 240 [Gra91] und dem Stabwerk aus Bild 6.13 ist ebenfalls mit Vorsicht zu genießen. Zwar kann die Zugkraft über der Mittelstütze bestätigt werden, jedoch fällt sie im Feld im Vergleich zum Betonkalender deutlich

geringer aus. Dies ist zum einen auf den etwas größeren inneren Hebelarm und der etwas geringeren Stützweite in Abbildung 6.13 zurückzuführen.

Zum reinen Verständnis des Tragverhaltens ist das Modell aus Abbildung 6.13 sehr gut geeignet. Die Vorstellung von Feld- und Stützmoment analog zum Zweifeldbalken ist hier hilfreich. In der schematischen Darstellung in Abbildung 6.16 wird die Tragwirkung der Zweifeldscheibe durch das Stabwerk verdeutlicht.

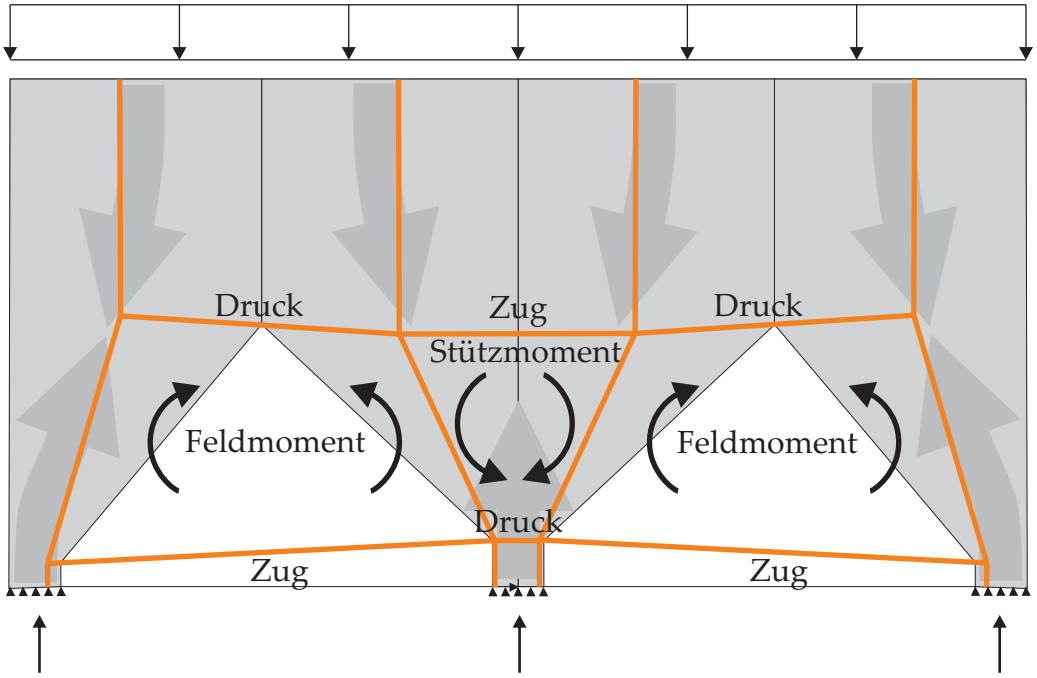


Abbildung 6.16: Schematische Darstellung der Tragwirkung der Zweifeldscheibe

Wie beim vorigen Beispiel bilden sich auch hier vertikale Lastpfade aus. Diese grau hinterlegten Bereiche weisen hohe vertikale Spannungen auf, die von den Knotenlasten ausgehen und eine Konzentration zu den Auflagern hin zeigen. Im oberen Drittel der Scheibe wird der nahezu homogene Spannungszustand wiedergegeben.

Die vertikalen Lastpfade erfahren an zwei wesentlichen Orten Umlenkungen, die durch die Anordnung horizontaler Gleichgewichtskräfte ermöglicht werden. Am unteren Tragwerksrand bildet sich daraus eine geneigte Zugstrebe in den Feldern, die am mittleren Auflager eine höhere Lage einnimmt. Dort wechselt das Vorzeichen der Stabkraft. Grund dafür ist das sich über dem mittleren Auflager einstellende Stützmoment.

Die obere Umlenkung ist etwa in halber Tragwerkshöhe und weist in beiden Feldern Druckspannungen auf, über der Mittelstütze die zu erwartende Zugspannung. Die Geometrie des entstandenen Stabwerkes ist durch die Methode eindeutig definiert und stellt wegen der Ausrichtung an den Trajektorien eine mögliche optimierte Gleichgewichtslage zur Lastabtragung dar.

Eine feinere Modellierung der Gleichlast mit acht Einzellasten verifiziert die Erkenntnisse über das Tragverhalten der Scheibe (vgl. Abbildung 6.17). Aufgrund der Übersichtlichkeit wird hier auf die Darstellung der Kräfte und Stabbreiten verzichtet.

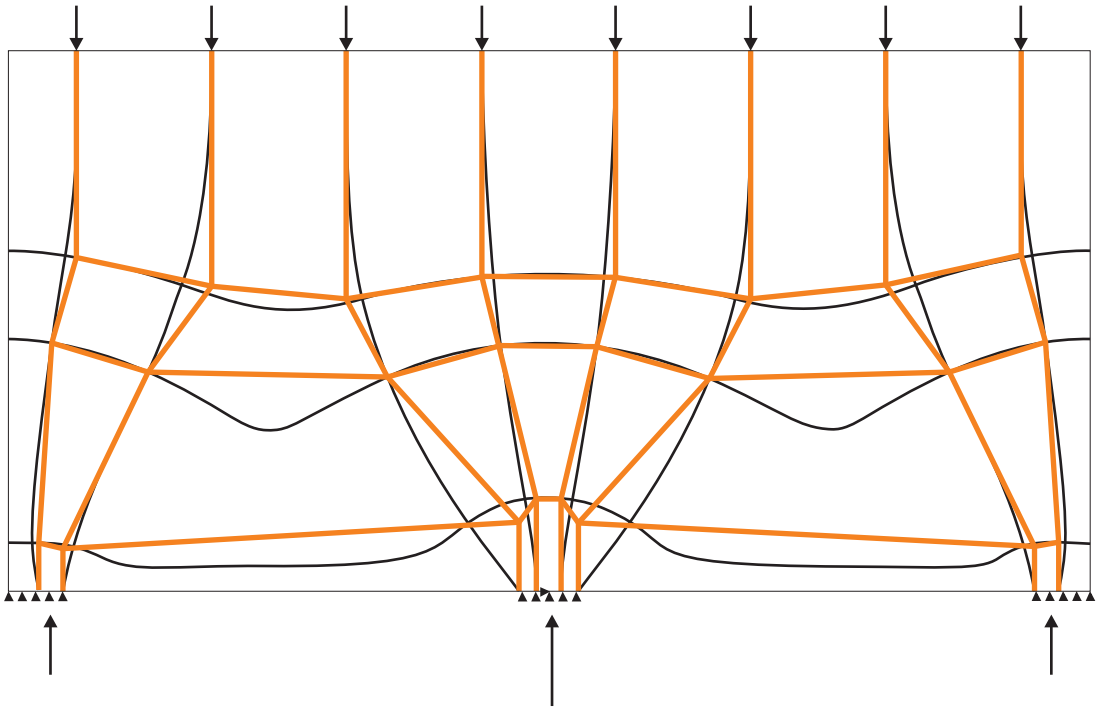


Abbildung 6.17: Lastpfade der Zweifeldscheibe bei der Modellierung mit acht Einzellasten

Es bleibt festzuhalten, dass durch die Methode zur Bestimmung optimierter Stabwerkmodelle eine sehr gute Aussage über deren Geometrie getroffen werden kann. Durch Einhaltung der Kompatibilität von Stabwerk und kontinuierlichem Problem sind auch qualitativ gute Ergebnisse zu erwarten. Das Tragverhalten wird damit konsistent beschrieben.

6.4 Punktgelagerte Scheibe unter Einzellasten

Ein weiteres Beispiel für statisch unbestimmte Stabwerkmodelle ist durch folgende zweiseitig punktgelagerte Scheibe unter zwei Einzellasten gegeben. In Abbildung 6.18 sind Geometrie und Randbedingungen gezeigt.

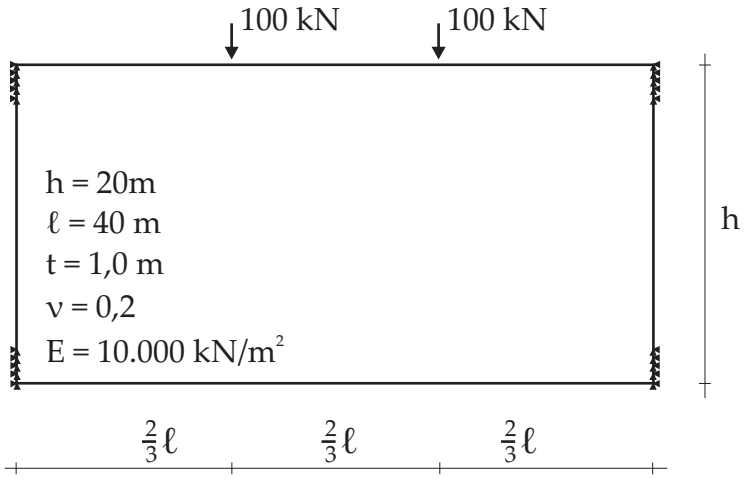


Abbildung 6.18: System und Belastung der zweiseitig gelagerten Scheibe

In Abbildung 6.19 links sind die Kraftflusspfade und das sich daraus ergebende statisch unbestimmten Stabwerkmodell gegeben.

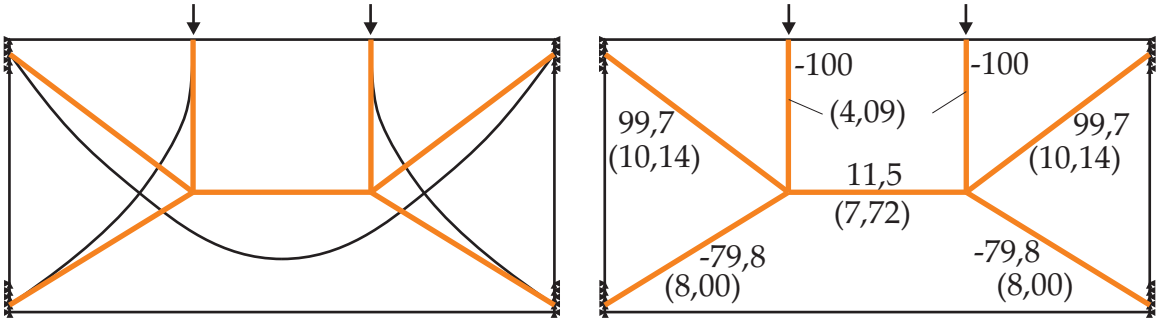


Abbildung 6.19: Kraftflusspfade und Stabwerkmodelle der zweiseitig gelagerten Scheibe

Betrachtet man hier nur die linke Hälfte des symmetrischen Systems, zeichnet sich eine vergleichbare Lösung wie bei der Kragscheibe in Abschnitt 6.5.2 ab. Bei einer feineren Diskretisierung der sekundären Kraftflusspfade ist die Ähnlichkeit noch signifikanter (Abbildung 6.20)

Der einzige Unterschied zeigt sich in der horizontalen Verbindung der beiden Hälften, durch die das Stabwerkmodell statisch unbestimmt wird. Dadurch können die Stabkräfte nur über die korrekte Verteilung der Steifigkeiten und damit über die Verformung berechnet werden.

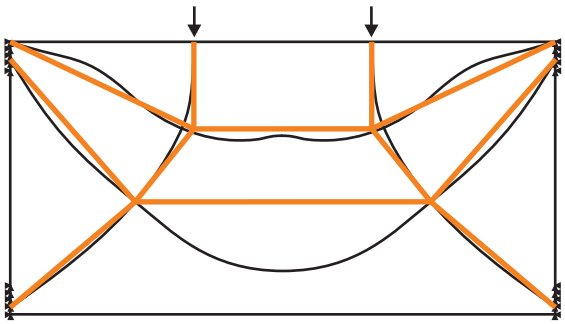


Abbildung 6.20: Kraftflusspfade bei feinerer Diskretisierung

Neben den Stabkräften sind in Abbildung 6.19 rechts die äquvalenten Stabbreiten aus der Minimierung der Differenzarbeit (Klammerausdrücke in [m]) angegeben.

Ein Vergleich der Knotenverformungen des Stabwerkes und der Deformation der Scheibe an diesen Knoten ergeben deckungsgleiche Werte. Das Stabwerk als diskretes Modell bildet damit das kontinuierliche Scheibenproblem zumindest an diskreten Punkten exakt ab. Die Stabkräfte stellen ein Maß für die Beanspruchung der ersetzten kontinuierlichen Bereiche dar. Im Zustand II (gerissener Beton in Zugbereichen) kann die Bewehrung nach der Zugkraft in den Stäben bemessen werden.

6.5 Qualitative Beispiele

Im Folgenden werden qualitative Untersuchungen an einer quadratischen Scheibe mit unterschiedlichen Lasten und Lagerungsbedingungen angestellt und Stabwerkmodelle aus den gefundenen Kraftflusspfaden gezeigt. Der linear elastischen Berechnung liegt ein Finite–Elemente–Netz von $20x20$ Elementen zugrunde. Bedingt durch die Verwendung von rein verschiebungsformulierter Elementen werden Einspannungen durch unverschiebliche Lagerung aller Knoten der betreffenden Kante modelliert. Die Ergebnisse sind auch auf Konsolen unter Berücksichtigung der korrekten Anschlüsse anwendbar.

6.5.1 Kragscheibe unter Gleichlast bzw. Einzellast

In der folgenden Studie wird das Tragverhalten einer Kragscheibe im Übergang von einer Einzellast zu einer Gleichlast mit Hilfe der Kraftflusspfade schrittweise verdeutlicht. Abbildung 6.21 zeigt die volleingespannte Scheibe unter den beiden Grenzbelastungen.

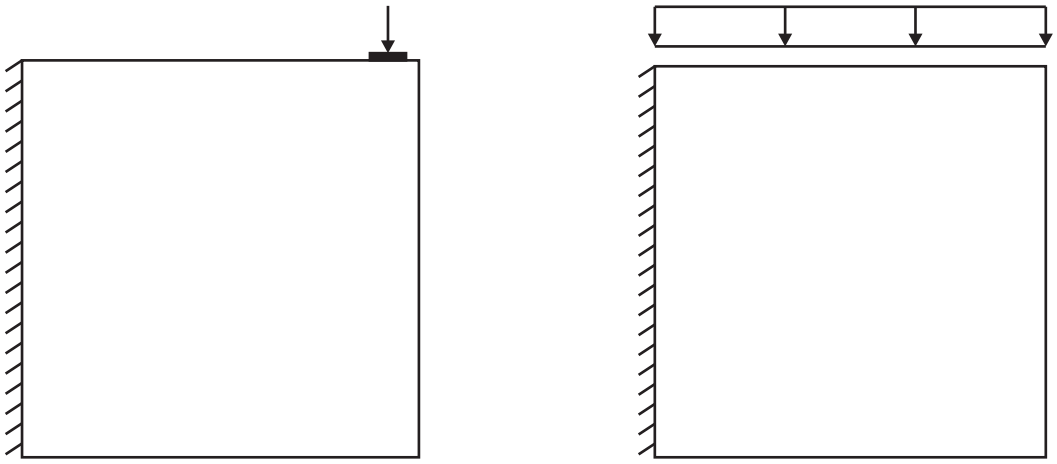


Abbildung 6.21: Kragscheibe unter Gleichlast und Einzellast

Zunächst wird der Kragarm unter Einzellast betrachtet. In Abbildung 6.22 sind zwei unterschiedliche Diskretisierungsgrade zu sehen. Daraus ist zu entnehmen, dass der Kraftflusspfad der Last – in der Grenzbetrachtung – durch kontinuierliche Umlenkung in das Auflager abgeleitet wird. Diskret gesehen bilden sich einzelne Speichen. Zum Vergleich der Kräfte ist auch das betreffende Stabwerkmodell eingezeichnet. Aufgrund der statischen Bestimmtheit des Stabwerkes ist die Verteilung der Steifigkeiten für die Ermittlung der Stabkräfte hier unnötig.

Aus der Abbildung ist eine zunehmende Stabkraft von der Last aus gesehen zum Auflager hin erkennbar. Die Zunahme ist durch die zusätzlichen Kraftkomponenten

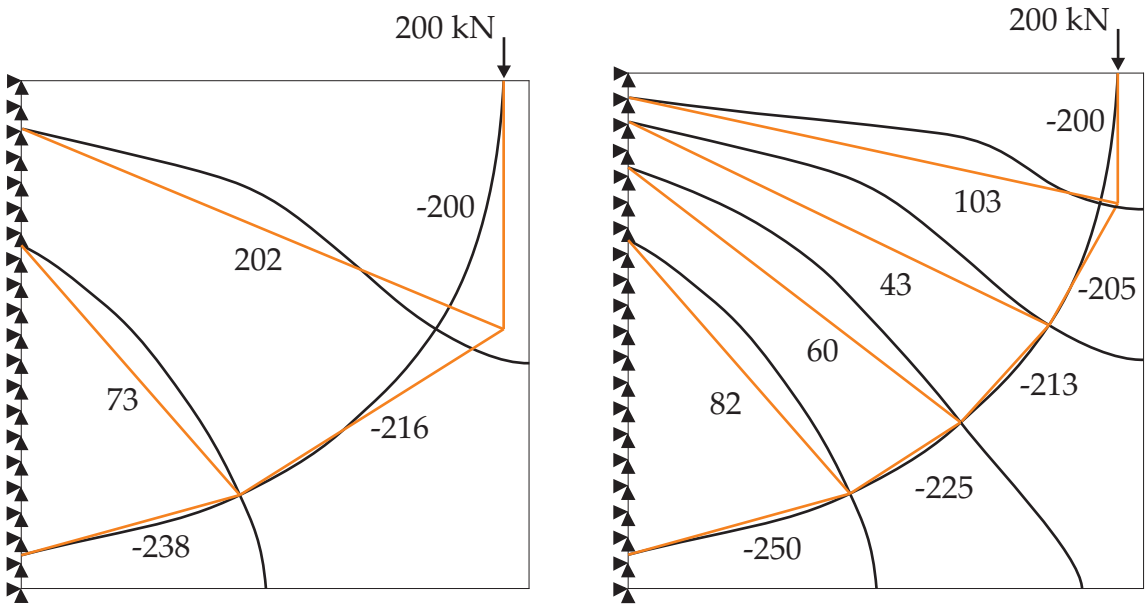


Abbildung 6.22: Stabwerkmodelle mit Unterschiedlichem Diskretisierungsgrad

aus den Speichen zu erklären. Durch die Einzellast ist am Lastangriffspunkt bei der linear elastischen Finite–Elemente–Berechnung eine ausgeprägte Spannungsspitze zu erwarten. Das zeigt sich deutlich durch die ausgeprägte Krümmung des horizontalen Kraftflusspfades innerhalb des St. Venant–Bereiches.

In Erweiterung der Einzellast ist in den Abbildungen 6.23 und 6.24 eine Serie von unterschiedlichen Belastungen des Systems und dessen Kraftflusspfade dargestellt.

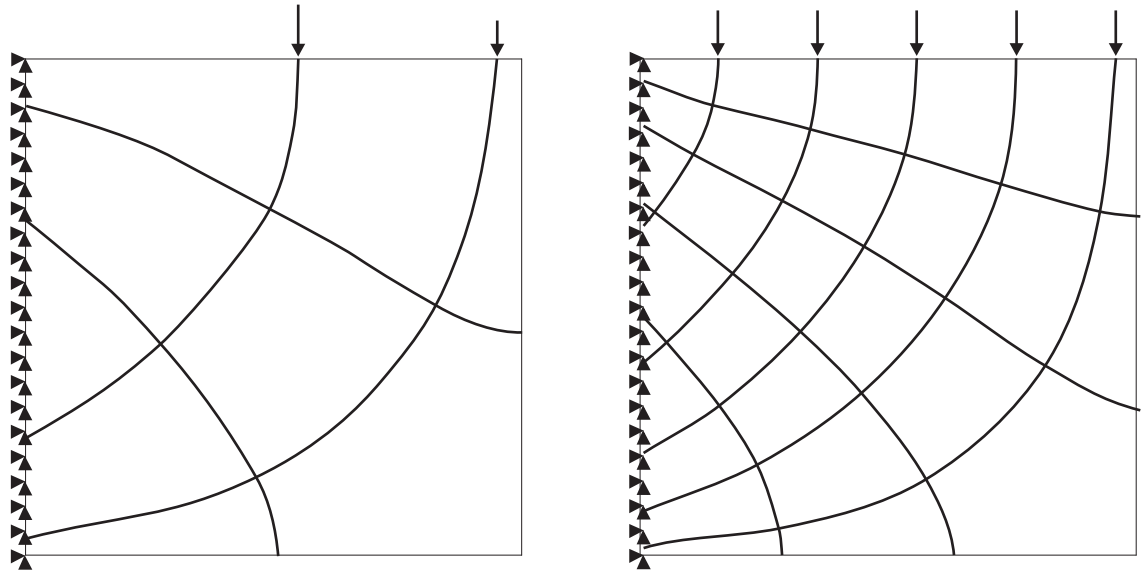


Abbildung 6.23: Kraftflusspfade der Kragscheibe mit zwei bzw. fünf Lasten

Angefangen mit zwei Einzellasten bis hin zur Gleichlast, die hier durch zehn konsistente Knotenlasten approximiert ist, wird der Übergang schrittweise vollzogen.

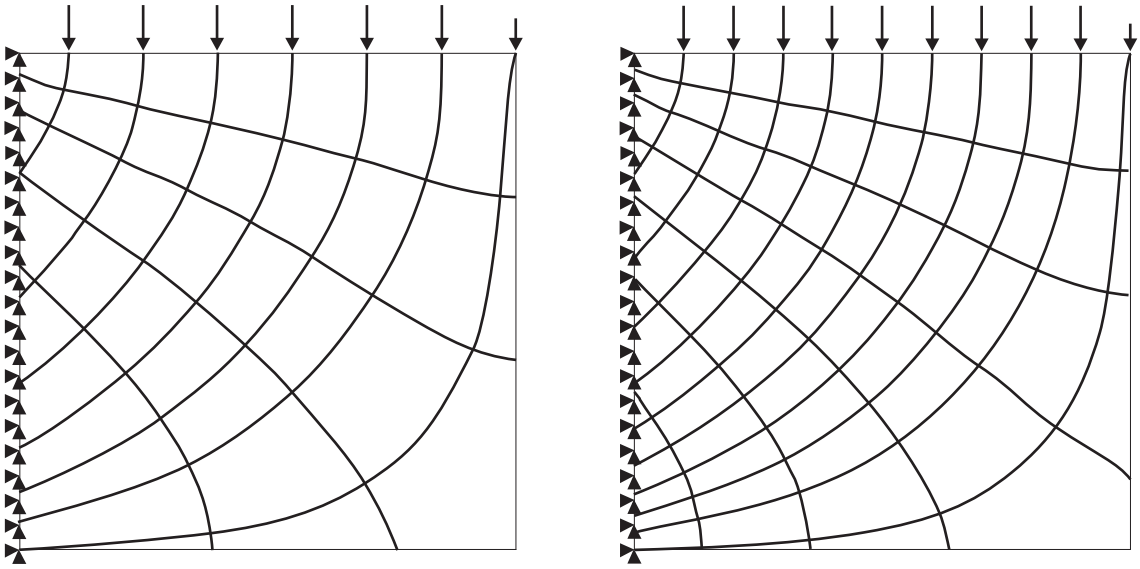


Abbildung 6.24: Kraftflusspfade der Kragscheibe mit sieben bzw. zehn Lasten

Gut zu erkennen ist, dass sich alle Zustände aus der Superposition von „Einzellastlösungen“ zusammensetzen lassen. Anders ausgedrückt kann eine Gleichlast auf einem Kragarm oder einer Konsole durch beliebig viele konsistente Einzellasten ersetzt werden. Die Kraftflussberechnung beschreibt hier das Tragverhalten des jeweilig diskretisierten Problems sehr gut, was in Abbildung 6.25 nochmals prinzipiell für eine Einzellast veranschaulicht ist.

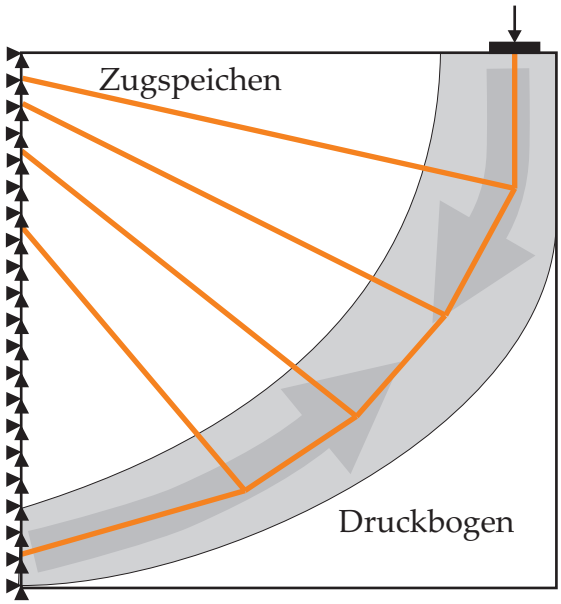


Abbildung 6.25: Schematisches Tragverhalten der Kragscheibe unter Einzellast

Von der Last ausgehend stellt sich ein Druckbogen zur Einspannung hin verlaufend ein. Die entsprechenden Zugkräfte für dessen Umlenkung werden von den „Speichen“ übernommen und ebenfalls ins Auflager abgeleitet.

6.5.2 Scheibe mit punktuellen Lagern und Einzellast

In Anlehnung an die in Abschnitt 3.3.6 erwähnten Michellstrukturen werden im Folgenden zwei punktgestützte Scheiben unter Einzellast betrachtet (vgl. Bild 6.26). Bei der Modellierung der Lager sei auf Abschnitt 6.1 verwiesen. Aufgrund verfahrenstechnischer Gründe ist die Einzellast im rechten Beispiel durch zwei Kräfte mit selber Wirkung aufgeteilt.

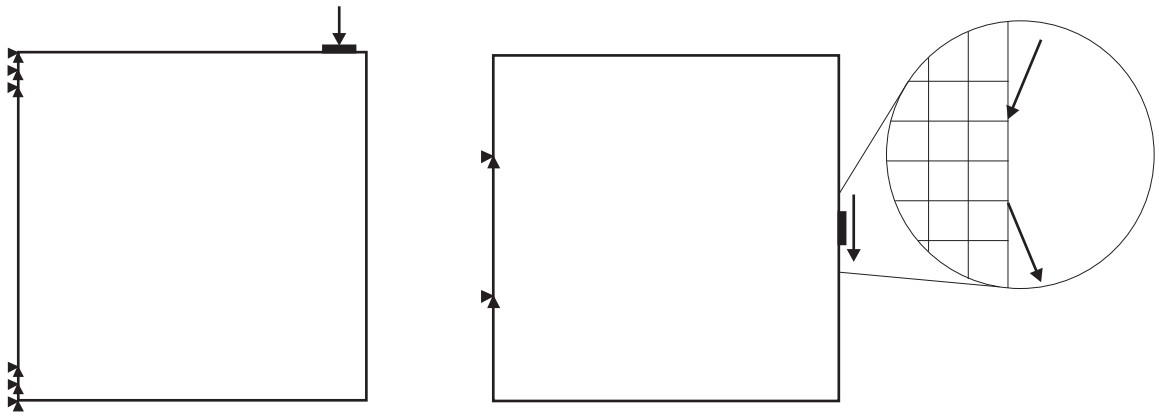


Abbildung 6.26: Punktuell gestützte Scheibe mit unterschiedlichen Einzellasten

Die Berechnung des Kraftflusses des linken Beispiels, dargestellt in Abbildung 6.27, zeigt eine ausgeprägte Speichenform, vergleichbar mit der vorher gezeigten Kragscheibe unter Einzellast. Eine feinere Diskretisierung (im Bild rechts) zeigt auch hier wieder den Übergang zur kontinuierlichen Lösung.

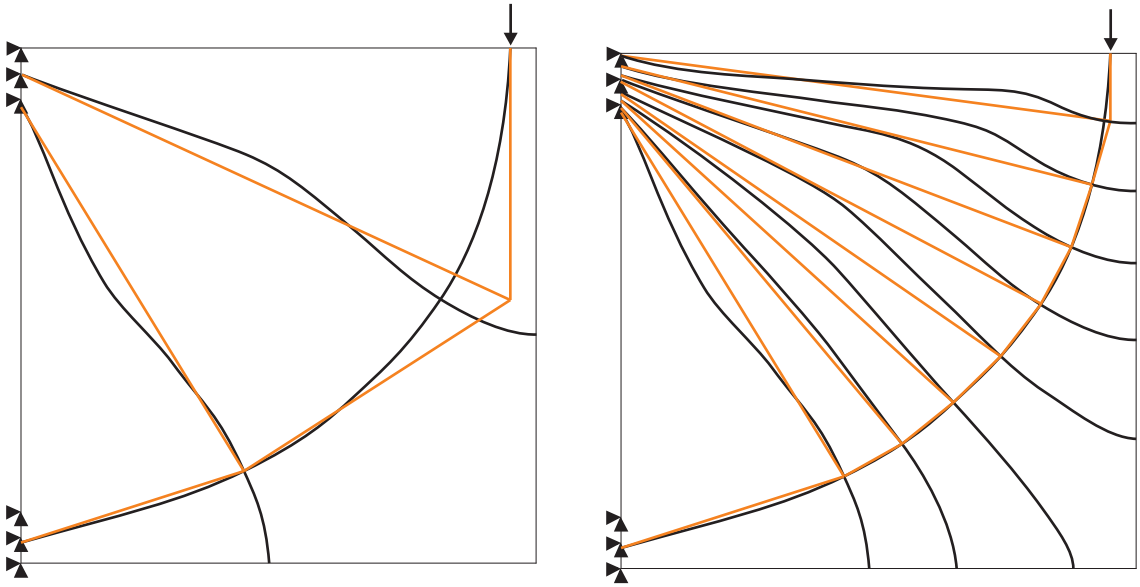


Abbildung 6.27: Speichenform der Kraftflusspfade bei der Scheibe mit Einzellast

Ein Vergleich mit der entsprechenden Michellstruktur z.B. aus Hemp [Hem73] oder Wiedemann [Wie96] in Abbildung 6.28 zeigt nahezu identische Ergebnisse.

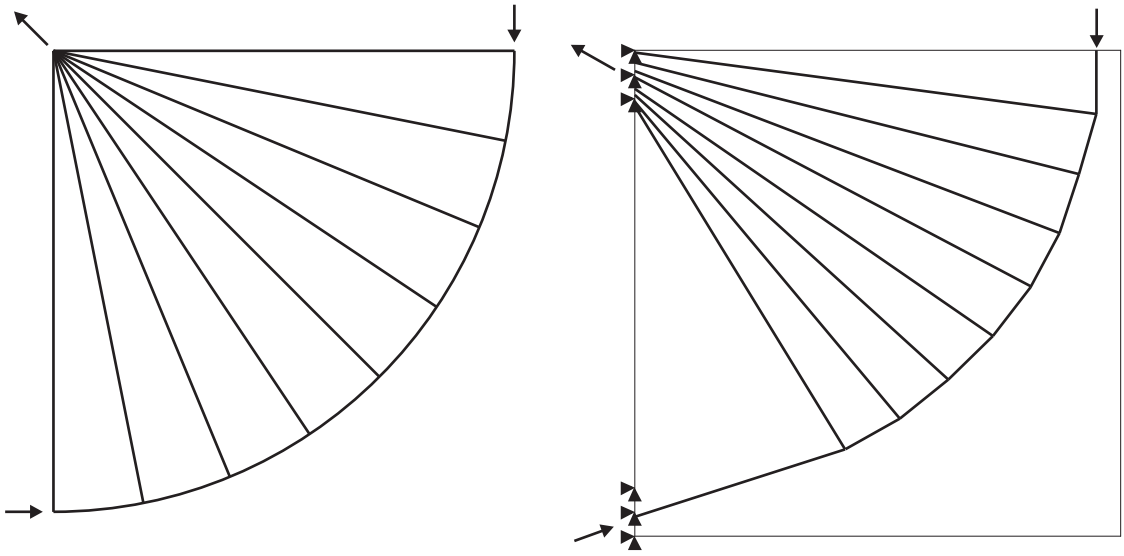


Abbildung 6.28: Vergleich zwischen Michellstruktur und Stabwerkmodell

Die Michellstruktur richtet sich an den orthogonalen Hauptverzerrungstrajektorien eines idealisierten theoretischen Systems aus, während sich die Stabwerke an den Hauptspannungstrajektorien eines real berechneten Systems orientieren. Da die Finite–Elemente–Berechnung das idealisierte System nur annähert, sind die entstehenden Kraftflusspfade nicht ganz deckungsgleich mit den von Michell vorgeschlagenen Hauptverzerrungstrajektorien (vgl hierzu Abschnitt 3.3.6). Abweichungen an Orten mit stark ausgeprägten Spannungsspitzen sind zu erwarten.

Dieses Ergebnis spiegelt sich auch im letzten Beispiel mit der antisymmetrischen Laststellung wieder. Die theoretisch zu erwartende antisymmetrische Kraftflussverteilung, wie sie links in Abbildung 6.29 in der Michellstruktur dargestellt ist, wird durch die Kraftflussberechnung gut wiedergegeben.

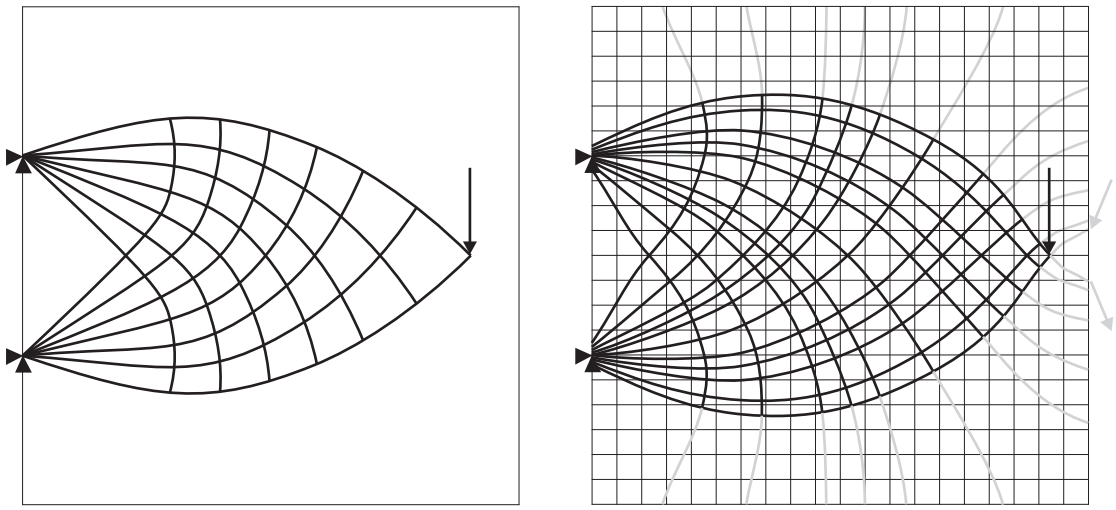


Abbildung 6.29: Michellstruktur

Kapitel 7

Zusammenfassung und Ausblick

In einem wesentlichen Teil der vorliegenden Arbeit ist eine neuen Methode zur Generierung optimierter Stabwerkmodelle von scheibenartigen Tragwerken beschrieben. Ausgehend von einer Finite–Elemente–Berechnung und dem daraus ermittelten Spannungsfeld werden Kraftflusspfade ermittelt, aus denen Stabwerkmodelle generiert und anschließend berechnet werden. Das methodische Vorgehen zur Ermittlung von Stabwerkmodellen ist bis auf einen Steuermechanismus zur Regelung der Dichte der Kraftflusspfade automatisiert. Es ist denkbar, auch diesen interaktiv veränderbaren Parameter durch einen geeigneten Algorithmus zu ersetzten. Jeder Krümmungswechsel unterteilt die primären Kraftflusspfades in Bereiche, in denen ein sekundärer Kraftflusspfad zum Erhalt des Gleichgewichts notwendig ist. Die Lage kann durch den Schwerpunkt des zum primären Kraftfluss orthogonalen Spannungsverlauf ermittelt werden.

Eine besondere Herausforderung stellen statisch unbestimmte Stabwerkmodelle dar, die nicht selten bei der Beschreibung des Tragverhaltens von scheibenartigen Systemen auftreten. Deren Stabkräfte sind nur durch Kenntnis der richtigen Steifigkeitsverhältnisse zu berechnen. Unter Einhaltung der Kompatibilität in den Verformungen zwischen diskretem Stabwerkmodell und kontinuierlicher Scheibenlösung kann eine realistische Aussage über die Steifigkeitsverhältnisse der Stäbe getroffen werden. Dies geschieht durch lösen des betreffenden Optimierungsproblems bei dem das Minimum der Arbeit der Verschiebungsdifferenzen, unter Wahrung des Gleichgewichts, gefordert wird. Zur Bestimmung der Steifigkeiten der Stäbe wurde eine effektive Methode auf Elementebene vorgestellt. Die Erweiterung auf beliebige Problemstellungen auch für räumliche Systeme ist denkbar.

Die Ergebnisse der Kraftflussberechnungen konnten die in der Literatur beschriebenen Stabwerke in ihrer Grundstruktur sehr gut verifizieren. Darüber hinaus wurde durch die beschriebene Methode der Übergang von einer kontinuierlichen Lösung zu einem diskreten Stabwerken plausibel dargestellt.

Durch die Methode zur Bestimmung von Kraftflusspfaden ist ein Modul geschaffen, welches auch in anderen Bereichen eingesetzt werden kann. Beispielsweise in der Topologieoptimierung, die sich mit der optimalen Verteilung von Material in

einem Entwurfsraum beschäftigt. Dies geschieht in der Regel durch eine Dichtefunktion, die jedoch meist neben eindeutigen Bereichen mit bzw. ohne Material auch unscharfe, schwer zu beherrschende Bereiche markiert. Gerade dort könnten zusätzliche Informationen über den Kraftfluss weiterhelfen.

Die Visualisierung der Ergebnisse war bei der Kraftflussberechnung eine besondere Herausforderung. Da gleichzeitig die Portierung des zentralen Berechnungsprogramms anfiel, musste die komplette Grafik neu geschaffen werden. Mit den vorhandenen Graphik–Bibliotheken wäre dies nur mit größerem Aufwand möglich gewesen, weswegen mit AVS/Express auf ein bestehendes Visualisierungsmodul zurückgegriffen wurde. Bei der Umsetzung der beschriebenen Kraftflussberechnung bestand zudem die Aufgabe in der Automatisierung einzelner, hintereinander ablaufender Teilaufgaben durch verknüpfen von bestehenden Komponenten und deren interaktiver Steuerung.

In einem weiteren Teil der Arbeit wurde deswegen ein allgemeingültiges und effektives Konzept für ein Softwaresystem beschrieben, welches die Kopplung von Prozessen in einem verteilten System ermöglicht. Dafür war eine Kapselung bestehender Softwaremodule mit Anbindung an adaptive Kommunikationsschnittstellen notwendig. Auf Basis eines Client–Server–Modells erfolgt der Daten- und Informationsaustausch untereinander mittels TCP/IP.

Die Verwendung der bestehenden und ausgereiften Softwarekomponenten war ohne einen gravierenden Eingriff möglich und stellt damit eine effektive Methode dar, hohe Qualität bei geringer Entwicklungszeit zu gewährleisten. Durch den modularen Aufbau kann das System ohne weiteres erweitert oder Teile davon ersetzt werden. Damit ist eine Möglichkeit zur schnellen Entwicklung von Spezialsoftware zur Lösung individueller Probleme geschaffen.

Durch den zusätzlichen Einsatz eines Workflow–Management–Systems konnten geeignete Module und Objekte, die auf beliebigen Rechnern verteilt sind, während der Laufzeit miteinander zu einem Softwaresystem verknüpft und die Ablaufintegrität sichergestellt werden. Eine Änderung am Quellcode mit dem notwendigen *compilieren* und *linken* war dabei nicht mehr notwendig.

In die Zukunft blickend sind intelligente Workflow–Management–Systeme denkbar, die bei Bedarf automatisch die am wenigsten ausgelasteten Rechner im Netzwerk und deren Prozesse ansteuern, um eine möglichst effektive Nutzung der verteilten Anwendung zu gewährleisten. Workflow–Management–Systeme als Steuereinheiten eröffnen den Weg zu weltweiter Kooperationssoftware, da weder Quellcode portiert, noch ein einheitliches Softwaresystem an einem Rechner zusammengebunden werden muss. Das steigert nicht nur die Produktivität, sondern verringert auch deutlich die zeitliche Verzögerung von Forschungsergebnissen bis hin zur Anwendung in der Praxis. Eine Globalisierung und die überdisziplinäre Zusammenarbeit kann dadurch deutlich erleichtert werden. Die Kooperation großer Softwarefirmen

auch im Bereich der Ingenieuranwendungen spiegelt diesen Trend jetzt schon wider. Das zeigt, wie wichtig die Verfolgung neuer Paradigmen ist, insbesondere für schon lange etablierte Lösungskonzepte.

Langfristig ist damit zu rechnen, dass vereinzelte Anbieter nicht mehr vollständige Softwaresysteme entwickeln werden, sondern nur noch sehr spezielle Einzelkomponenten. Diese werden dann auf Basis standardisierter Kommunikationssoftware an bestehende Systeme angebunden. Die *Object Management Group* hat mit CORBA einen Technologiestandard festlegt, mit der objektorientierte Systeme miteinander kommunizieren können. Zwar sind derartige Standards in vielen Bereichen von Vorteil oder gar notwendig, jedoch sind damit auch Einschränkungen verbunden, weswegen ein derartiges System hier nicht zur Anwendung kam.

Wenn Softwaresysteme über öffentlich zugängliche Netzwerke miteinander verbunden werden, ist die Sicherheit gegen Hackerangriffe ein wesentlicher Aspekt, der jedoch aus Gründen des Umfangs in dieser Arbeit keine Berücksichtigung fand.

Literaturverzeichnis

[Alt94] J. ALTENBACH / H. ALTENBACH: *Einführung in die Kontinuumsmechanik*. Teubner Studienbücher Mechanik, 1994.

[Arn98] CARSTEN A. ARNHOLM: *Portable mixed laguage programming using C++ and Fortran*, Band 477. C/C++ User Group Libary CD-ROM, Web Addresse: http://www.hal9k.com/cug/, 1998.

[AVS] AVS: *Advanced Visual Systems Inc. (Homepage 2005)*. Web Adresse: http://www.avs.com.

[Bar94] J.J. BARTON / L.R. NACKMAN: *Sientific and Engineering C++: An Introduction With Advanced Techniques and Examples*. Addison Wesley Longman Publishing Co., Inc., Boston, MA, 1994.

[Bat86] KLAUS-JÜRGEN BATHE: *Finite–Elemente–Methoden*. Springer-Verlag, New York, Tokyo, Berlin, Heidelberg, 1986.

[Bel01] T. BELYTSCHKO / W.K. LIU / B. MORAN: *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, LTD, Chichester, 2001.

[Bir84] BIRELL / NELSON: *Implementing Remote Procedure Calls*. ACM Trans. Comp. Syst., Bd. 2, Nr. 1, S. 39-59, Feb. 1984.

[Bis04] BISCHOFF / WALL / BLETZINGER / RAMM: *Encyclopedia of Computational Mechanics*. Wiley, 2004.

[Ble90] KAI-UWE BLETZINGER: *Formoptimierung von Flächentragwerken*. Dissertation, Institut für Baustatik, Universität Stuttgart, 1990.

[Bor95] BORGHOFF / SCHLICHTER: *Rechnergestützte Gruppenarbeit: Eine Einführung in Verteilte Anwendungen*. Springer-Verlag, Berlin, Heidelberg, 1995.

[Bro01] BRONSTEIN / SEMENDJAJEW / MUSIOL / MÜHLIG: *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2001.

[Com91] DOUGLAS E. COMER: *Internetworking with TCP/IP: Principles, Protocols and Architecture*, Band 1. Englewood Cliffs, NJ: Prentice-Hall, 2. Auflage, 1991.

[Com93] DOUGLAS E. COMER: *Internetworking with TCP/IP: Principles, Protocols and Architecture*, Band 3. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[Coo95] ROBERT D. COOK: *Finite Element Modeling for Stress Analysis*. Wiley, 1995.

[Cox58] H. L. COX: *The Theory of Design*. Aeronautical Research Coucil, 1958.

[Dij65] EDSGER W. DIJKSTRA: *Cooperating Sequential Processes*. Technological University, Eindhoven, 1965.

[Eck00] BRUCE ECKEL: *Thinking in C++: Volume One: Introduction to Standard C++*. Prentice Hall, 2000.

[Eck03] BRUCE ECKEL / CHUCK ALLISON: *Thinking in C++: Volume Two: Practical Programming*. Prentice Hall, 2003.

[Eur92] EUROCODE 2: *Planung von Stahlbeton- und Spannbetontragwerken - Teil 1 - Grundlagen und Anwendungsregeln für den Hochbau. Europäische Vornorm ENV 1992-1-1*. 1992.

[Fis99] S. FISCHER / W. MÜLLER: *Netzwerkprogrammierung unter LINUX und UNIX*. Carl Hanser Verlag, München, Wien, 2. Auflage, 1999.

[Fon95] JOAO FONSECA: *Zum Bemessen und Konstruieren von Stahlbetonplatten und -scheiben mit Lastpfaden*. Dissertation, Institut für Tragwerksentwurf und -konstruktion, Universität Stuttgart, 1995.

[GAU] GAUG: *Deutsche AVS–Nutzergruppe (Homepage 2005)*.
Web Adresse: http://www.uni-ulm.de/gaug/.

[Gra91] E. GRASSER / G. THIELEN: *Hilfsmittel zur Berechnung der Schnittgrößen und Formänderungen von Stahlbetontragwerken*. Deutscher Ausschuß für Stahlbeton, Heft 240, 3. überarbeitete Auflage, 1991.

[Haf99] RAPHAEL T. HAFTKA / ZAFER GÜRDAL: *Elements of Structural Optimization*. Kluwer Academic Publishers, Dordrecht, Boston, London, 3. Auflage, 1999.

[Haj90] RADE HAJDIN: *Computerunterstützte Berechnung von Stahlbetonscheiben mit Spannungsfeldern*. Technischer Bericht : Institut für Baustatik und Konstruktion, ETH Zürich, Birkhäuser Verlag: Basel, Boston, Berlin, 1990.

[Hal01] F. HALSALL: *Multimedia Communications: Applications, Networks, Protocols and Standards*. MA: Addison-Wesley, 2001.

[Han73] P. BRINCH HANSEN: *Operating System Principles*. Prentice-Hall, Englewood Cliffs, N.J., 1973.

[Har02] HARTMANN / KATZ: *Statik mit Finiten Elementen*. VDI–Buch. Springer Verlag, 2002.

[Hem58] W. S. HEMP: *Theory of Structural Design*. Nr. 115. Rep. Coll. Aeronaut., Cranfield, 1958.

[Hem64] W. S. HEMP: *Studies in the Theory of Michell Structures*. München, 1964. International Congress of applied Mechanics.

[Hem73] W. S. HEMP: *Optimum Structures*. Oxford Engineering Science Series. Clarendon Press, Oxford, 1973.

[Hol00] GERHARD A. HOLZAPFEL: *Nonlinear Solid Mechanics*. John Wiley & Sons, Ltd., Chichester, 2000.

[Hug00] THOMAS J. R. HUGHES: *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Inc., Mineola, 2000.

[Hun03] CRAIG HUNT: *TCP/IP - Netzwerkadministration*. O'Reilly, Köln, 3. Auflage, 2003.

[IAC] IAC: *International AVS Centre (Homepage 2005)*.
Web Adresse: http://www.iavsc.org.

[Jen89] M. JENNEWEIN: *Ein Beitrag zum Verständnis der Lastabtragung und des Tragverhaltens von Stahlbetontragwerken*. Dissertation, Institut für Massivbau, Universität Stuttgart, 1989.

[Ker90] KERNIGHAN / RITCHIE: *Programmieren in C*. Carl Hanser Verlag, München Wien, Zweite Auflage, 1990.

[Lan94] H. LANGENDÖRFER / B. SCHNOR: *Verteilte Systeme*. Anwendungsorientierte Informatik. Carl Hanser Verlag, München, Wien, 1994.

[Leo77] FRITZ LEONHARDT: *Vorlesungen über Massivbau*. Springer Verlag, Berlin, 1977.

[Lip00] STANLEY B. LIPPMAN: *Essential C++*. C++ in depth. Addison-Wesley Longman, Amsterdam, 2000.

[Mal69] LAWRENCE E. MALVERN: *Introduction to the Mechanics of a Continuous Medium*. Prentice Hall, Englewood Cliffs, N.J., 1969.

[Mar83] J.E. MARSDEN / T.J.R. HUGHES: *The Mathematical Foundation of Elasticity*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[Mat82] JÜRGEN MATTHEISS: *Platten und Scheiben*. Werner-Verlag, Düsseldorf, 1982.

[Mau98] KURT MAUTE: *Topologie- und Formoptimierung von dünnwandigen Tragwerken*. Dissertation, Institut für Baustatik, Universität Stuttgart, 1998.

[Max69] C. MAXWELL: *Scientific Papers II*. Seite 175. Cambridge University Press, 1869.

[Mey97] BERTRAND MEYER: *Object-oriented Software Construction*. Prentice Hall, 2. Auflage, 1997.

[Mic04] A. G. M. MICHELL: *The Limit of Economy of Material in Frame Structures*. Phil. Mag., 8:589 – 597, 1904.

[Mör12] EMIL MÖRSCH: *Der Eisenbetonbau, seine Theorie und Anwendung*. Wittwer, Stuttgart, 1912.

[Mow99] T. MOWBRAY / R. ZAHAVI: *The Essential CORBA: Systems Integration Using Distributed Objects*. John Wiley & Sons Inc., 1999.

[Müh92] M. MÜHLHÄUSER / A. SCHILL: *Software Engineering für verteilte Anwendungen: Mechanismen und Werkzeuge*. Springer-Verlag, Berlin, 1992.

[Mül00] MÜLLER/GROTH: *FEM für Praktiker – Band 1: Grundlagen*. Expert Verlag, 5. Auflage, 2000.

[Mut88] A. MUTTONI / J. SCHWARTZ / B. THÜRLIMANN: *Bemessen und Konstruieren von Stahlbetontragwerken mit Spannungsfeldern*. Vorlesung Stahlbeton AK, Institut für Baustatik und Konstruktion, ETH Zürich, 1988.

[Neh85] JÜRGEN NEHMER: *Softwaretechnik für verteilte Systeme*. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

[OON] OON: *Object-Oriented-Numerics (Homepage 2004)*.
Web Adresse: http://www.oonumerics.org.

[Pro02] W. E. PROEBSTER: *Rechnernetze: Technik, Protokolle, Systeme, Anwendungen*. OldenbourghWissenschaftsverlag GmbH, München, 2. Auflage, 2002.

[Rüc92] KLAUS J. RÜCKERT: *Entwicklung eines CAD–Programmsystems zur Bemessung von Stahlbetontragwerken mit Stabwerkmodellen*. Dissertation, Institut für Tragwerksentwurf und -konstruktion, Universität Stuttgart, 1992.

[Rus89] D. RUSSEL: *The Principles of Computer Networking*. Cambridge University Press, Cambridge, England, 1989.

[Sch93] G. SCHLAICH / K. SCHÄFER: *Betonkallender 1993 Teil II: Konstruieren im Stahlbetonbau*. Wilhelm Ernst & Sohn KG Verlag, Berlin, 1993.

[Sch94] KLAUS-JÜRGEN SCHNEIDER: *Bautabellen für Ingenieure*. Werner-Verlag, 11. Auflage, 1994.

[Sch01] U. SCHNEIDER / D. WERNER: *Taschenbuch der Informatik*. Fachbuchverlag Leipzig im Carl Hanser Verlag, München, 4. Auflage, 2001.

[Sha95] WILLIAM A. SHAY: *Understanding Data Communications and Networks*. Technischer Bericht PWS Publishing Company, Boston, 1995.

[Ste00] W. RICHARD STEVENS: *Programmieren von UNIX-Netzwerken*. Carl Hanser Verlag, München, Wien, Zweite Auflage, 2000.

[Str98] BJARNE STROUSTRUP: *Die Programmiersprache C++*. Addison-Wesley-Longman, Bonn, Dritte Auflage, 1998.

[Sun94] WOLFGANG SUNDERMANN: *Tragfähigkeit und Tragverhalten von Stahlbeton-Scheibentragwerken*. Institut für Tragwerksentwurf und -konstruktion, Universität Stuttgart, 1994.

[Tan02] ANDREW S. TANENBAUM: *Verteilte Systeme*. Pearson Education, München, 2002.

[Tan03] ANDREW S. TANENBAUM: *Computernetzwerke*. Pearson Education, Englewood Cliffs, New Jersey, 4. überarbeitete Auflage, 2003.

[Uma93] AMJAD UMAR: *Distributed Computing and Client-Server Systems*. PTR Prentice Hall, Piscataway, New Jersey, 1993.

[Van84] GARRET N. VANDERPLAATS: *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill Book Company, New York, 1984.

[Wei02] MANFRED WEISS: *TCP/IP Handbuch*. Professional Series. Franzis Verlag GmbH, Poing, 2002.

[Wer95] HORST WERKLE: *Finite Elemente in der Baustatik - Band 1: Lineare Statik der Stab- und Flächentragwerke*. Vieweg Verlagsgesellschaft mbH, Braunschweig, Wiesbaden, 1995.

[Wie96] JOHANNES WIEDEMANN: *Leichtbau 2: Konstruktion*. Springer Verlag, Berlin, Heidelberg, New York, 2. neubearbeitete Auflage, 1996.

[Yan01] DAOQI YANG: *C++ And Object Oriented Numeric Computing For Scientists and Engineers*. Springer-Verlag, New York, 2001.

[Zie00] O.C. ZIENKIEWICZ / R.L. TAYLOR: *The Finite Element Method: The Basis*, Band 1. Butterworth Heineman, Oxford, 5. Auflage, 2000.

Anhang A

Client–Server

Im Folgenden ist die vollständige Umsetzung eines Clients und eines Servers abgedruckt. Die Umsetzung ist in C++ auf UNIX und LINUX getestet. Die Datenübermittlung ist nur exemplarisch eingearbeitet und entspricht nicht der eigentlichen Anwendung. Im abgedruckten Quellcode werden Knoten und deren Koordinaten eines Finite–Elemente–Netzes übertragen. Dabei fungiert der Server als Absender und der Client als Empfänger der Daten.

A.1 Server

Zunächst sind einige Bibliotheken zur Kommunikation einzubinden, die jedoch für alle gängigen Betriebsysteme und Programmiersprachen vorhanden sind.

```
#include <sys/types.h>   //Prozessbehandlung
#include <sys/socket.h>  //Socketfunktionen (socket, listen, accept...)
#include <netinet/in.h>  //Socketadressstruktur IPv4
#include <netinet/tcp.h> //TCP-Header-Datei
#include <netdb.h>       //Netzwerkfunktionen (gethostbyname,  ...)
#include <unistd.h>      //für close(connfd)
```

In der Hauptroutine des Servers übernimmt die Funktion `sendnodes()` den Datentransfer. Auf die Darstellung der Routine wird der Übersicht halber verzichtet. Prinzipiell wird ein Container übergeben, der Pointer auf Instanzen einer Klasse namens „Knoten“ beinhaltet. Eine detaillierte Beschreibung des Konzepts zur Datenübermittlung ist in Kapitel 4.6 gegeben.

```
_____________________________________________________________TCP-SERVER
int main(int argc, char **argv)
 {
  vector<node*>& nodevec;            //Kontainer mit Pointer auf Knoten
  int listenfd;    //Socketdeskriptor für horchenden (listening) Socket
  int connfd;     //Socketdeskriptor für verbundenen (connected) Socket
  socklen_t clilen;    //Bytelänge der Socketadressstruktur des Clients
  struct sockaddr_in cliaddr, servaddr;          //Socketadressstruktur
  struct servent *service;    //Datentruktur: TCP-Port und Service-Name


//_____________________________________________________VERBINDUNGSAUFBAU
  int blocksizen=100;  //Blockgröße in der die Knoten übertragen werden
  servaddr.sin_family=AF_INET;//Socketadressstruktur des Servers (IPv4)
  portnummer = atoi(argv[1]);               // Übergabe von Portnummer

//Verbindungen an jeder Netzwerkschnittstelle akzeptieren

  servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

//Ist ein Port fest für einen Dienst (hier:avs) definiert, kann er auch
//durch "get service by name" vom Betriebsystem abgerufen werden.

  if (service = getservbyname("avs","tcp"))
   {servaddr.sin_port = service->s_port;}
  else
   {servaddr.sin_port = htons((u_short)portnummer;}  //well-known-port


_____________________________________________________ERSTELLEN DES SOCKETS
// socket(int family, int type, int protocol)
// family gibt die Protokollfamilie an: AF_INET für Internet
// type ist der Typ des Sockets: SOCK_STREAM steht für Daten-Streams
// protocol ist das Protokoll selbst: hier wird das Standardprotokoll
// für Daten-Streams verwendet (TCP)

  listenfd = socket(AF_INET, SOCK_STREAM, 0);

//Binden der Serveradressstruktur an den Socket

  bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
```

```
//Umwandlung des aktiven in einen passiven, also horchenden Socket und
//warten auf einen eingehenden Verbindungsaufbauwunsch

  listen(listenfd, 1);     //Warteschlange von Clients, hier maximal 1

//Der Server wartet auf den Verbindungswunsch eines Clients. Der erste
//Eintrag aus der Warteschlange erhält einen neuen Socket-Deskriptor
//(connfd) in dem die Clientadresse und deren Länge gespeichert sind.

  clilen = sizeof(cliaddr);//Länge der Socketadressstruktur vom Client
  connfd = accept(listenfd, (struct sockaddr*)&cliaddr, &clilen);


------------------------------------------------------------SENDEN DER DATEN
//Sobald eine Verbindung aufgebaut ist, kehrt accept vom wartenden
//Zustand zurück, und Daten können darüber versendet werden. In diesem
//Fall durch Übergabe eines Kontainers an eine Sende-Funktion.

  sendnodes(nodevec, connfd, blocksizen)


-------------------------------------------------BEENDEN DER VERBINDUNG
  close (connfd);
  return 0;
 }
```

A.2 Client

Zunächst sind einige Bibliotheken zur Kommunikation einzubinden, die jedoch für alle gängigen Betriebsysteme und Programmiersprachen vorhanden sind. Die eigentliche `read()` Anweisung zum Empfang der Daten, wird in zwei Funktionen eingebettet, für einen besseren Umgang mit den Daten–Streams. Neben den Funktions–Deklarationen findet sich deren Definitionen.

```
#include <sys/types.h>   //Prozessbehandlung
#include <sys/socket.h>  //Socketfunktionen (socket, listen, accept...)
#include <netinet/in.h>  //Socketadressstruktur IPv4
#include <netinet/tcp.h> //TCP-Header-Datei
#include <netdb.h>       //Netzwerkfunktionen (gethostbyname,  ...)
#include <unistd.h>      //für close(connfd)

static ssize_t my_read(int fd, char *ptr);
ssize_t readline(int fd, char *ptr, size_t maxlen);



static ssize_t my_read(int fd, char *ptr)
{
  static int  read_cnt = 0;
  static char  *read_ptr;
  static char  read_buf[10000];

  if (read_cnt <= 0) {
again:
    if ( (read_cnt = read(fd, read_buf, sizeof(read_buf))) < 0) {
      if (errno == EINTR)
        goto again;
      return(-1);
    } else if (read_cnt == 0)
      return(0);
    read_ptr = read_buf;
  }
  read_cnt--;
  *ptr = *read_ptr++;
  return(1);
}
```

```
ssize_t readline(int fd, char *ptr, size_t maxlen)
{
  int n, rc;
  char c;
  for (n = 1; n < maxlen; n++) {
    if ( (rc = my_read(fd, &c)) == 1) {
      *ptr++ = c;
      if (c == '\n')
        break;     // neue Zeile, wie fgets()
    } else if (rc == 0) {
      if (n == 1)
        return(0); // EOF, keine Daten gelesen
      else
        break;     // EOF, ein paar Daten wurden gelesen
    } else
      return(-1);  // Fehler
  }
  return(n);
}
```

Die anschließenden Programmabschnitte des Clients für den Aufbau der Verbindung, den Datenaustausch und das Beenden der Verbindung ist im Programmcode gekennzeichnet. Die wesentlichen Befehle sind zusätzlich kommentiert.

Bei der Übertragung von großen Datenmengen ist die Einteilung und das Verschicken in Blöcken sinnvoll. Damit begrenzt sich der Schaden bei Verlust und der erneuten Sendung eines Datenpakets. Diesem Umstand wird in der Beschreibung ebenfalls Rechnung getragen.

```
_____________________________________________________________TCP-CLIENT
int main(int argc, char **argv)
 {
  char linein[1000000];
  int last, blocksize, nodenb, nnodes;
  double x, y, z;

  int sockfd, portnummer;
  char* hostname;
  struct hostent *host_id;
  struct servent *service;
  struct sockaddr_in servaddr;
```

```
  hostname = argv[1];                          // Übergabe von Hostname
  portnummer = atoi(argv[2]);                  // Übergabe von Portnummer

  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons((u_short)portnummer);

//service = getservbyname("avs","tcp"); //Alternative
//servaddr.sin_port = service->s_port;

  host_id = gethostbyname(hostname);
  bcopy(host_id->h_addr, &servaddr.sin_addr, host_id->h_length);


_____________________________________________________ERSTELLEN DES SOCKETS
  sockfd = socket(AF_INET, SOCK_STREAM, 0);

//socket(int family, int type, int protocol)
//family gibt die Protokollfamilie an: AF_INET für die Internet
//type ist der Typ des Sockets: SOCK_STREAM steht für Daten-Streams
//protocol ist das Protokoll selbst: hier wird das Standardprotokoll
//für Streams verwendet (TCP)


_____________________________________________________VERBINDUNGSAUFBAU
  connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr))


_____________________________________________________DATENEMPFANG
//Anzahl aller Knoten und die Blockgröße, in der sie gesendet werden.
  readline(sockfd, linein, 1000)
  istrstream innodeinfo(linein);
  innodeinfo  >> nnodes >> blocksize;

//Knoten und deren Koordinaten x,y,z in ganzen Blöcken
  last = 0;
  if (nnodes>blocksize && blocksize != 0){ // 0: alle Knoten auf einmal
   for(int i=0; i<(nnodes/blocksize);i++){
    readline(sockfd, linein, 100000)
    istrstream innodes(linein);
    for (int j=0; j<blocksize; j++){
     innodes >> nodenb >> x >> y >> z;
     }
```

```
     last = (i+1)*blocksize;
    }
   }

//Die restlichen Knoten und deren Koordinaten x,y,z vom letzten Block
  if (last != nnodes){
   readline(sockfd, linein, 100000)
   istrstream innodes(linein);
   for (int j=last; j<nnodes; j++){
    innodes >> nodenb >> x >> y >> z;
    }
   }

_____________________________________________________BEENDEN DER VERBINDUNG
   close(sockfd);
   return(1);
}
```

A.3 Bibliotheksfunktionen

Funktionen zum Verbindungsaufbau

```
int socket(int family, int type, int protocol);
```
family: Protokollfamilie (Tabelle A.1)
type: Übertragungsart (Tabelle A.2)
protocol: 0 für das Default–Protokoll für den jeweiligen Sockettyp

AF_UNIX	Unix Domain Sockets	AF_APPLETALK	AppleTalk DDP
AF_LOCAL	POSIX für AF_UNIX	AF_INET6	IP Version 6
AF_INET	Internet IP Protocol	AF_IRDA	IRDA Sockets
AF_IPX	Novell IPX	AF_BLUETOOTH	Bluetooth Sockets

Tabelle A.1: Unterstützte Adress–Familien

streamorientiert	SOCK_STREAM
paketorientierten	SOCK_DGRAM

Tabelle A.2: Übertragungsarten

```
int bind(int sockfd, struct sockaddr_in *my_addr, int addrlen);
```
sockfd: Socket–Deskriptor, über den auf die Verbindung zugegriffen wird
sockaddr_in, addrlen: protokollspezifische Adresse des Servers und dessen Länge

```
struct sockaddr_in {
 short int          sin_family; //Adressfamilie normalerweise AF_INET
 unsigned short int sin_port;   //Port der Verbindung
 struct in_addr     sin_addr;   //Adresse zu der verbunden werden soll
 unsigned char      sin_zero[8];//Fülldaten um auf 14 Bytes zu kommen
 };
```

```
int listen(int sockfd, int backlog);
```
sockfd: Socket–Deskriptor
backlog: maximale Anzahl der Verbindungsanfragen in der Warteschlange

```
int accept(int sockfd, void *addr, int *addrlen);
```
sockfd: horchender Socket (Übergabe); verbundener Socket(Rückgabe)
backlog: Adressstruktur des Verbindungspartners (IP–Adresse, Portnummer)

```
int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```
sockfd: Socket–Deskriptor
sockaddr, addrlen: Adressstruktur des Servers und dessen Länge (s.o.)

Funktionen zur Datenübertragung

```
int write(int sockfd, char * data, int datalen);
```

sockfd: Socket–Deskriptor; Anzahl der gesendeten Daten in Bytes, im Fehlerfall -1
data, datalen: Speicheradresse und Länge der Daten

```
int read(int sockfd, char * buffer, int buflen);
```

sockfd: Socket–Deskriptor, Anzahl der gesendeten Daten in Bytes (Rückgabe)
buffer, buflen: Adresse des Datenpuffers und dessen Länge

Funktion zum Beenden einer Verbindung

```
int close(int sockfd);
```

sockfd: zu terminierender Socket–Deskriptor

Auflösen des Rechnernamens durch den DNS–Dienst in seine IP–Adressen:

```
struct hostent* gethostbyname(const char* name);
```

name: Struktur mit Informationen über den Rechnernamen

```
struct hostent {
   char  *h_name;
   char **h_aliases;
   int    h_addrtype;
   int    h_length;
   char **h_addr_list;
   };
```

h_name: offizieller Name des Rechners
haliases: eine Liste von Aliasnamen des Hosts
h_addr_list: Liste der IP-Adressen des Rechners

Auflösen der IP–Adresse und Portnummer eines Prozesses:

```
int getpeername(int sockfd, struct sockaddr *peer, int *addr_len);
```

sockfd: Socketdeskriptor
sockaddr, addr_len: Adresse des Kommunikationspartners und deren Länge

Anhang B

Mixed Language Programming

Zwei Faktoren, nämlich der Compiler und der passende Linker, sind für eine Verbindung von gemischt–sprachigem Quellcode von wesentlicher Bedeutung. Folgende Betrachtungen gelten für die Verwendung von LINUX auf einem PC X86 mit dem Compiler und Linker von GNU. Die Anwendung der MLP sollte allerdings auf alle Systeme mit geringen Modifikationen auf die selbe Weise möglich sein.

Bei dem hier betrachteten Beispiel werden drei Quellcode Dateien mit Programmcode in FORTRAN, C und C++ miteinander verbunden. Zunächst werden die einzelnen Quellcodes mit den jeweiligen Compilern in Objektdateien übersetzt (compilieren). Das Binden der Objektdateien (linken) erfordert nun gewisse Aufmerksamkeit und kann auf zwei unterschiedliche Weisen erfolgen.

Bei Verwendung des g++ Linkers werden alle notwendigen Bibliotheken für das Objektdateien von AnsiC und C++ automatisch hinzugebunden. Die für die FORTRAN Objektdatei notwendige Bibliothek *libg2c.a* muss mit dem Compilerflag *-lg2c* hinzugefügt werden.

Wird der g77 Linker verwendet, erfolgt die Anbindung der nötigen FORTRAN Bibliotheken automatisch. Hier muss die erforderliche Bibliothek *libstdc++.a* für den AnsiC und den C++ Code durch den Compilerflag *-lstdc++* hinzugefügt werden.

Folgende Beobachtungen sind dabei Erwähnenswert. Wird der FORTRAN Code vom g77 Compiler übersetzt, fügt dieser an jeden Namen einer Routine oder Funktion ein Underscore (_) an. Dies muss bei der Deklaration der FORTRAN Routine unter C++ und bei dessen Aufruf berücksichtigt werden. Bei Verwendung eines Underscore als Teil des Funktionsnamens müssen zwei Underscore angefügt werden. Beim compilieren der AnsiC Routinen mit dem gcc Compiler und der C++ Routinen mit dem g++ Compiler treten keine Besonderheiten auf. Die Hauptroutine des gemischt–sprachigen Beispiels muss sich im C++ Quellcode befinden.

C++ Quellcode:

```
// C++ Hauptroutine 'main'
#include<iostream>
#include<stdio.h>
using namespace std;

extern "C" int fortran_(); // Deklaration der Fortran Routine
extern "C" void cpp();     // Deklaration der C++ Routine
extern "C" void ansic_();  // Deklaration der AnsiC Routine

int main()  // Das Hauptprogramm
{
  cout << "Rufe AnsiC-Routine\n" << endl;
  ansic_();  //Der Underscore am Ende der Routine ist notwendig,
         //da die Routine auch aus FORTRAN mit "ANSIC" aufgerufen wird

  cout << "Rufe Fortran-Routine\n" << endl;
  fortran_();//Der Underscore wird durch den Compiler g77 angefügt,
         //weswegen die Funktion hier derart aufgerufen werden muss.

  cout << "Rufe C++ -Routine\n" << endl;
  cpp();    //Der Aufruf von C++ Code ist wie gewohnt.

  printf("Ende!\n\n");
 return 0;
}


//von C++ und über AnsiC von Fortran aus aufgerufene C++ Routine
void cpp()
 {
  cout << " C++ Subroutine aktiv!" << endl;
  return;
 }
```

AnsiC Quellcode:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int cpp1_() //Wrapper Funktion, die aus Fortran aufgerufen wird
         //und lediglich die Parameter an die C++ Routine übergibt
 {
  printf("Wrapper ruft nur C++ auf!\n");
  cpp();   // Aufruf der C++ Routine
  return 0;
 }



int ansic_() // Definition der AnsiC Routine
 {
  printf("Hier ist die Routine von Ansi C\n");
  return 0;
 }
```

Fortran Quellcode:

```
       SUBROUTINE FORTRAN
       INTEGER*4 I
       WRITE(*,'(A)') 'Aufruf der Fortran Subroutine '
        I=1
        CALL SUBF (I)
       WRITE (*,'(A)') 'Aufruf der AnsiC Routine'
        CALL ANSIC
       END


       SUBROUTINE SUBF (I)
        INTEGER*4 I
       WRITE (*,'(A)') 'Aufruf der Wrapper Funktion (AnsiC)'
       WRITE (*,'(A)') 'Sie ruft wiederum C++ auf'
        CALL CPP1()
        RETURN
       END
```