Umsetzung der Modalanalyse für das Entwicklungspaket Diffpack

Diplomarbeit

Sebastian Peetz
Fachbereich Mathematik

2. Dezember 2008

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Bayreuth, den 2. Dezember 2008 Sebastian Peetz

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Neugier steht immer an erster Stelle eines Problems, das gelöst werden will.

Galileo Galilei

Jede Struktur, die uns im Alltag begegnet, ist in der Lage zu schwingen. Sei es ein Fahnenmast, eine Autokarosserie oder gar ein ganzes Haus. Auch wenn diese Schwingungen oft nicht sichtbar sind, so kann man sie doch oft spüren, manchmal sogar hören. Dabei spielt die Frequenz der Schwingung, also die Anzahl der Schwingungen in einem bestimmten Zeitraum, die ausschlaggebende Rolle.
Auch wenn solche Schwingungen immer von einer von außen einwirkenden Kraft angeregt werden, so besitzt doch jede Struktur ihre charakteristischen Eigenschwingungen, die das Verhalten der Bewegungen steuern. Daher schwingt auch ein langer Stahlbalken ganz anders, als ein dünnes Seil, also in ganz anderen Frequenzen. Diese Änderung des Zustands in Abhängigkeit von der Zeit wird als Kinetik bezeichnet und ist ein Teilgebiet der Mechanik. Die wesentlichen Grundlagen und Gesetze der Kinetik wurden von Galileo Galilei (1564-1642) und Isaac Newton (1643-1727) entwickelt. Auf dieser Basis wurde schließlich durch Claude Louis Marie Henri Navier (1785-1836) die Elastizitätstheorie in eine mathematisch überschaubare Form gebracht. Mit Hilfe dieser Theorie ist es möglich zahlreiche Probleme der Baustatik und Dynamik in den Griff zu bekommen.

Eine dynamische Bewegung wird immer mit Hilfe einer Bewegungsgleichung dargestellt, die eine Verformung der Struktur in Abhängigkeit von der Zeit beschreibt. Um diese Gleichung zu bestimmen, muss man sich ein physikalisches Modell erstellen und mit den Grundgesetzen der Mechanik einen Gleichgewichtszustand des Systems beschreiben. Um die daraus entstehende Gleichung jedoch noch mit überschaubarem Aufwand lösen zu können, bedarf es oft einiger Einschränkungen der möglichen Verformungen. Im Laufe dieser Arbeit werden daher

immer nur kleine Schwingungsauslenkungen im Vergleich zur Länge der Struktur betrachtet. Somit kann man die Schwingungen in Längsrichtung ganz vernachlässigen und kommt so zu einer einfacheren Beschreibung, die auch einfacher berechnet werden kann. Das physikalische Modell und die Grundlagen, die man dazu benötigt, werden im ersten Abschnitt des Kapitels 2.1 zu den Grundlagen beschrieben.

Die Simulation der Schwingungen kann jedoch mit einem Computer nie auf der gesamten, kontinuierlichen Struktur berechnet werden, sondern immer nur auf einzelnen Punkten, bzw. Knoten. Man benötigt also eine Diskretisierung des Lösungsgebietes und eine mathematische Theorie, die eine Lösung auf dieser Basis berechnet. Eine solche Möglichkeit bietet sich mit der *Finite Elemente Methode*, die in Kapitel 2.3 erklärt wird. Zudem wird eine Einführung zum benötigten mathematischen Hintergrund gegeben (Kapitel 2.2), der für eine korrekte Beschreibung dieser Methode notwendig ist. Die praktische Durchführung geschieht dann mit dem Softwarepaket Diffpack, welches umfangreiche Methoden zur Berechnung von Differenzialgleichungen zur Verfügung stellt. Die Funktionsweise dieser C++-Bibliothek wird in Kapitel 2.4 beschrieben, sowie die im weiteren verwendeten Elemente mit den zugehörigen Basisfunktionen.

Der wesentliche Augenmerk in dieser Arbeit liegt jedoch, wie der Titel schon sagt, auf einer Implementierung der Modalanalyse. Dabei wird die Schwingung eines Systems durch die Anfangs beschriebenen Eigenfrequenzen und den dazu gehörenden Eigenformen oder Eigenmoden der Struktur bestimmt. Diese müssen zwar zunächst mit Hilfe eines Eigenwertlösers approximiert werden, jedoch bietet sich dann die Möglichkeit, die Verformung zu einem Zeitschritt durch eine einfache Linearkombination aus den Eigenformen mit einem zeitabhängigen Gewichtsfaktor zu berechnen. Lediglich dieser Faktor muss jeweils neu berechnet werden und die Überlagerung der Moden daraus abgeleitet werden. Zudem kann die Anzahl der Terme in der Linearkombination auf die wesentlichen Schwingungen beschränkt werden. Dies bedeutet in der Praxis, dass nur die stark angeregten Schwingungsanteile berücksichtigt werden, was den Rechenaufwand erheblich vereinfacht. Die Grundlagen, sowie eine Beschreibung der Implementierung und das Verfahren zur Berechnung der Eigenformen für diese Methode finden sich in Kapitel 3.

Da nun eine numerische Methode für die Berechnung der Schwingungen zur Verfügung steht, sollen die Möglichkeiten des Verfahrens an einigen Bespielen verdeutlicht werden. In Kapitel 4 wird zunächst die Schwingung einer gespannte Saite untersucht und die Simulation mit der analytischen Lösung vergleichen. Für dieses noch recht einfache System können zahlreiche unterschiedliche Schwingungen in recht kurzer Zeit simuliert werden und ein Vergleich von verschiedenen Diskretisierungen bis zur Überlagerung einer unterschiedlichen Anzahl von Mo-

den ist möglich.

Die Untersuchungen zur Querverformung eines Balkens in Kapitel 5 erweitern die Betrachtungen auf die Verwendung von Biegemomenten, die in einer Struktur mit Biegesteifigkeit zu finden sind. Daher kann der Balken neben verschiedenen Einspannungen am Ende auch frei im Raum schwingen. Solche Simulationen sind gerade im Hinblick auf die Baustatik sehr hilfreich und man kann wiederum den Fehler der Simulation anhand der analytischen Lösung angeben. Jedoch zeigen sich hier auch die Grenzen der numerischen Berechnung, da der Eigenwertlöser unter gewissen Bedingungen die Eigenformen des Systems nicht mehr berechnen kann.
Dieses Problem verstärkt sich, wenn mehrdimensionale Systeme betrachtet werden, also die Querschwingung einer Struktur aus mehreren Balken in einer Ebene. Man muss somit zunächst das System so anpassen, dass die Berechnung der Eigenmoden möglich ist, was auch von der Anzahl der zu berechnenden Werte abhängt. Wurden diese jedoch erfolgreich bestimmt, so bietet die Modalanalyse die Möglichkeit vielfältige Schwingungen zu simulieren.

Zuletzt werden die erzielten Ergebnisse nochmals analysiert und die Vor- und Nachteile des Verfahrens beleuchtet. Die Simulationen und Ergebnisse finden sich auf einer beigelegten CD wieder, die auch Anhand von Videos und weiteren Grafiken die Schwingungen veranschaulicht.

Kapitel 2

Grundlagen

In dieser Arbeit werden die Schwingungen unterschiedlicher physikalischer Systeme betrachtete und mittels numerischer Methoden simuliert. Um die dafür notwendigen Bewegungsgleichungen aufzustellen, benötigt man ein gutes physikalisches Modell, das die Struktur möglichst präzise beschreibt. Mit Hilfe diese Modells stellt man dann die Gleichungen auf, welche die Berechnung der Verformungen in Abhängigkeit von der Zeit ermöglichen. Eine Darstellung der Modelle und die Grundlage zur Herleitung der daraus resultierenden Gleichungen mit den zugehörigen Modellparametern wird in Kapitel 2.1 vorgestellt.
Die Lösung der Bewegungsgleichung ist jedoch oft mit größerem Aufwand verbunden, da es sich üblicherweise um partielle Differenzialgleichungen handelt. Man verwendet daher numerische Methoden, die eine approximative Lösung berechnen, die der analytischen Lösung möglichst nahe kommen soll. Allerdings sind dazu andere funktionalanalytische Grundlagen notwendig, um die Existenz und Eindeutigkeit der so berechneten Lösung nachzuweisen. Eine kurze Einführung der dazu notwendigen Definitionen und Sätze wird in Kapitel 2.2 behandelt.
Im Kapitel 2.3 wird schließlich die *Finite Elemente Methode* als numerisches Lösungsverfahren vorgestellt und auf die Unterschiede der Elemente bei verschiedenen Strukturen eingegangen. Dieses Verfahren ist im Softwarepaket *Diffpack* umgesetzt worden, was im Kapitel 2.4 in Grundzügen dokumentiert ist. Es wird zudem auf die allgemeine Vorgehensweise beim Erstellen einer Simulation mit *Diffpack* eingegangen. Dies geschieht anhand einer Beschreibung der einzelnen Methoden, die vom Benutzer verwendet oder implementiert werden. Die konkrete Umsetzung findet sich dann in den Ausführungen zur Implementierung der Beispiele.
Unter Verwendung dieser Hilfsmittel wird in den folgenden Kapiteln die Umsetzung der Modalanalyse beschrieben und es werden verschiedenen Beispiele aufgezeigt, welche die Vorteile einer Simulation auf dieser Basis verdeutlichen.

2.1 Physikalische Grundlagen

Die Modalanalyse wird im wesentlichen genutzt, um die Deformationen einer Struktur mit Hilfe der systemeigenen Eigenformen zu beschreiben. Diese werden aus dem charakteristischen Polynom der Gleichung bestimmt, die das physikalische System beschreibt. Um eine gute Beschreibung der Verhaltens der Struktur zu erhalten, benötigt man also ein physikalisches Modell, das alle wesentlichen Parameter der Struktur beschreiben kann. Um jedoch die Gleichung zunächst aufstellen zu können, muss man oft einige einschränkende Annahmen treffen. Im Folgenden werden die Modelle für eine schwingende Saite oder allgemein ein Kontinuum, sowie das des eindimensionalen Balkens mit Querverschiebung erläutert. Diese werden dann in den Kapiteln 4 und 5 genutzt, um die jeweiligen Bewegungsgleichungen herzuleiten.

2.1.1 Schwingendes Kontinuum

Defintion: Kontinuum. *Ein System heißt Kontinuum (Mehrz.: Kontinua), wenn die Eigenschaften des schwingungsfähigen Systems stetige Funktionen des Ortes sind [11, Kap. 22].*

Das bedeutet, jede Struktur deren Deformation bei Schwingungen durch eine stetige Funktion dargestellt werden kann, stellt ein solches Kontinuum dar. Für die Ausführungen in dieser Arbeit wird die schwingende Saite unter Vorspannung repräsentativ für diese Kontinua betrachtet. Die Spannkraft soll dabei stets so groß sein, dass die Deformation quer zur Saiten-Achse sehr viel kleiner ist, als die Länge der Saite. Zudem wird ein Koordinatensystem gewählt, bei dem die x-Achse mit der Längsachse der Saite zusammenfällt und die Saite im Ursprung beginnt, also nur positive x-Werte aufweist (vgl. [14], Kap. 7).

Die Deformation wird nun ausschließlich in der xz-Ebene betrachtet, was sich als nützlich erweist, wenn man die Struktur um eine Dimension in y-Richtung erweitern will. Man erhält somit bei einer Länge L des Kontinuums folgendes Gebiet, auf dem man die Differenzialgleichung der Saitenbewegung lösen will:

$$\Omega = \{(x, z) : \ 0 \leq x \leq L, \ z << L\} \tag{2.1.1}$$

Soll das System diskret berechnet werden, was bei der *Finite Elemente Methode* der Fall ist, muss eine Aufteilung in mehrere Saitenelemente vorgenommen werden. Man erhält dann eine Anzahl von N Knoten auf der Saite, auf denen die Lösung der Bewegungsgleichung berechnet werden muss. Aufgrund der kleinen Querverschiebungen im Vergleich zur Saitenlänge können die Längsverschiebungen der Knoten vernachlässigt werden und man muss nur einen Freiheitsgrad an jedem Knoten berücksichtigen.
Entscheidend für das Schwingungsverhalten der Saite ist zum einen die Spannkraft $S(x)$, die auf die Saite einwirkt und zum anderen die Masse pro Länge

S	50	N
ρ	$7.9 \cdot 10^3$	kg/m^2
A	$\pi\ 2.5 \cdot 10^{-5}$	m^2
$c = \sqrt{S/(\rho A)}$	8.977	m/s

Tabelle 2.1: Parameter der Saite

$\mu(x) = \rho(x)A(x)$ mit der Dichte ρ und dem Querschnitt A (vgl. [14], Kap. 7). Da in dieser Arbeit nur isotrope Körper betrachtet werden, fällt bei all diesen Parametern die Ortsabhängigkeit von x weg und in die Bewegungsgleichungen geht nur der Faktor $c^2 = S/\mu$ ein, der das Quadrat der Wellengeschwindigkeit der Schwingung darstellt. Für die Berechnungen werden die in Tabelle 2.1 dargestellten Standardwerte verwendet. Die Details der entstehenden Gleichung werden im Abschnitt 4.1 genauer beschrieben.

2.1.2 Balkenmodell

Der hier betrachtete Balken soll wie auch die Saite nur Deformationen in z-Richtung aufweisen. Ein solcher Balken wird oft auch als *Biegebalken* bezeichnet, der im Vergleich zum *Zugbalken* nicht in die Länge gezogen wird, sondern eine Durchbiegung quer zur Balkenachse erfährt. Allgemein wird der Balken wie folgt definiert:

Definition: Balken. *Wird ein Bauteil durch eine Kraft senkrecht zu seiner Achse oder durch Momente belastet und ist seine Länge um mindestens eine Größenordnung größer als sein Querschnitt, so wird es überwiegend auf Biegung beansprucht. Solche Bauteile werden Biegestäbe oder Balken genannt (siehe [8], Kap. 1).*

Damit der Einfluss der Momente im Balken beobachtet werden kann, benötigt man den Begriff der Biegesteifigkeit. Diese ist definiert als das Produkt aus dem Elastizitätsmodul E und dem Flächenträgheitsmoment I. Diese werden wie folgt definiert:

Definition: Elastizitätsmodul. *Der Elastizitätsmodul ist als Steigung des Graphen im Spannungs-Dehnungs-Diagramm bei einachsiger Belastung innerhalb des linearen Elastizitätsbereichs definiert. Dieser lineare Bereich wird auch als Hookesche Gerade bezeichnet.*

$$E = \frac{\mathrm{d}\sigma}{\mathrm{d}\varepsilon} = \text{const.} \tag{2.1.2}$$

Dabei bezeichnet σ die mechanische Spannung und ε die Dehnung (siehe [1]).

Somit stellt E einen Materialparameter für den Widerstand gegen Verformung dar und der Wert von E ist umso höher, je steifer das Material der Struktur ist.

Definition: Flächenträgheitsmoment. *Das axiale Flächenträgheitsmoment ist ein Maß für die Steifigkeit eines ebenen Querschnitts gegen Biegung auf dieser Achse. In der xz-Ebene wird das Flächenträgheitsmoment in y-Richtung berücksichtigt (siehe [2]).*

$$I_y = \int_A z^2 \, \mathrm{d}A. \tag{2.1.3}$$

Der hier betrachtete Balken soll einen rechteckigen Balkenquerschnitt besitzen und die Achsen des globalen Koordinatensystems jeweils durch die Mitte der Querschnittsabmessungen laufen. Die Lage auf der x-Achse wird wiederum durch den Ursprung am linken Balkenende festgelegt (vgl. Abbildung 2.1).

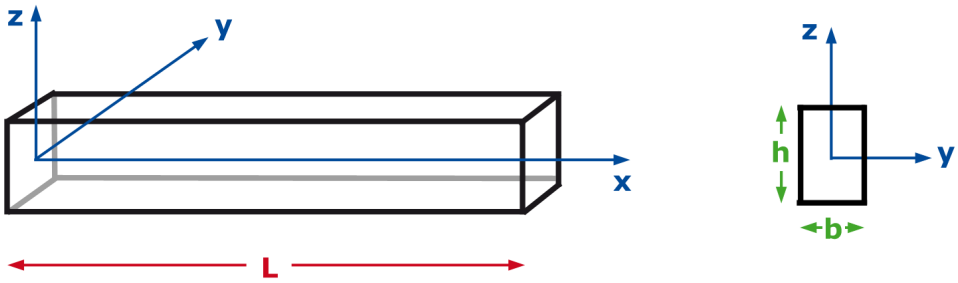

Abbildung 2.1: Lage und Abmessungen des Balkens (Quelle: [10])

Bei diesem Balkenmodell kann man den Flächenträgheitsmoment auch direkt in Abhängigkeit von den Querabmessungen angeben (vgl. [8], Kap. 4).

$$I_y = \int_{-h/2}^{+h/2} z^2 \, b \, \mathrm{d}z = \frac{2}{3} \, b \, z^3 \Big|_0^{h/2} = \frac{b \, h^3}{12}. \tag{2.1.4}$$

Daher benötigt man für die Berechnungen der Schwingungen neben dem Elastizitätsmodul E und der Dichte ρ des Balkens nur die Abmessungen, um die Balkenparameter zu bestimmen. Die Standardwerte für die Simulation sind in Tabelle 2.2 aufgelistet (vgl. E-Modul in [1]).
Im Finite Elemente Modell wird der Balken wieder in mehrere Balkenelemente unterteilt, die aufgrund des einheitlichen Querschnitts und der resultierenden konstanten Biegesteifigkeit auf eindimensionale Elemente reduziert werden können. Um jedoch den Biegemoment $M_b(x,t) = -EI(x)w''(x,t)$ (vgl. [14], Kap. 7)

E	$2.1 \cdot 10^{11}$	N/m^2	b	0.02	m
ρ	$7.9 \cdot 10^{3}$	kg/m^2	h	0.01	m
$I = bh^3/12$	$1.66 \cdot 10^{-9}$	m^4	L	1.0	m

Tabelle 2.2: Balkenparameter

an den Knoten zu berücksichtigen, werden auch die Ableitungen der Verschiebung w mit in die Formulierung des Finiten Balkenelements aufgenommen. Die Ausführungen dazu finden sich im Kapitel 2.3.2 über die Elemente und in der Herleitung der Bewegungsgleichung des Balkens in Kapitel 5.1.

2.1.3 Bewegungsgleichungen

Die Gleichungen, die Schwingungen von Strukturen beschreiben, können oft auf unterschiedliche Weise hergeleitet werden. Ein verbreiteter Ansatz ist das Hamiltonsche Integralprinzip für die Energieen, das den Energieerhaltungssatz der Physik beschreibt (vgl. [13], Kap. 7).

$$\delta \int_{t_1}^{t_2} (\Pi_{(i)} + \Pi_{(a)} - \Pi_{(k)}) dt = \int_{t_1}^{t_2} F_D^T \delta U dt. \tag{2.1.5}$$

Darin wird beschrieben, dass die Summe der in einem System enthaltenen Energie immer gleich der von Außen zugeführten Energie ist. Die nicht-konservativen Dämpfungskräfte F_D sind äußere Kräfte, die dem System Energie entziehen können. Es treten auf der linken Seite der Gleichung drei Formen von Energie auf.

- Formänderungsenergie (innere Energie):

$$\Pi_{(i)} = \frac{1}{2} \int_V \sigma^T \varepsilon \ \mathrm{dV}. \tag{2.1.6}$$

  .

- Äußere Energie:

$$\Pi_{(a)} = -F^T U. \tag{2.1.7}$$

  .

- Kinetische Energie:

$$\Pi_{(k)} = \frac{\rho}{2} \int_V \left(\dot{u}^2 + \dot{v}^2 + \dot{w}^2 \right) \mathrm{dV} \tag{2.1.8}$$

  .

Dabei gehen die Werte für die Spannung σ und die Dehnung ε in die Formänderungsenergie ein, die äußere Energie ist das negative Produkt der konservativen äußeren Kräfte F und der Verschiebung U und die kinetische Energie setzt sich zusammen aus dem Integral der Quadrate der Geschwindigkeiten in den einzelnen Koordinaten über das Volumen. ρ bezeichnet dabei die Dichte der untersuchten Struktur.
Durch die ständige Umwandlung von potentieller (innerer) in kinetische Energie und umgekehrt, wird dann die Schwingung beschrieben, die sich durch eine Differenzialgleichung in Abhängigkeit von der Zeit beschreiben lässt. Die Durchführung einer Herleitung mit Hilfe dieses Ansatzes ist in Kapitel 5.1 für die Balkengleichung zu finden.
Der zweite oft angewendete Ansatz basiert auf den Newtonschen Grundgesetzen und beschreibt die Auswirkungen einer Kraft auf ein infinitesimales Teilstück der Struktur. Durch einen Grenzübergang $\Delta x \to 0$ der Länge des Teilstücks erhält man schließlich eine Gleichung, die das gesamte System beschreibt. Dieser Weg der Herleitung des Bewegungsgleichung wird in Kapitel 4.1 durchgeführt, um die schwingende Saite zu beschreiben.
Die Lösung der Bewegungsgleichungen soll jedoch mit der Finite Elemente Methode durchgeführt werden, was noch einer ganz anderen Betrachtung dieser Gleichungen bedarf. Dazu sind jedoch noch Grundlagen aus dem Bereich der Funktionalanalysis notwendig.

2.2 Funktionalanalytische Grundlagen

Zur numerischen Lösung von vielen physikalischen Problemen, wie auch den hier betrachteten Bewegungsgleichungen, sind einige Grundlagen aus der Fuktionalanalysis notwendig. Besonders die in der Finite Elemente Methode zu Anwendung kommende *schwache Formulierung* von Differenzialgleichungen und die Existenz und Eindeutigkeit einer Lösung wird in diesem Kapitel erläutert. Dazu müssen jedoch zunächst einige Begriffe eingeführt werden, wobei die Beweisführung der folgenden Sätze den Rahmen dieser Arbeit sprengen würde und daher auf Spezialliteratur verwiesen wird.

2.2.1 Sobolew-Räume

Alle in dieser Arbeit behandelten Bewegungsgleichungen gehören zur Klasse der Differenzialgleichungen. Diese werden wie folgt charakterisiert.

Definition: Differenzialgleichung. *Jede Gleichung, die mindestens einen Differenzialquotienten enthält heißt* Differenzialgleichung. *Sie heißt von n-ter Ordnung, falls die höchste vorkommende Ableitung von Ordnung n ist.*

Man unterscheidet zwischen *gewöhnlichen* Differenzialgleichungen (ODE, *Ordinary Differential Equation*), die nur Ableitungen nach einer Variablen enthalten und *partiellen* Differenzialgleichungen (PDE, *Partial Differential Equation*), bei denen mindestens zwei Ableitungen verschiedener Variablen vorkommen. Eine Besonderheit solcher Gleichungen besteht darin, dass ihre Lösung nur unter Einbeziehung von Anfangs- oder Randbedingungen eindeutig ist. Zudem sind in den meisten Fällen keine analytischen Lösung bekannt und man muss sich numerischer Methoden bedienen.
Die am häufigsten verwendeten Randwerte auf dem Rand des Lösungsgebietes Ω sind *Dirichlet-* oder *Neumann-Randbedingungen*. Im ersten Fall wird ein fester Wert der Lösung u auf dem Rand $\partial\Omega$ vorgegeben und man bezeichnet sie als wesentliche Randbedingungen. Der zweite Fall bezeichnet dagegen natürliche Randwerte, die durch die Normalenableitung auf dem Rand definiert werden.
Die Lösung eines solchen Randwertproblems ist nun durch die Differenzialgleichung selbst an hohe Glattheitsanforderungen gebunden. Um allgemeinere Lösungsansätze zu ermöglichen versucht man diese abzuschwächen und führt daher die *schwache Formulierung* oder *Variationsformulierung* der Gleichung ein. Die Herleitung dieser Gleichungen wird in Kapitel 4.3 und 5.3 beschrieben und ausgeführt. Die schwache Formulierung besteht aus einer Bilinearform und einem linearen Funktional der Testfunktionen, welche nur auf geeigneten Vektorräumen Sinn machen. Für die Lösung dieser Gleichungen führt man daher den so genannten *Sobolew-Raum* ein. Dafür benötigt man jedoch noch die Begriffe der L_p-Räume und der *schwachen Ableitung*.

Definition: L_p-Räume. *Sei $\Omega \in \mathbb{R}^n$ Lebesgue-messbar, $1 \leq p \leq \infty$, Y Banach-Raum mit Norm $y \to |y|$. Dann ist*

$$\begin{aligned} L_p(\Omega) &:= \{u : \Omega \to Y \ : \ u \text{ messbar, } |u|^p \text{ Lebesgue-integrierbar}\} \text{ für } p < \infty, \\ L_\infty &:= \{u : \Omega \to Y \ : u \text{ messbar und } \operatorname{ess\,sup}_{x\in\Omega} ||u(x)|| < \infty\} \end{aligned} \tag{2.2.1}$$

ein Banach-Raum mit der Norm

$$||u||_{L_p(\Omega)} := \left(\int_\Omega |u|^p dL_n) \right)^{1/p} \tag{2.2.2}$$

und für $p = 2$ ein Hilbert-Raum, falls Y Hilbert-Raum ist (vgl. [3], Kap. 1). Das Skalarprodukt auf $L_2(\Omega)$ ist dann definiert durch

$$\langle u_1, u_2 \rangle_{L_2(\Omega)} := \int_\Omega \langle u_1, u_2 \rangle_Y \, dL_n(x). \tag{2.2.3}$$

Definition: Schwache Ableitung. *Sei $\alpha \in \mathbb{N}_0^d$ ein Multi-Index und $1 \leq p \leq 1$. $u \in L_p(\Omega)$ besitzt eine schwache Ableitung $D^\alpha u \in L_p(\Omega)$ genau dann, wenn*

$$\int_\Omega D^\alpha u(x)\phi(x)dx = (-1)^{|\alpha|} \int_\Omega u(x)\partial^\alpha \phi(x)dx \quad \forall \phi \in \mathbb{C}_0^\infty(\Omega) \tag{2.2.4}$$

mit

$$\alpha = (\alpha_1, \ldots, \alpha_d),\ \partial^\alpha \phi(x) = \frac{\partial^{\alpha_1} \ldots \partial^{\alpha_d}}{\partial^{\alpha_1} x_1^{\alpha_1} \ldots \partial^{\alpha_d} x_d^{\alpha_d}} \phi(x) \text{ und } |\alpha| = \sum_{i=1}^{d} \alpha_i \tag{2.2.5}$$

erfüllt ist. Dabei bezeichnet $\mathbb{C}_0^\infty(\Omega)$ die Menge der beliebig oft stetig differenzierbaren Funktionen mit kompaktem Träger (vgl. [5], Kap. 1).

Definition: SobolewRaum. *Es seien $\Omega \subset \mathbb{R}^n$ offen und $1 \leq p \leq \infty$. Der Raum derjenigen reellwertigen Funktionen $u \in L_p(\Omega)$, deren gemischt partielle schwache Ableitungen bis zur Ordnung k in $L_p(\Omega)$ liegen, ist der Sobolew-Raum $H^{k,p}(\Omega)$ (vgl. [3], Kap. 1).*

$$H^{k,p}(\Omega) = \{u \in L_p(\Omega) : D^\alpha u \in L_p(\Omega),\ \alpha = 1, \ldots, k\} \tag{2.2.6}$$

Mit der so genannten *Sobolew-Norm*

$$\|u\|_{H^{k,p}(\Omega)} = \left(\sum_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_p(\Omega)}^p \right)^{1/p} \tag{2.2.7}$$

für $p < \infty$ bzw.

$$\|u\|_{H^{k,\infty}(\Omega)} = \max_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_\infty(\Omega)} \tag{2.2.8}$$

ist $H^{k,p}(\Omega)$ ein Banach-Raum, für $1 < p < \infty$ ist er sogar reflexiv.
Für $p = 2$ wird die Norm durch das Skalarprodukt

$$\langle u, v \rangle_{H^{k,2}(\Omega)} := \sum_{|\alpha| \leq k} (\partial^\alpha u, \partial^\alpha v)_{L^2(\Omega)} \tag{2.2.9}$$

induziert. $H^{k,2}(\Omega)$ ist daher ein Hilbert-Raum, und man schreibt auch $H^k(\Omega) := H^{k,2}(\Omega)$.

2.2.2 Existenz und Eindeutigkeit

Nachdem der Sobolew-Raum nun eingeführt wurde, sind noch weitere funktionalanalytische Grundlagen notwendig, um die Existenz und Eindeutigkeit der schwachen Lösung zu folgern.

Definition: Dualraum. *Sei V ein Hilbert-Raum. Der Dualraum von V wird mit V^* bezeichnet und ist definiert als*

$$V^* := \{v^* : V \to \mathbb{R} \ : \ v^* \text{ linear und stetig}\}\,. \tag{2.2.10}$$

Die Elemente des Dualraums sind also lineare und stetige Funktionale. Außerdem ist der Duale Raum wegen der Vollständigkeit von $\mathbb{R}$ stets ein Banach-Raum. Die Elemente aus V^* können mittels des folgenden Satzes dargestellt werden:

Satz: Rieszscher Darstellungssatz. *Sei V ein reeller Hilbert-Raum. Dann ist jedes lineare und stetige Funktional $J \in V^*$ darstellbar durch:*

$$J(u)(v) = \langle u, v\rangle \quad \forall u, v \in V. \tag{2.2.11}$$

[Beweis [3], Kap. 4.6]

Zwar gilt diese Aussage allgemein für Bilinearformen $\langle ., .\rangle$, jedoch soll für die folgenden Betrachtungen das oben definierte Skalarprodukt $\langle ., .\rangle_{L_2}$ verwendet werden. Diesen Satz benötigt man, um das folgende Lemma von *Lax-Milgram* zu beweisen, durch das dann schließlich die Existenz und Eindeutigkeit der schwachen Lösung gefolgert werden kann.

Satz: Lemma von Lax-Milgram. *Es sei V ein Hilbert-Raum und $B : V \times V \to \mathbb{R}$ eine Bilinearform, so dass gilt*

$$|B(u, v)| \ \leq \ M\|u\|\|v\|, \quad \forall\, u, v \in V \tag{2.2.12}$$
$$B(u, u) \ \geq \ m\|u\|^2, \quad \forall u \in V \tag{2.2.13}$$

mit $0 < m \leq M < \infty$.
Dann existiert genau eine stetige, bijektive Abbildung $A : V \to V$ so, dass

$$B(u, v) = \langle u, Av\rangle_V \quad \forall u, v \in V. \tag{2.2.14}$$

[Beweis [3], Kap. 4.7]

Mit Hilfe dieses Satzes kann man nun eine Formulierung der Gleichung angeben, deren eindeutige Lösung bekannt ist. Da diese der schwachen Formulierung der Differenzialgleichung entspricht, erhält man einen Existenz- und Eindeutigkeitsbeweis für die Lösung einer solchen Gleichung.

Lemma. *Sei A die lineare Abbildung aus dem Lemma von Lax-Milgram 2.2.14 und J die Isometrie aus dem Rieszschen Darstellungssatz 2.2.11, so erhält man für $v^* \in V^*$ mit $u := A^{-1}J^{-1}v^*$ die eindeutige Lösung der Gleichung*

$$B(u, v) = v^*(v) \quad \forall u, v \in V. \tag{2.2.15}$$

Kann eine Differenzialgleichung also in diese Gestalt der schwachen Formulierung gebracht werden, so garantiert das Lemma sowohl die Existenz der Lösung, als auch deren Eindeutigkeit. Dies ist eine wesentliche Grundlage der Methode der Finiten Elemente, die auf dieser Formulierung aufbaut.

2.3 Finite Elemente Methode

2.3.1 Theorie

Die Finite Elemente Methode (*Finite Element Method* (FEM)) ist ein weit verbreitetes numerisches Verfahren, um partielle Differenzialgleichungen zu lösen. Dabei können auch komplizierte geometrisch Strukturen durch die Diskretisierung einfach gehandhabt werden. Das folgende Kapitel gibt einen kurzen Einblick in die grundlegenden Ideen der Methode und erklärt die Funktionsweise der Approximation.

Im Gegensatz zu einer *Finite Differenzen Methode* werden hier die Ableitungen nicht durch Differenzenquotienten ersetzt, sondern man sucht eine Approximation der unbekannten Funktion $u(x)$ der Form (vgl. [12], Kap. 2.1)

$$\hat{u}(x) = \sum_{j=1}^{M} u_j \, N_j(x). \tag{2.3.1}$$

Die Summe $\hat{u}$ setzt sich dabei aus vorgegebenen *Basisfunktionen* N_j zusammen und den zugehörigen unbekannten Koeffizienten u_j. Das Ziel ist es nun, diese Faktoren so zu berechnen, dass der Fehler $|u - \hat{u}|$ minimiert wird. Obwohl die Lösung u nicht bekannt ist und somit auch der Fehler der Approximation, kann man den Fehler der PDE bei Einsetzen von $\hat{u}$ abschätzen.
Man benötigt dazu einen Differenzialoperator $\mathcal{L}$ mit $\mathcal{L}(u(x)) = 0, x \in \Omega$. Setzt man nun $\hat{u}$ in die PDE ein, so erhält man ein $\mathcal{L}(\hat{u}(x)) \neq 0$ und man bezeichnet diesen Fehler $\mathcal{R} = \mathcal{L}(\hat{u})$ als Residuum. Mit Hilfe dieses Operators kann man nun eine Methode einführen, die die Koeffizienten u_j aus (2.3.1) bestimmt, indem man fordert, dass das gewichtete Residuum über dem Gebiet Ω verschwindet. Durch die Gewichtung mit den Gewichtsfunktionen W_i spricht man auch von der Methode der gewichteten Residuen.

$$\int_{\Omega} RW_i \, d\Omega = 0, \qquad i = 1, \ldots, M. \tag{2.3.2}$$

Die Gewichtsfunktionen W_i können unterschiedlich definiert werden, zum Beispiel erhält man die *Least-Squares-Method* durch die Funktionen $W_i = 2\partial R/\partial u_i$, die oft auch als *Test*-Funktionen bezeichnet werden. Eine andere Möglichkeit nach der Methode von Galerkin besteht darin, die Gewichtsfunktionen gleich den Basisfunktionen N_i zu setzen.
Fordert man nun, dass die Basisfunktionen N_i auf dem Rand $\partial\Omega$ des untersuchten Gebietes Ω verschwinden, also $N_i = 0$ auf $\partial\Omega$, so kann man Randbedingungen in die Gleichung mit aufnehmen. Dazu sei Ψ eine Funktion, die die Lösung u auf dem Rand $\partial\Omega$ erfüllt, dann folgt:

$$\hat{u}(x) = \Psi(x) + \sum_{j=1}^{M} u_j \, N_j(x). \tag{2.3.3}$$

Beinhalten die Randbedingungen Ableitungen, so kann man diese durch partielle Integration in die Gleichung mit aufnehmen. Betrachtet man zum Beispiel die Poissongleichung

$$-\Delta u(x) = f(x) \qquad\qquad u(0) = 1, u'(1) = \beta, \tag{2.3.4}$$

so erhält man bei Multiplikation mit Basisfunktionen $N_i \in \mathbb{C}_0^\infty(\Omega)$ und Integration über Ω:

$$-\int_\Omega \Delta u N_i \, dx = \int_\Omega f N_i \, dx, \qquad i = 1 \ldots M. \tag{2.3.5}$$

Durch partielle Integration auf der linken Seite ergibt sich:

$$\int_\Omega \nabla u \nabla N_i \, dx - N_i(1)\hat{u}'(1) + N_i(0)\hat{u}'(0) = \int_\Omega f N_i \, dx. \tag{2.3.6}$$

Nun kann man die Gleichung umstellen und die Randbedingung einsetzen:

$$\int_\Omega \nabla u \nabla N_i \, dx = \int_\Omega f N_i \, dx + N_i(1)\beta. \tag{2.3.7}$$

Mit Hilfe der oben beschriebenen Galerkin Methode setzt man nun $\hat{u}$ aus 2.3.3 ein:

$$\sum_{j=1}^{M} \left(\int_\Omega \nabla N_i \nabla N_j \, d\Omega \right) u_j = \int_\Omega f(x) N_i \, d\Omega - \int_{\partial\Omega} g(x) N_i \, d\Gamma - \int_\Omega \nabla N_i \nabla \Psi \, d\Omega. \tag{2.3.8}$$

$g(x)$ bezeichnet hierbei eine Funktion, die Informationen über die Normalenableitungen auf dem Rand enthält.

2.3.2 Formulierung der Elemente

Die Grundlegende Idee bei Konstruktion des Gleichungssystems bei der Finite Elemente Methode ist eine Formulierung über die einzelnen Elemente selbst (vgl. [12], Kap. 2.3). Die Matrix A lässt sich schreiben als

$$A_{i,j} = \int_{\Omega} N_i N_j \, d\Omega = \sum_{e=1}^{m} A_{i,j}^{(e)}, \qquad \text{wobei } A_{i,j}^{(e)} = \int_{\Omega_e} N_i N_j \, d\Omega_e. \tag{2.3.9}$$

Das bedeutet, man stellt zunächst nur Matrizen $A^{(e)}$ für die einzelnen Elemente auf und erstellt die resultierende Massen- und Steifigkeitsmatrix durch Aufsummieren der Elementmatrizen. Um die Basisfunktionen für dieses Elemente zu bestimmen, werden hier zunächst nur eindimensionale Elemente betrachtet. Das Gebiet $\Omega_e = [x^{[e]}, x^{[e+1]}]$ wird dafür auf das Intervall $[-1, 1]$ transformiert und die Koordinate $\xi \in [-1, 1]$ auf diesem Gebiet eingeführt. Bei mehrdimensionalen Elementen wird diese Transformation entsprechend auf ein Referenzelement der gleichen Dimension durchgeführt, was in Abbildung 2.2 für ein Viereckselement in 2D dargestellt ist.

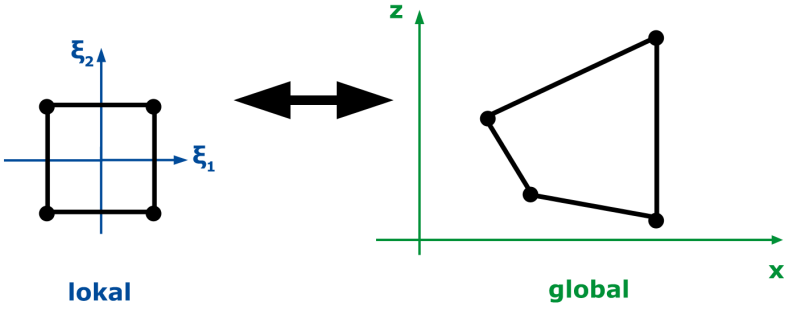

Abbildung 2.2: Transformation eines Elementes (Quelle: [10])

Bei eindimensionalen linearen Elementen ist $x = x^{(e)}(\xi)$ gegeben durch

$$x^{(e)}(\xi) = \frac{1}{2}\left(x^{[e]} + x^{[e+1]}\right) + \xi\frac{1}{2}\left(x^{[e+1]} - x^{[e]}\right). \tag{2.3.10}$$

Diese Transformation kann meist direkt mit den Basisfunktionen N durchgeführt werden. Man spricht dann von einem so genannten *isoparametrischen* Ansatz, im Gegensatz zu *nicht-isoparametrischen* Ansätzen, bei denen weitere Geometriefunktionen notwendig sind.

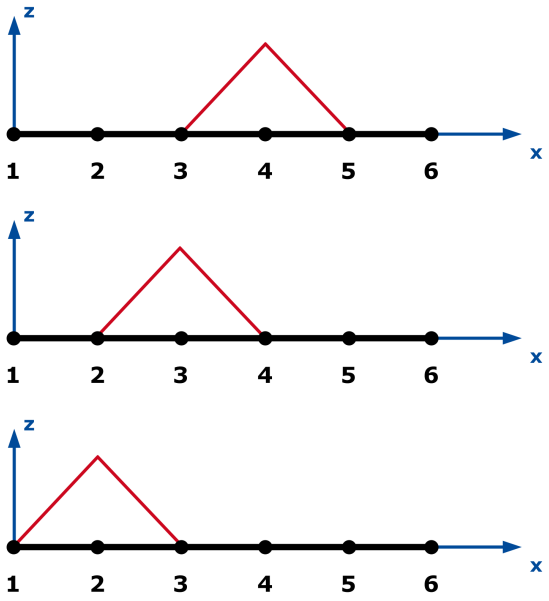

Abbildung 2.3: Dreiecks-Basisfunktionen (Quelle: [10])

Mit Hilfe der *lokalen Koordinaten* ξ kann man nun die Basisfunktionen $\tilde{N}_r(\xi) = N_i(x^{(e)}(\xi))$ ausdrücken. Dabei muss man die globale Knotennummer i über eine Funktion $i = q(e, r)$ noch in Bezug zur lokalen Knotennummer r des Elementes e setzen. Im linearen eindimensionalen Fall lassen sich die Basisfunktionen nun einfach konstruieren, da $\tilde{N}_1(-1) = \tilde{N}_2(1) = 1$ und $\tilde{N}_1(1) = \tilde{N}_2(-1) = 0$ gegeben ist. Die linearen Funktionen mit diesen Randwerten ergeben sich zu:

$$\tilde{N}_1(\xi) = \frac{1}{2}(1 - \xi), \quad \tilde{N}_2(\xi) = \frac{1}{2}(1 + \xi). \tag{2.3.11}$$

Diese Basisfunktionen sind die so genannten *Dreiecks-* oder *Hütchen-Funktionen* (siehe Abbildung 2.3) und können beispielsweise für die schwingende Saite genutzt werden.
Ein wichtiger Aspekt ist auch die Auswirkung der Transformation der Koordinaten von x nach ξ auf das Integral und die Ableitungen der Basisfunktionen. Man benötigt dazu die Jacobi-Matrix $J = dx/d\xi$ für die Abbildung $x = x^{(e)}(\xi)$. Daraus kann man die Ableitungen umformen zu

$$\frac{dN_i}{dx} = \frac{d\tilde{N}_r}{d\xi}\frac{d\xi}{dx} = J^{-1}\frac{d\tilde{N}_r}{d\xi}, \; i = q(e, r), \tag{2.3.12}$$

und für das Integral des Produkts der Ableitungen ergibt sich entsprechend

$$\int_{x^{[e]}}^{x^{[e+1]}} N_i'(x)N_j'(x)\, dx = \int_{-1}^{1} J^{-1}\frac{d\tilde{N}_r}{d\xi} J^{-1}\frac{d\tilde{N}_s}{d\xi} \det J \, d\xi, \tag{2.3.13}$$

wobei $i = q(e, r)$ und $j = q(e, s)$ gilt.
Will man allerdings die Gleichung für den schwingenden Balken lösen, benötigt man dazu auch die zweite Ableitung der Basisfunktionen, die im linearen Fall natürlich verschwindet. Man bedient sich hier der so genannten *Hermiten Polynome*, die zusätzlich noch stetige Ableitungen an den Knoten besitzen. Um diese jedoch eindeutig definieren zu können, muss man einen weiteren Freiheitsgrad für die Ableitung der Deformation an den Knoten einführen. Das Balkenelement besitzt somit für jeden Knoten einen Freiheitsgrad w für die Deformation und einen weiteren θ für der Ableitung.
Man erhält somit vier Basisfunktionen mit folgenden Eigenschaften, wobei l die Länge des Elementes bezeichnet:

$$\tilde{N}_1(-1) = 1, \qquad \tilde{N}_1(1) = 0, \qquad \tilde{N}_1'(-1) = \tilde{N}_1'(1) = 0, \tag{2.3.14}$$
$$\tilde{N}_3(-1) = 0, \qquad \tilde{N}_3(1) = 1, \qquad \tilde{N}_3'(-1) = \tilde{N}_3'(1) = 0, \tag{2.3.15}$$
$$\tilde{N}_2(-1) = \tilde{N}_2(1) = 0, \qquad \tilde{N}_2'(-1) = l, \qquad \tilde{N}_2'(1) = 0, \tag{2.3.16}$$
$$\tilde{N}_4(-1) = \tilde{N}_4(1) = 0, \qquad \tilde{N}_4'(-1) = 0, \qquad \tilde{N}_4'(1) = l. \tag{2.3.17}$$

Daraus lassen sich nur Polynome 3ten Grades eindeutig ableiten. Man nennt diese *Hermite Polynome* (nach Charles Hermite) und diese sind gegeben durch

$$\tilde{N}_1 = 0.25\xi^3 - 0.75\xi + 0.5, \tag{2.3.18}$$
$$\tilde{N}_2 = (0.25\xi^3 - 0.25\xi^2 - 0.25\xi + 0.25)l, \tag{2.3.19}$$
$$\tilde{N}_3 = -0.25\xi^3 + 0.75\xi + 0.5, \tag{2.3.20}$$
$$\tilde{N}_4 = (0.25\xi^3 + 0.25\xi^2 - 0.25\xi - 0.25)l. \tag{2.3.21}$$

(vgl. [13], Kap. 5.2). Diese Funktionen sind in Abbildung 2.4 dargestellt.

Die Verwendung dieser Basisfunktionen ist auch in der Implementierung eine Besonderheit, da für ein Element mit zwei Knoten und jeweils einem Freiheitsgrad pro Knoten dennoch 4 Funktionsauswertungen notwendig sind. Jedoch lässt sich somit auch die Biegesteifigkeit eines Balkens in ein eindimensionales Modell mit einbeziehen.

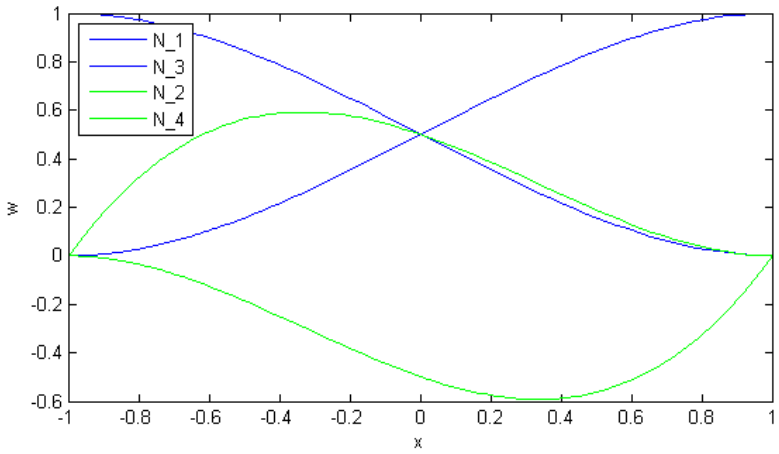

Abbildung 2.4: Hermite Basisfunktionen

2.4 Umsetzung in Diffpack

Zur effizienten Lösung von Differenzialgleichungen benötigt man ein umfangreiches Software-Paket, um die Methode der Finiten Elemente anwenden zu können. Die Klassenbibliothek *Diffpack* bietet hierfür eine objektorientierte Lösung zur Entwicklung numerischer Software in C++, die insbesondere auf die Lösung von partiellen Differenzialgleichungen ausgelegt ist.

2.4.1 Die Geschichte von Diffpack

Unter der Leitung von Hans Peter Langtangen und Are Magnus Bruaset wurde seit 1991 an der Universität Oslo die Programmierumgebung *Diffpack* entwickelt. Im Vergleich zu bestehenden Fortran oder C Bibliotheken wurden hier insbesondere die Vorteile der objektorientierten Programmierung genutzt und eine Vielzahl von mathematischen Klassen zur Modellierung komplexer Probleme geschaffen. Im Jahre 2003 übernahm die Firma InuTech GmbH in Nürnberg die Weiterentwicklung sowie Wartung und bietet zudem Schulungen zur Nutzung der Software. Inzwischen wird *Diffpack* zur Lösung zahlreicher komplexer Probleme wie die Simulation der Wärmeverteilung in Kernreaktoren und anderer partieller Differenzialgleichungen verwendet. Zudem werden ständig neue Erweiterungen wie die hier behandelte Modalanalyse zur Klassenbibliothek hinzugefügt. Der Aufbau und die Abläufe bei einem solchen Simulator in *Diffpack* werden im nächsten Abschnitt näher erläutert.

2.4.2 Programmieren mit Diffpack

Die Lösung einer Differenzialgleichung folgt im *Diffpack*-Kern einem festen vorgegebenen Ablauf an Lösungsschritten. Dazu muss in der `main` Funktion zunächst eine Initialisierung durch `initDiffpack(int argc, char argv)` mit den Eingabewerten stattfinden. Die für den Simulator frei wählbaren Parameter werden später über ein globales Menü übergeben, welches hier ebenfalls mit `global_menu.init("new simulator", "Solver")` initialisiert wird. Man legt nun ein Objekt des neuen Simulators an, hier `Solver` genannt, und startet das Lösungsverfahren mit dem Aufruf `global_menu.multipleLoop( simulator )`. Die gesamte main-methode sieht dann wie folgt aus:

```
int main( int argc , const char* argv[] )
{
  initDiffpack( argc, argv ) ;
  global_menu.init( "new simulator", "Solver" ) ;
  Solver simulator ;
  global_menu.multipleLoop( simulator ) ;
  return 0 ;
}
```

Die eigentliche Applikation befindet sich nun in der Klasse `Solver`, in der zunächst durch `multipleLoop` die virtuellen Funktionen `adm(MenuSystem& menu)`, `solveProblem()` und `resultReport()` aufgerufen werden. Dabei wird auch die Struktur einer *Diffpack* Applikation deutlich, die sich aus der Zuweisung der Systemparameter, der Methode zum Erstellen und Lösen des Gleichungssystems und dem Dokumentieren der Ergebnisse zusammen setzt. Dieser Ablauf ist auch in Abbildung 2.5 dargestellt.

`adm( MenuSystem& menu )`

Die Funktionen unter `adm` sind lediglich für die Administration und Zuweisung der Systemvariablen zuständig und werden in [12] genauer beschrieben. Zudem finden sich in den Beispielen 4.4 und 5.4 noch Ausführungen zu den genauen Implementierungen.

`solveProblem()`

Die Lösung der Differenzialgleichung wird über die Methode `solveProblem` eingeleitet und gesteuert. Hier werden alle benötigten Funktionen aufgerufen vom Setzten der Randbedingungen über das Erzeugen des Gleichungssystems bis hin zur Lösung. Dazu werden üblicherweise zunächst die unten beschriebenen Funktionen `fillEssBC` und `setIC` aufgerufen, um danach mit `makeSystem` die in der

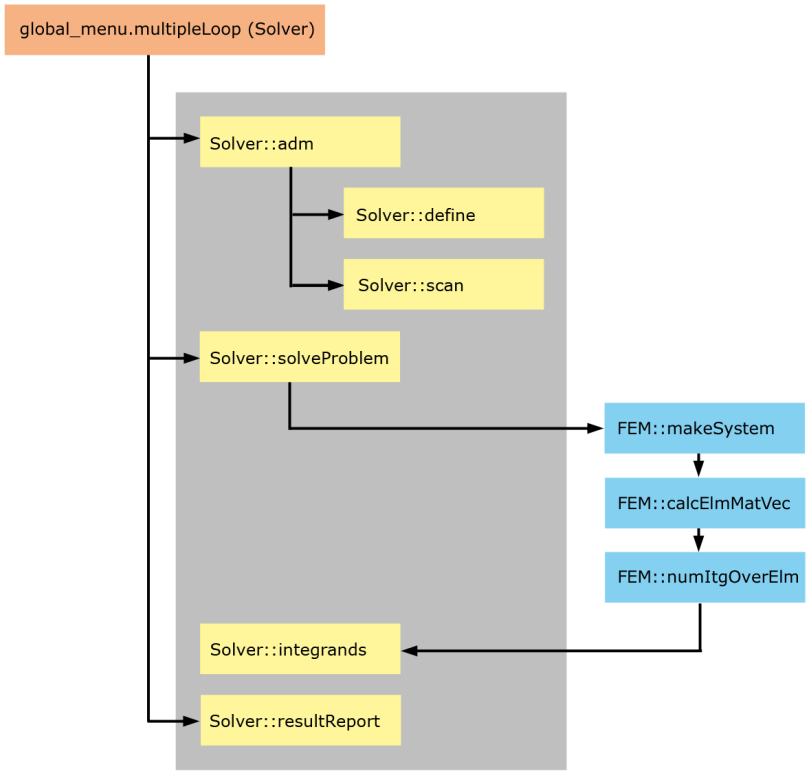

Abbildung 2.5: Programmablauf in Diffpack (Quelle: [10])

Grafik 2.5 blau dargestellten Programmschritte auszuführen. Diese sind Teil der Finiten Elemente Klasse `FEM` und müssen nicht vom Benutzer implementiert werden. Das mit Hilfe der schwachen Formulierung der Gleichung in `integrands` erzeugte System kann dann mit verschiedenen Lösern berechnet werden. Der Aufruf von `LinEqAdmFE::solve()` würde beispielsweise das durch Finite Elemente erzeugte lineare Gleichungssystem mittels Gaußelimination als Standardverfahren lösen.

`setIC()`

Zum Setzen möglicher Anfangsauslenkungen A nach Gleichung 4.2.19 der Struktur werden hier die Werte für das `FieldsFE u_init` gesetzt. Im Fall von nur einem Freiheitsgrad pro Knoten genügt hier auch ein einfacheres `FieldFE`. Die Werte können entweder aus den oben genannten Gleichung bestimmt oder beliebig für die Knotenauslenkungen gewählt werden.

`fillEssBC()`

In dieser Funktion werden die essentiellen Randbedingungen festgelegt. Üblicherweise führt man dazu Randindikatoren im *Gridfile* ein, in dem auch die Knoten und Elemente festgelegt werden. Diese Randwerte werden mit Hilfe von `initEssBC()` einem Objekt der Klasse `DegFreeFE` für die Freiheitsgrade des Systems übergeben.

`makeSystem( DegFreeFE dof , LinEqAdmFE lineq )`

Um die Matrizen des linearen Systems zu erzeugen werden dieser FEM-Methode die Objekte für die Freiheitsgrade `DegFreeFE` und für das Gleichungssystem selbst `LinEqAdmFE` übergeben. Im *Diffpack*-Kern werden nun die Funktionen `FEM::calcElmMatVec` und `FEM::numItgOverElm` ausgeführt, welche für die Berechnung der Elementmatrix und des Elementvektors nach Gleichung 2.3.9 und die numerische Integration zuständig sind. Dabei wird auch die schwache Formulierung der Problemstellung durch die Funktion `intergrands` einbezogen.

`intergrands( ElmMatVec& elmat , const FiniteElement& fe )`

Die Definition der Elementmatrix und des Elementvektors geschieht schließlich in der `integands` Funktion, die die schwache Formulierung der Differenzialgleichung beinhaltet. Auf die benötigten Basisfunktionen und die Jacobideterminante (vgl. Gleichung 2.3.13) kann man dabei über ein Objekt der Klasse `FiniteElement` zugreifen. Die Ableitungen der Basisfunktionen können nach Gleichung 2.3.12 ebenfalls über diese Objekt abgefragt werden (siehe Kapitel 4.4). Im `ElmMatVec` Objekt wird dann die Matrix und der Vektor des Elements gespeichert.

Hat man nun mit Hilfe dieser Methoden ein lineares Gleichungssystem für die Lösung der Differenzialgleichung erstellt, kann man dies bequem über `LinEqAdmFE::solve()` lösen. Dabei können über das Menü verschieden Berechnungsmethoden ausgewählt werden, in den meisten Fällen genügt hier die Gaußelimination, die als Standardlöser eingestellt ist. Für die hier behandelte Modalanalyse wird das Gleichungssystem jedoch nicht auf diese Weise gelöst, sondern zunächst als Grundlage für eine Eigenwertberechnung verwendet. Die Lösung wird dann aus einer Linearkombination der approximierten Eigenformen und den zugehörigen zeitabhängigen Gewichtsfaktoren berechnet.

Kapitel 3

Berechnung von Schwingungen

3.1 Das Verfahren der Modalanalyse

Eines der wichtigsten Verfahren zur Lösung von Bewegungsgleichungen ist die Modalanalyse. Dabei versucht man die Schwingung eines Systems durch Überlagerung der darin auftretenden Eigenschwingungen zu simulieren. Dazu betrachtet man zunächst die allgemeine Bewegungsgleichung 3.1.1, die mit Hilfe der Ansätze aus Kapitel 2.1.3 hergeleitet werden kann (siehe [13], Kap. 7) und eine lineare Differenzialgleichung 2. Ordnung darstellt.

$$M\ddot{U}(t) + C\dot{U}(t) + KU(t) = F(t). \tag{3.1.1}$$

Die linearen Abbildungen M, C und K werden in diesem Fall durch die Methode der Finiten Elemente bestimmt. Der Kraftvektor F auf der rechten Seite kann auch wegfallen, was zu einem homogenen System führt. Zunächst soll hier der Fall der ungedämpften Schwingung betrachtet werden, bei dem der Dämpfungsterm in der Mitte entfällt und die Gleichung nur noch von U und $\ddot{U}$ abhängig ist.

$$M\ddot{U}(t) + KU(t) = 0. \tag{3.1.2}$$

Um diese Gleichung zu Lösen bedient man sich des allgemeinen harmonischen Ansatzes $U(t) = X \cos \omega t$ und erhält durch Einsetzen das ungedämpfte Eigenschwingungsproblem

$$(-\omega^2 M + K)X = 0 \tag{3.1.3}$$

(vgl. [13], Kap. 9). Man setzt nun $\lambda = \omega^2$ und stellt die Gleichung so um, dass man ein verallgemeinertes Eigenwertpoblem erhält. Dabei gibt es es genau so viele Eigenwerte λ_r wie Dimensionen N in den Matrizen ($r = 1, 2, \ldots, N$), was von der Feinheit der Diskretisierung abhängt. Die zugehörigen Eigenvektoren werden mit X_r bezeichnet und werden auch Eigenformen genannt.

$$KX_r = \lambda_r M X_r. \tag{3.1.4}$$

Fasst man die Eigenformen in einer Matrix zusammen, so erhält man die Modalmatrix

$$\Phi = (X_1\, X_2 \cdots X_n). \tag{3.1.5}$$

Aufgrund der Orthogonalität der Eigenvektoren, die im Kapitel 3.6 zur Berechnung der Eigenwerte nachgewiesen wird, kann man mit Hilfe dieser Matrix eine orthogonale Projektion der Systemmatrizen vornehmen. Durch diese Diagonalisierung der Massenmatrix M und Steifigkeitsmatrix K wird das System der Bewegungsgleichungen entkoppelt. Dies bedeutet, dass man die Differenzialgleichung der Schwingung in N eindimensionale Wellengleichungen zerlegt.

$$\Phi^T M \Phi = diag(\mu_r) \tag{3.1.6}$$
$$\Phi^T K \Phi = diag(\gamma_r) \tag{3.1.7}$$

Die Werte auf den Diagonalen der Matrizen μ und γ bezeichnen dann eine modale Massenbelegung und einen entsprechenden Faktor für die Steifigkeit der Struktur. Da die Normierung der Eigenformen frei wählbar ist, kann man das System noch vereinfachen, indem man die Projektionsmatrix mit $\frac{1}{\sqrt{\mu_r}}$ multipliziert und $\omega^2 = \gamma/\mu$ erhält.

$$\bar{\Phi}^T M \bar{\Phi} = I, \tag{3.1.8}$$
$$\bar{\Phi}^T K \bar{\Phi} = diag(\omega_r^2). \tag{3.1.9}$$

Im Folgenden wird diese normierte Transformationsmatrix wieder mir Φ bezeichnet. Durch die Transformation und Ersetzen von $U = \Phi X$ erhält man nun aus (3.1.2) ein System von entkoppelten Bewegungsgleichungen für X.

$$\ddot{X} + \Omega^2 X = 0 \tag{3.1.10}$$

Ω^2 steht hier für die Diagonalmatrix mit den Eigenwerten auf der Diagonale.
Betrachtet man nun ein inhomogenes System, bei dem eine äußere Kraft F aufgebracht wird, dann muss diese entsprechend mit der Modalmatrix bzw. für jede Mode mit dem entsprechenden Eigenvektor Φ_k transformiert werden.

$$r_k = \Phi_k^T F. \tag{3.1.11}$$

Die einzelnen Gewichtsfaktoren x_k ergeben sich nun aus der Lösung der eindimensionalen Gleichung

$$\ddot{x}_k + \omega_k^2 x_k = r_k. \tag{3.1.12}$$

Um letztendlich die Verformung des Systems zu einem Zeitpunkt t zu erhalten, muss man die Eigenformen, multipliziert mit dem entsprechenden Gewichtsfaktor, überlagern. Die Eigenform k als k-te Spalte der Modalmatrix wird nun mit Φ_k bezeichnet. Für die Überlagerung alle N Moden bedeutet dies

$$U(t) = \sum_{k=1}^{N} \Phi_k \cdot x_k(t). \tag{3.1.13}$$

Ein wesentlicher Vorteil der Modalanalyse besteht darin, dass die Berechnung erheblich vereinfacht werden kann, indem nur ein Teil der Eigenformen berücksichtigt wird. Beschränkt man die Zahl der Moden auf $K < N$, so muss man zum einen nur die ersten K Eigenwerte und Eigenvektoren der Matrizen berechnen, zum anderen nur diese beschränkte Anzahl an modalen Bewegungsgleichungen lösen. Dieser Ansatz findet seine Berechtigung darin, dass in vielen schwingenden Systemen die oberen Eigenfrequenzen einen sehr geringen Anteil am Schwingungsverhalten der Struktur aufweisen. Meist genügt für eine gute numerische Approximation schon eine Zahl von 5 berücksichtigten Moden.

3.2 Einbeziehung der Dämpfung

Bei den Betrachtungen der Modalanalyse wurde die allgemeine Gleichung 3.1.1 zunächst ohne den Dämpfungsterm mit der Dämpfungsmatrix C betrachtet. Unter bestimmten Bedingungen kann diese auch bei den Berechnungen der Eigenformen vernachlässigt werden. Man kann zeigen, dass dies genau dann der Fall ist, wenn die unten aufgeführte Bequemlichkeitshypothese 3.2.2 angewendet werden kann. Man führt in diesem Fall die bereits bekannte Transformation mit der Modalmatrix durch.

$$\Delta = \Phi^T C \Phi. \tag{3.2.1}$$

Diese Transformation erzeugt üblicherweise keine Diagonalmatrix, da die Eigenvektoren aus der Modalmatrix nur zu den Matrizen K und M aus 3.1.1 diagonalisiert werden kann. Um dennoch ein entkoppeltes System für die Berechnung der Gewichtsfaktoren x_k zu erhalten, fordert man nun, das die Einträge der Matrix, die nicht auf der Diagonale liegen, verschwinden. Dies wird dann als die *Bequemlichkeitshypothese* bezeichnet.

$$\Delta_{ij} = 0,\ i \neq j \tag{3.2.2}$$

Diese Annahme stellt zwar eine starke Einschränkung an mögliche äußere Dämpfungsfaktoren dar, die natürliche Dämpfung der Struktur selbst kann dadurch jedoch ausreichend gut beschrieben werden. Die gedämpften modalen Bewegungsgleichungen des entkoppelten Systems ergeben sich mit diesen Annahmen zu

$$\ddot{x}_k + \Delta_k \dot{x}_k + \omega_k^2 x_k = r_k \tag{3.2.3}$$

Die Werte Δ_k in dieser Gleichung bezeichnen die verbleibenden Diagonaleinträge der Dämpfungsmatrix und lassen sich nun aus dem Dämpfungsterm ξ (vgl. 3.2.11) bestimmen, falls die Eigenwerte des Systems paarweise verschieden sind. Gibt man für jede Mode k einen Faktor für die Dämpfung an, so gilt nach [13], Kap. 10

$$\xi_k = \frac{\Delta_{kk}}{2\omega_k},\ \frac{\lambda_k}{\lambda_j} < 1. \tag{3.2.4}$$

Durch Auflösen nach Δ_k und Einsetzen in Gleichung 3.2.3 erhält man nun eine Gleichungssystem, welches direkt vom Dämpfungsfaktor ξ_k abhängig ist.

$$\ddot{x}_k + 2\omega_k \xi_k \dot{x}_k + \omega_k^2 x_k = r_k. \tag{3.2.5}$$

Durch die Dämpfung verschieben sich auch die Eigenfrequenzen des Systems. Diese können bei der Anwendung der Bequemlichkeitshypothese jedoch aus den Frequenzen des ungedämpften Systems berechnet werden. Man führt dazu den Faktor ν_k als multiplikativen Faktor für die jeweilige Frequenz ein, der nach der Gleichung

$$\nu_k^2 = 1 - \xi_k^2 \tag{3.2.6}$$

berechnet wird. Tritt nun eine Anregung $r_k \neq 0$ auf, was gleichbedeutend mit dem Aufbringen einer Last auf die Struktur ist, so erhält man daraus eine neue Gleichgewichtslage x^{Gl}.

$$x_k^{Gl} = \begin{cases} 0 & \text{für } t < 0, \\ \frac{r_k}{\omega^2} = x_k^0 & \text{für } t \geq 0. \end{cases} \tag{3.2.7}$$

Somit kann man auch eine Lösung für die gedämpfte Bewegungsgleichung angeben (vgl. [14, Kap. 5]).

$$x_k = x_k^0 + C_k \cdot e^{-\xi\omega_k t} \cos(\ \nu\omega_k t - \varphi_k). \tag{3.2.8}$$

Die Faktoren C_k, φ_k werden aus den Anfangsbedingungen berechnet, was in Kapitel 4.2 noch ausführlich erläutert wird (vgl. Gleichung 4.2.21 und 4.2.22).

Bei Anwendung der Bequemlichkeitshypothese spricht man auch von einem *modal proportiolnal* gedämpften System oder auch von der Rayleigh-Dämpfung. Die Dämpfung ist dabei proportional zu den Systemmatrizen für die Massen und die Steifigkeit (vgl. [4, Kap 9.3]).

$$C = \alpha M + \beta K. \tag{3.2.9}$$

Diese Zerlegung mit den Faktoren $\alpha, \beta \in \mathbb{R}$ ist möglich, da in 3.2.2 auch die Diagonalgestalt von C gefordert wird. Die Transformation mit der Modalmatrix Φ der Eigenvektoren aus 3.2.1 nimmt dann für die Dämpfungsmatrix folgende Form an:

$$\Delta = \Phi^T C \Phi = \alpha \Phi^T M \Phi + \beta \Phi^T K \Phi = \alpha I + \beta \Omega^2 \tag{3.2.10}$$

Auch hier wirkt sich wiederum die Orthogonalität der Eigenvektoren bezüglich M und K aus und bei Hinzunahme der Gleichung 3.2.4 kann man daraus eine Gleichung erhalten aus der sich die Dämpfungsfaktoren des Systems bestimmen lassen.

$$\xi_k = \frac{1}{2}(\frac{\alpha}{\omega_k} + \beta\omega_k) \tag{3.2.11}$$

Man kann somit bei Kenntnis von mindestens 2 der Dämpfungsfaktoren ξ_k die Proportionalitätsfaktoren α und β bestimmen und daraus alle anderen Dämpfungswerte berechnen. Ein Beispiel dafür findet sich in [4], Beispiel 9.9. Diese Proportionalität der Dämpfunsparameter tritt in der Praxis jedoch eher selten auf (vgl. [13, Kap. 10]) und man muss daher oft auf experimentelle Methoden zur Bestimmung dieser Faktoren zurückgreifen.

3.3 Zeitdiskretisierung

Das Verfahren der Modalanalyse beruht auf der Idee, die Bewegungsgleichungen durch eine modale Transformation in einzelne Eigenformen zu zerlegen. Die allgemeine Lösung erhält man dann durch Überlagerung dieser Eigenformen mit entsprechenden Gewichtsfaktoren. Um diese zu erhalten, muss man jedoch noch die entkoppelten Bewegungsgleichungen (3.1.12) für jede Mode berechnen. Diese Gleichungen enthalten noch Ableitungen nach der Zeit und müssen für ein vorgegebenes Zeitintervall gelöst werden. Hier werden zwei verschiedene Verfahren vorgestellt, die auch in der Implementierung unterstützt werden.

3.3.1 Trigonometrischer Ansatz

Aus der modalen Transformation erhält man ein System von gewöhnlichen Differentialgleichungen der Form

$$\ddot{x}_k + \omega_k^2 x_k = r_k. \tag{3.3.1}$$

Diese Gleichung kann unter Einbeziehung der Anfangsbedingungen mit einem trigonometrischen Ansatz gelöst werden. Für jede Mode k erhält man dann eine Gleichung für x_k, die von der Gleichgewichtslage x^0 und den üblichen Werten C und φ für die Amplitude und die Phasenverschiebung abhängig ist.

$$x_k = x_k^0 + C_k \cdot \cos(\omega_k t - \varphi_k). \tag{3.3.2}$$

Tritt zudem ein positiver Dämpfungsfaktor $\xi < 1$ auf, so muss ein Dämpfunsterm auf den Term für die Schwingung multipliziert werden und die Gleichung verändert sich zu

$$x_k = x_k^0 + C_k \cdot e^{-\omega_k \xi t} \cos(\nu \omega_k t - \varphi_k). \tag{3.3.3}$$

Dabei wird die Schwingung durch den Faktor $e^{-\omega_k \xi t}$ mit der Zeit abklingen und die Eigenfrequenzen werden durch $\nu = \sqrt{1 - \xi^2}$ verschoben. Berechnet man mit Hilfe des gegebenen Dämpfungsfaktors ξ die neuen Eigenfrequenzen $\nu\omega_k$ und aus den Vektoren für die Last und die Impulskraft die Faktoren C_k und φ_k, sowie die Ruhelage x_k^0, so kann man die Gleichung in Abhängigkeit von der Zeit t lösen. Da bei numerischen Berechnungen immer eine Diskretisierung notwendig ist, muss

man noch einen Zeitschritt Δt festlegen, bei dem eine neue Position der Struktur berechnet werden soll. Dies kann unter Einbeziehung der Eigenfrequenzen geschehen, da durch die stärkere Dämpfung höherer Frequenzen die zum kleinsten Eigenwert λ_1 gehörende Eigenfrequenz die Schwingungsperiode maßgeblich beeinflusst. So kann gerade bei längerer Schwingungsdauer oder einer Anregung ohne Impulsanteil beispielsweise ein Zeitschritt $\Delta t = \frac{T}{10} = \frac{2\pi}{10\omega_1}$ für ein Zehntel der Schwingungsperiode T verwendet werden. Da bei Aufbringen einer Impulskraft die hohen Frequenzen stärker angeregt werden können, empfiehlt sich hier ein feinerer Zeitschritt für die Berechnungen.
Zur Ergänzung der Analyse des Schwingungsverhaltens bei gedämpften Systemen wird hier noch das Verhalten bei dem Dämpfungsfaktor $\xi = 1$ betrachtet. Dies beschreibt einen Grenzfall, bei dem keine Schwingungen mehr auftreten, sondern ein kriechender Übergang in die neue Gleichgewichtslage beobachtet werden kann. Die Lösung der Bewegungsgleichungen wird dann beschrieben durch

$$x_k = x_k^0 - (x_k^0(1 + \omega_k t) + r_k \omega_k t) \cdot e^{-\omega_k t}. \tag{3.3.4}$$

Man erkennt, dass der Kosinus-Anteil aus der Gleichung verschwindet und nach Abklingen der Exponentialfunktion die Ruhelage x_k^0 erreicht wird.

3.3.2 Finite Differenzen Methode

In der Finite Differenzen Methode (FDM) unterteilt man das gegebene Gebiet, also in diesem Fall das Zeitintervall, in eine endliche Anzahl von Teilgebieten. Die Lösung der Gleichung erfolgt dann zu den Zeitpunkten an den Rändern dieser Teilintervalle. Für eine Unterteilung in gleich große Zeitabstände Δt wird folglich für jeden Zeitschritt $i\Delta t,\ i = 0, \ldots, N$ bei N Intervallen eine Lösung berechnet. Dies geschieht durch die Einführung eines Differenzenquotienten, der den Wert für die 2. Zeitableitung approximiert.

$$\ddot{x} = \frac{x^+ - 2x + x^-}{\Delta t^2}. \tag{3.3.5}$$

Dabei bezeichnen x^+ und x^- den Wert x beim nächsten bzw. vorherigen Zeitschritt. Man führt also eine Zeitdiskretisierung mit dem Zeitschritt Δt durch.
Nun kann man die Approximation in die Bewegungsgleichung (3.1.12) einsetzen und so umformen, dass man x^+ in Abhängigkeit von x und x^- berechnen kann.

$$x^+ = 2x - x^- - \Delta t^2 \omega^2 x + \Delta t^2 r. \tag{3.3.6}$$

Man erhält somit ein direktes Verfahren zur Berechnung der Gewichtsfaktoren in jedem Zeitschritt der Modalanalyse. Falls eine Anfangsbedingung $f(x) \neq 0$ für $t = 0$ vorgegeben ist, kann diese für die Initialisierung von x verwendet werden. Die Anfangswerte für die modalen Gewichtsfaktoren ergeben sich aus der modalen Transformation $\Phi^T f(x)$.

Da zu Beginn noch keine Informationen von x^- verfügbar sind, ergänzt man für den ersten Schritt eine Gleichung mit symmetrischen Anfangsbedingungen $x^- = x^+$ und erhält nun x^+ in Abhängigkeit von x:

$$x^+ = x + \frac{1}{2}\Delta t^2(r - \omega^2 x). \tag{3.3.7}$$

Betrachtet man nun die Gleichung (3.2.3) für die Gewichtsfaktoren eines gedämpften Systems, so benötigt man für den Dämpfungsterm noch einen Differenzenquotienten für die erste Ableitung nach der Zeit. Dieser ist gegeben durch:

$$\dot{x} = \frac{x^+ - x^-}{\Delta t}. \tag{3.3.8}$$

Setzt man die beiden Differenzenquotienten nun in Gleichung (3.2.3) ein, so erhält man:

$$\frac{x^+ - 2x + x^-}{\Delta t^2} + D\frac{x^+ - x^-}{\Delta t} + \omega^2 x = r. \tag{3.3.9}$$

D bezeichnet hier das Dämpfungsmaß Δ_k der betrachteten Eigenform. Dieses ergibt sich bei Annahme der Bequemlichkeitshypothese (3.2.2) und Einführung des modalen Dämpfungsmaßes ξ aus (3.2.4) zu $D = 2\omega\xi$.
Nach Umformung ergibt sich nun die Gleichung für die Gewichtsfaktoren des gedämpften Systems:

$$x^+ = \frac{2x - x^- + \Delta t D x^- - \Delta t^2\omega^2 x + \Delta t^2 r}{1 + D\frac{\Delta t}{2}}. \tag{3.3.10}$$

3.4 Implementierung in Diffpack

Die Simulation der Schwingungen einer Struktur mit Hilfe der Modalanalyse wird in *Diffpack* durch eine neue Klasse *ModalAnalysis* erreicht. Diese wird von der bestehenden *Diffpack*-Klasse `FEM` abgeleitet, um die Methoden der Finiten Elemente nutzen zu können. Die wesentlichen Funktionen wie `solveProblem` werden in der neuen Klasse überschrieben und können wie bei üblichen `FEM`-Klassen genutzt werden. Die Methoden und die Funktionsweise die Klasse `ModalAnalysis` wird nun im Folgenden beschrieben.

```
defineStatic( Handle( MenuSystem ) menu , int level )
```

Zur Definition aller Parameter für die Modalanalyse, wie z.B. die Anzahl der zur Berechnung verwendeten Moden, wird die Funktion `defineStatic` oder `define` genutzt. Da diese Werte über das globale Menü des Systems gesetzt werden, wird hier ein gesonderter Zeiger (Handle) auf dieses Menü übergeben. Das `level` gibt die Ebene an, in der sich das Untermenü für die neuen Einträge befindet.
Es können folgende Parameter gesetzt werden:

- *Number of Modes*
  Integer Wert für die Anzahl der verwendeten Moden.
- *time parameters*
  String mit Zeitinterval und Zeitschritt der Simulation.
- *dampFactor*
  Double Wert des modalen Dämpfungsparameters.
- *time solver*
  String für Bezeichner des Lösungsverfahrens zur Zeitdiskretisierung.
- *plot_Node*
  Integer Wert für Knoten dessen Verschiebung separat gespeichert wird.

`scan()`

Genau wie die Klasse des Simulators (vgl. Kapitel 2.4) besitzt auch die Klasse `ModalAnalysis` die Funktion `scan`, um die Variablen für das Untermenü aus dem *InputFile* einzulesen. Diese werden dann in globalen Parametern gespeichert und können in allen Methoden der Klasse verwendet werden. Zudem werden hier alle Felder und Objekte für die Freiheitsgrade, den Eigenwertlöser, etc. angelegt.

`setInitialField( FieldsFE &u_init )`

Bevor die Simulation einer Schwingung gestartet werden kann, werden die Anfangsbedingungen des Systems benötigt. Dies kann eine Anfangsauslenkung, eine Last auf die Struktur oder ein Anfangsimpuls zur Zeit $t = 0$ sein. Befindet sich das System zum Startzeitpunkt nicht in der Ruhelage, so wird dieser Zustand hier mit Hilfe der Funktion `setInitialField` gesetzt. In einem System mit einem Freiheitsgrad pro Knoten, wird dieser Wert in dem `FieldFE u_init` der entsprechenden Größe festgehalten. Bei Mehrfreiheitsgradsystemen übergibt man ein Objekt vom Typ `FieldsFE`, das mehrere Felder repräsentiert. Im später behandelten Beispiel zur Schwingung eines Balkens findet sich auch noch die Möglichkeit ein Objekt der abgeleiteten Klasse `FieldHFE` zu verwenden, das die Besonderheiten der hermiten Basisfunktionen berücksichtigt.

`setSource( FieldsFE &source )`

Wie in der vorherigen Funktion beschrieben, ist es auch möglich die Schwingungen durch Aufbringen einer Last auf die Struktur zu berechnen. Diese Last wird durch das `FieldFE source` beschrieben, es kann also für jeden Knoten eine separate Last angegeben werden. Genau wie im Fall der Anfangsauslenkung wird das System durch ein `FieldsFE` auf mehrere Freiheitsgrade angepasst.

`setImpulse( FieldsFE &impulse )`

Wenn die Schwingung durch einen Impuls zum Zeitpunkt $t = 0$ angeregt wird, wird dieser Impuls durch die Werte der Impulskraft auf jeden Knoten in dem `FieldFE impulse` widergespiegelt. Dieser Impuls wirkt auf die Struktur wie eine Anfangsgeschwindigkeit $\dot{U}$, die das System in Schwingung versetzt. Wiederum können durch ein `FieldsFE` mehrere Freiheitsgrade angesprochen werden, oder impulsartig auftretende Biegemomente berücksichtigt werden.

`calcEV( Matrix(NUMT) &K , Matrix(NUMT) &M )`

Bevor die Überlagerung der Moden zu den gegebenen Zeitpunkten berechnet werden kann, müssen diese mit Hilfe einer Eigenwertanalyse approximiert werden. Dies geschieht in der Funktion `calcEV`, der die beiden Systemmatrizen übergeben werden. Hier findet auch der Aufruf der C++-Klasse `Eigensolver` statt, die eine Anbindung zur Softwarebibliothek *Arpack* bietet und die Eigenvektoren und Eigenwerte des verallgemeinerten Eigenwertproblems liefert. Diese werden dann in einer globalen Matrix `eigenvectors` und in dem entsprechenden Vektor `eigenvalues` abgelegt. Die gespeicherten Eigenvektoren und die Eigenwerte werden zuletzt noch in ein Textfile **EV.txt** geschrieben, wobei die Werte noch durch eine zusätzliche Funktion sortiert werden, um eine aufsteigender Reihenfolge zu gewährleisten.

`solveProblem( )`

Dies ist die wesentliche Funktion zur Berechnung und Überlagerung der Moden. Dazu werden die Eigenwerte und Eigenvektoren der Systemmatrizen benötigt, was voraussetzt, dass zuvor `calcEV` erfolgreich ausgeführt wurde. Eine sinnvolle Simulation erhält man nur, wenn vor dem Aufruf dieser Funktion mindestens eine Anfangsbedingung gesetzt wurde. Der triviale Fall $U \equiv 0$ wird hier bereits abgefangen und keine Berechnungen ausgeführt. Es werden nun der Reihe nach die Funktionen `init`, `calcTip` und `timeLoop` aufgerufen, um Schritt für Schritt die modale Überlagerung zu berechnen.

`init()`

Die Initialisierung aller in der Modalanalyse verwendeten Variablen findet in der Funktion `init` statt. Um die Gewichtsfaktoren für die modale Überlagerung zur Zeit $t = 0$ zu bestimmen, werden hier alle gesetzten Anfangsbedingungen mit der aus den Eigenvektoren bestehenden Modalmatrix transformiert. Im Falle einer

Anfangsauslenkung außerhalb der Ruhelage muss zudem die spezielle Normierung der Eigenvektoren berücksichtigt werden, um die modalen Faktoren zum Startzeitpunkt zu erhalten. Die Anfangswerte für Gewichtsfaktoren werden dann in einem Vektor $x \in \mathbb{R}^K$ für die K verwendeten Moden abgelegt. Entsprechend wird ein Vektor x_0 für die Faktoren der durch die Last hervorgerufenen neuen Ruhelage und ein Vektor r für die transformierte Impulskraft initialisiert. Die Methode gibt dann die modale Transformation der gesetzten Parameter aus und speichert die einzelnen Moden bzw. Eigenvektoren als `CurvePlotFile`.

`calcTip()`

Falls im *InputFile* keine Parameter für das Zeitintervall und die Zeitschritte übergeben wurden, werden diese mit Hilfe der Eigenfrequenzen des Systems bestimmt. Üblicherweise ist man bei Schwingungen in Strukturen besonders an den tiefen Frequenzen interessiert, was hier berücksichtigt wird, indem der Zeitschritt und die Periode auf der Basis des kleinsten Eigenvektors berechnet wird. Dies kann gerade bei Verwendung der Methode der *Finiten Differenzen* zur Instabilität führen, falls sehr viel größere Frequenzen in die Berechnung mit einbezogen werden. Daher sollte bei Auftreten solcher nicht stabilen Zeitschritte die Schrittweite noch angepasst werden.

`timeLoop( )`

Die Berechnung der modalen Überlagerung in den diskreten Zeitschritten wird in der `timeLoop`-Methode durchgeführt. Diese beschreibt die Berechnung der Lösung für jeden vorgegebenen Zeitschritt Δt. Dazu werden auch einige Zeitunabhängige Faktoren benötigt, wie zum Beispiel die Anfangsamplitude und Phasenverschiebung der Schwingung. Durch den Aufruf von `setCondForTimeloop` werden diese Parameter global berechnet und stehen somit bei der Berechnung der Überlagerung zur Verfügung. Die Berechnung selbst findet dann in der Methode `solveAtThisTimestep` statt, die in einer `while`-Schleife ausgeführt wird, bis der Endzeitpunkt erreicht ist.

```
while( !tip->finished( ) )
{
   tip->increaseTime( );
   solveAtThisTimestep( ) ;
   resultReport( ) ;
}
```

Um auch die Lösung zur Startzeit t_0 zu erhalten, wird zuvor noch das berechnete Startfeld `D` in die Datenbank geschrieben. Die Ergebnisse der gesamten Modalanalyse werden dann durch die Methode `resultReport` dokumentiert.

`setCondForTimeloop( )`

Die entkoppelten Bewegungsgleichungen, die für die Berechnung der modalen Gewichtsfaktoren noch gelöst werden müssen, enthalten die Faktoren C_k und φ_k für die Amplitude und die Phasenverschiebung der Schwingungen (vgl. 3.4.4). Diese werden üblicherweise aus zwei weiteren Faktoren A_k und B_k bestimmt, die wiederum von den Anfangsbedingungen der globalen Bewegungsgleichung abhängen. Sie entsprechen dabei den Faktoren in den Vektoren x und r aus der modalen Transformation, die in `init` durchgeführt wurde. Daraus berechnen sich dann C_k und φ_k für $A_k \neq 0$ zu:

$$C_k = \sqrt{{A_k}^2 + {B_k}^2}, \tag{3.4.1}$$

$$\varphi_k = \tan^{-1}\left(\frac{B_k}{A_k}\right). \tag{3.4.2}$$

Die Faktoren x_k^0 repräsentieren die durch die Last entstehende Gleichgewichtslage der Struktur. ω_k bezeichnet die aus den Eigenwerten berechneten Eigenfrequenzen des Systems. Mit Hilfe dieser Parameter kann nun die modale Bewegungsgleichung gelöst werden.

`solveAtThisTimestep( )`

Bei der Berechnung der Überlagerung der Moden zu einem Zeitpunkt wird im Wesentlichen zwischen einem Lösungsverfahren mit trigonometrischen Funktionen und dem der *Zentralen Finiten Differenzen* unterschieden.
Der erste Ansatz basiert auf der analytischen Lösung der Wellengleichung, die hier für den Gewichtsfaktor jeder einzelnen Mode gelöst werden muss. Man erhält die Gewichtsfaktoren x_k zum Zeitpunkt t durch folgende Gleichung für die Mode k:

$$x_k(t) = A_k \cdot \sin(\omega_k\ t) + B_k \cdot \cos(\omega_k\ t) \tag{3.4.3}$$

Zur Vereinfachung können die beiden trigonometrischen Funktionen noch zusammengezogen werden und es ergibt sich eine Schwingungsgleichung mit der Amplitude C_k und einer Phasenverschiebung φ_k:

$$x_k(t) = x_k^0 + C_k \cdot \cos(\omega_k\ t - \varphi_k). \tag{3.4.4}$$

Die Faktoren C_k und φ_k, sowie x_0 wurden bereits in `setCondForTimeloop` berechnet und können unabhängig vom Zeitschritt t verwendet werden.
Eine gerade im Hinblick auf die Einbeziehung der Dämpfung einfachere Methode zur Lösung der entkoppelten Wellengleichungen liefert das als Alternative implementierte *Finite Differenzen Verfahren*. Hier wird der Gewichtsfaktor x_k^+ zum

nächsten Zeitschritt $t^+ = t + \Delta t$ für die Mode k mit Hilfe eines Diffenenzenquotienten aus den Faktoren x und x^- berechnet.

$$x_k^+ = 2x_k - x_k^- + \Delta t^2 \left(r_k - \omega_k^2 x_k\right) \tag{3.4.5}$$

Die ausführlicheren Gleichungen mit Dämpfungsterm befinden sich im Abschnitt 3.3 zur Zeitdiskretisierung.
Die approximierte Lösung der Bewegungsgleichung 3.1.1 wird nun durch die Überlagerung aller betrachteten Moden, multipliziert mit dem entsprechenden Gewichtsfaktor, berechnet. Zum Zeitpunkt t erhält man die Verschiebung

$$u(t) = \sum_{k=1}^{K} x_k(t) \cdot \mathrm{EV}_k. \tag{3.4.6}$$

K steht hierbei für die Anzahl der betrachteten Moden. Neben der Speicherung des resultierenden Feldes durch `resultReport` für den aktuellen Zeitschritt wird bei Bedarf noch der Wert des im Menü angegebene Knoten in ein `CurvePlotFile` geschrieben. Code-Ausschnitte befinden sich im Abschnitt der Implementierung mit Dämpfung, der noch die Erweiterungen zur Berechnung eines gedämpften Systems enthält.

`resultReport()`

Am Ende jeder Schleife in der `timeLoop`-Funktion werden die Ergebnisse durch die Methode `resultReport` gespeichert. Dazu wird das Objekt `database` der Klasse `SaveSimRes` verwendet, das die Methode `dump( f , *tip)` besitzt, um die Werte für jedes Feld `f` zum entsprechenden Zeitpunkt in eine Datei mit der Endung `.field` zu schreiben. Für jeden Zeitschritt und jeden Freiheitsgrad wird dabei ein eigenes Feld gespeichert. *Diffpack* verfügt über Skripten wie `simres2vtk`, um diese Felder zu visualisieren. Diese Funktion kann bei bedarf im Simulator selbst überschrieben werden, wenn dieser von `ModalAnalysis` abgeleitet wird. Somit können auch beliebige Felder zu unterschiedlichen Zeitschritten ausgegeben werden oder für weitere Berechnungen herangezogen werden.

3.5 Implementierung mit Dämpfung

Um die Dämpfung in modaler Form bei der Implementierung zu berücksichtigen, muss man die Bewegungsgleichungen mit Dämpfungsterm aus Gleichung 3.2.5 betrachten.

Für den Fall der Finiten Differenzen als Lösungsmethode kann man die Berechnung der Gewichtsfaktoren anpassen, wie in Kapitel 3.3 zur Zeitdiskretisierung beschrieben. Dabei kommt im Wesentlichen nur ein Term mit dem modalen Dämpfungsmaß $D = 2\omega\xi$ hinzu, der jedoch eine zusätzliche Division in der

Gleichung für den nächsten Zeitschritt notwendig macht. Zur besseren Übersicht ist bei der Implementierung die Berechnung des Gewichtsfaktors x_k in mehrere Schritte aufgeteilt:

```
for( k = 1 ; k <= iNoModes ; k ++ )
{
   damping = tip->Delta() * sqrt(eigenvalues(k)) * m_dDampFac ;
   fac1  = 2 * x(k) - xm(k) ;
   fac2  = tsquare * eigenvalues(k) * x(k) ;
   fac3  = tsquare * x_0(k) ;
   fac4  = damping * xm(k) ;
   fac5  = 1 + damping ;

   xp(k) = ( fac1 - fac2 + fac3 + fac4 ) / fac5 ;
}
```

Die Faktoren `fac1` bis `fac3` sind äquivalent zu der Berechnung ohne Dämpfung, wobei `tsquare` das Quadrat des Zeitschrittes Δt^2 bezeichnet. Die globale Variable `m_dDampFac`, die den im Menü gesetzten modalen Dämpfungsfaktor enthält, entspricht dem Wert ξ aus Gleichung 3.2.4. Daraus wird mit Hilfe der approximierten Eigenwerte das Dämpfungsmaß, bzw. der für die Berechnung benötigte Wert `damping` $= \Delta t \cdot \omega \cdot \xi$ berechnet. Durch Aufsummieren der Faktoren 1 bis 4 und die Division durch Faktor 5 wird der neue Gewichtsfaktor x_k^+ bestimmt.

Wird die Lösung der Bewegungsgleichung mit Hilfe trigonometrischer Funktionen berechnet, so muss man einen zusätzlichen Exponentialterm einführen, um die Dämpfung zu berücksichtigen. Die Lösung für die Gewichtsfaktoren x hat nun folgende Form:

$$x = x_k^0 + C_k \cdot e^{-\xi\tau} \cdot \cos(\nu\tau - \varphi_k). \tag{3.5.1}$$

Dabei entspricht der Wert τ der dimensionslosen Zeit $\tau = \omega t$ und der Faktor ν, bestimmt durch die Gleichung $\nu^2 = 1 - \xi^2$, bezeichnet die Abweichung der gedämpften Eigenfrequenzen von denen des ungedämpften Systems. Diese Berechnung ist jedoch nur möglich, wenn man die Bequemlichkeitshypothese (Gleichung 3.2.2) für die Dämpfungsmatrix voraussetzt. Für die Implementierung bedeutet dies, dass zunächst die Dämpfungsfaktoren für die Frequenzen berechnet werden müssen, die wiederum aus der Wurzel der Eigenwerte der Matrizen gebildet werden.

```
ny(k)    = sqrt( fabs( 1 - sqr( m_dDampFac(k) ) ) ) ;
omega(k) = sqrt( fabs( eigenvalues(k) ) ) ;
```

In den behandelten Beispielen wurde nur ein einziger Dämpfungsfaktor für alle Frequenzen verwendet, wodurch auch nur ein einzelner Faktor ν notwendig ist. Da

hier nur der periodische Fall mit $\xi < 1$ betrachtet wird, kann man auch den Absolutbetrag `fabs` in der Berechnung von ν weglassen. Für jeden Zeitschritt werden nun die Gewichtsfaktoren für jede Mode durch die Gleichung (3.5.1) bestimmt und die Überlagerung $\sum x_k(t) \cdot EV_k$ berechnet.

```
for( k = 1 ; k <= m_iNoModes ; k ++ )
{
   if( m_dDampFac > 0.0 && m_dDampFac < 1.0 )
   {
      damping = exp( -1*omega(k)*m_dDampFac*tip->time() ) ;
      x(k) = x_0(k) + vC(k) * damping
              * cos( ny(k) * omega(k) * tip->time() - vPhi(k) ) ;
   }
}

for( k = 1 ; k <= m_iNoModes ; k ++ )
{
   eigenvectors.getRow( k , curEV ) ;
   curEV.mult( x(k) ) ;
   fieldV.add( fieldV , curEV ) ;
}
dof->vec2field( fieldV , *D ) ;
```

Die Zeit t des aktuellen Zeitschrittes wird durch die Methode `time` der Klasse `TimePrm` zur Verfügung gestellt. Die Vektoren für die Amplitude `vC` und die Phasenverschiebung `vPhi` sind analog zum ungedämpften Fall (vgl. Abschnitt 3.4). Für das Abklingen der Schwingung ist der Term `damping` verantwortlich, der dem Wert $e^{-\xi_k \omega_k t}$ entspricht. Zusätzlich zur ungedämpften Schwingung mit `m_dDampFac` $= 0$ wurde noch der Fall der Dämpfung bei dem Grenzfall mit einem Wert von 1 implementiert, andere kriechende Systeme mit größeren Werten werden nicht betrachtet.
Die zur Überlagerung benötigten Eigenvektoren befinden sich in den Zeilen der Matrix `eigenvectors` und werden in den Vektor `curEV` übertragen. Das resultierende Feld wird dann zunächst im Vektor `fieldV` aufsummiert und anschließend in das `FieldsFE`-Objekt `D` gespeichert. Die dafür notwendige Funktion `vec2field` wird durch das Objekt `DegFreeFE dof` geliefert, was sicherstellt, dass alle Freiheitsgrade des Systems im Feld repräsentiert werden.

3.6 Eigenwert-Berechnung

3.6.1 Grundlagen

Wie aus den vorherigen Kapiteln hervor geht, spielt bei der Analyse von Verschiebungen und Bewegungsgleichungen die Berechnung der Eigenwerte und Ei-

genvektoren eine wichtige Rolle. Erst die Kenntnis diese charakteristischen Werte macht die modale Transformation und Überlagerung sogar erst möglich.
Hier sollen nun die wesentlichen Eigenschaften der Eigenformen und die Möglichkeiten der numerischen Berechnung aufgezeigt werden.

Definition. *Sei $A \in \mathbb{R}^{n\times n}$ eine reelle quadratische Matrix, dann heißt $\lambda \in \mathbb{C}$* Eigenwert *von A, wenn die Eigenwertgleichung*

$$Ax = \lambda x \tag{3.6.1}$$

erfüllt ist. Der Vektor $x \in \mathbb{C}^n$ heißt Eigenvektor *zum Eigenwert λ.*

Zur Berechnung von Eigenwerten und Eigenvektoren wird häufig folgende Charakterisierung verwendet, die auch zeigt, weshalb man auch bei reellen Matrizen komplexe Eigenwerte berücksichtigen muss.

Lemma: Eigenwert. *λ ist ein Eigenwert von A genau dann, wenn:*

$$det(A - \lambda I) = 0. \tag{3.6.2}$$

Dies folgt daraus, dass wegen 3.6.1 auch $(A - \lambda I)x = 0$ gilt und daher $(A - \lambda I)$ singulär ist, was gleichbedeutend mit 3.6.2 ist.
Dabei stellt die Funktion $det(A - \lambda I)$ gerade das *charakteristische Polynom* dar, dessen Nullstellen somit die Eigenwerte der Matrix A bilden. Die numerische Berechnung über das charakteristische Polynom ist jedoch aufgrund der schlechten Konditionierung des Problems sehr aufwendig und für Verfahren in der Praxis nicht geeignet. Man bedient sich daher eines weiteren Lemmas, das eine wichtige Aussage zum Spektrum einer Matrix enthält.

Lemma. *Sei $\sigma(A) := \{\lambda \in \mathbb{C} \mid det(A - \lambda I) = 0 \}$ das* Spektrum *der Matrix A. Betrachtet man zwei ähnliche Matrizen, d.h. ein $B = T^{-1}AT$ für eine beliebige nichtsinguläre $n \times n$-Matrix T, so gilt:*
Zwei ähnliche Matrizen haben das gleiche Spektrum, d.h.

$$\sigma(A) = \sigma(T^{-1}AT). \tag{3.6.3}$$

[Beweis [6, Kap. 7]]

Diese Eigenschaft kann man nun nutzen, um eine Matrix in obere Dreiecksform oder Diagonalform zu bringen, was es ermöglicht die Eigenwerte direkt abzulesen bzw. zu speichern. Das wesentliche Lemma hierzu ist benannt nach dem Mathematiker Issai Schur.

Lemma: Komplexe Schur-Faktorisierung (Schur-Normalform). *Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ gibt es eine unitäre Matrix $Q \in \mathbb{C}^{n\times n}$, so dass:*

$$Q^*AQ = \begin{pmatrix} \lambda_1 & & & & \\ & \lambda_2 & & * & \\ & & \lambda_3 & & \\ & 0 & & \ddots & \\ & & & & \lambda_n \end{pmatrix} := R. \tag{3.6.4}$$

Dabei ist $\{\lambda_1, \ldots, \lambda_n\} = \sigma(A)$.

Bei einer reellen Matrix A können auf der Diagonalen der Matrix R auch Blöcke der Dimension 2 auftreten, falls ein komplexer Eigenwert mit dem komplex konjugierten Gegenstück vorkommt. Wenn alle Eigenwerte paarweise verschieden sind, kann man sogar noch eine schönere Form der Matrix R erhalten, was die nächste Folgerung zeigt.

Folgerung. *Jede symmetrische Matrix $A \in \mathbb{R}^{n\times n}$ lässt sich mittels einer orthogonalen Matrix Q ähnlich in Diagonalgestalt bringen, d.h.*

$$Q^{-1}AQ = \operatorname{diag}(\lambda_1, \ldots, \lambda_n). \tag{3.6.5}$$

[Beweis [6, Kap. 7]]

A besitzt somit nur reelle Eigenwerte und die Spalten der Matrix Q bilden die zueinander orthogonalen Eigenvektoren der Matrix A.

3.6.2 Vektoriteration

Eine der einfachsten Methoden zur Berechnung des betragsmäßig größten Eigenwertes und des dazugehörigen Eigenvektors ist die *Vektoriteration* (oder *Potenzmethode*). Zwar bietet sie nur die Möglichkeit einen Eigenwert zu bestimmen, jedoch dient das Verfahren als Grundlage für leistungsfähigere Methoden zur Eigenwert- und Eigenvektorberechnung.

Für die folgenden Schritte sei vorausgesetzt, dass die Matrix A diagonalisierbar ist, d.h. eine Basis aus Eigenvektoren besitzt. Zudem werden die Eigenvektoren so skaliert, dass $\|v_i\|_2 = 1, i = 1, \ldots, n$ gilt und die Eigenwerte werden der Größe nach sortiert mit einfachem dominanten Eigenwert:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|\,. \tag{3.6.6}$$

Beschränkt man sich auf eine reelle Matrix A, so erhält man mit dem einfachen Eigenwert λ_1 gleichzeitig, dass λ_1 und v_1 reell sind. Man kann nun einen beliebigen Startvektor x_0 darstellen als Linearkombination aus Eigenvektoren:

$$x_0 = c_1 v_1 + c_2 v_2 + \ldots + c_n v_n, \tag{3.6.7}$$

wobei vorausgesetzt wird, dass $c_1 \neq 0$ gilt. Wendet man nun die k-te Potenz von A auf x_0 an, so erhält man:

$$A^k x_0 = \sum_{j=1}^{n} c_j A^k v_j = \sum_{j=1}^{n} c_j \lambda_j^k v_j. \tag{3.6.8}$$

Um den ersten Eigenwert zu bestimmen, zieht man diesen nun aus der Summe heraus:

$$x^k := A^k x_0 = \lambda_1 \left\{ c_1 v_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k c_j v_j \right\}. \tag{3.6.9}$$

Wegen $\frac{|\lambda_j|}{|\lambda_1|} \leq \frac{|\lambda_2|}{|\lambda_1|} < 1$ für $j = 2, \ldots, n$ konvergiert die Summe nun gegen Null für $k \to \infty$. Somit strebt $x^k = A^k x_0$ in Richtung des ersten Eigenvektors v_1. Für die Annäherung des betragsmäßig größten Eigenwertes kann man nun

$$\lambda^{(k)} = \frac{(x^k)^T A^k x^k}{\|x^k\|_2^2} = \frac{(x^k)^T x^{k+1}}{\|x^k\|_2^2} \tag{3.6.10}$$

verwenden. Bei genauerer Betrachtung (vgl. [6, Kap. 7]) erhält man eine Konvergenzrate von

$$\left| \lambda^{(k)} - \lambda_1 \right| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right). \tag{3.6.11}$$

Für den Algorithmus ist es noch zweckmäßig die Iterierten x^k auf 1 zu skalieren, da $\left\|x^k\right\| \to \infty$ für $\lambda_1 > 1$ und $\left\|x^k\right\| \to 0$ für $\lambda_1 < 1$ gilt. Die Resultate aus den vorherigen Gleichungen bleiben dabei unverändert.

Algorithmus 1 : Vektoriteration

Wähle Startvektor y^0 mit $\|y^0\|_2 = 1$
for $k = 0, 1, 2, \ldots$ **do**

$$\begin{aligned} \tilde{y}^{k+1} &= A y^k \\ \lambda^{(k)} &= (y^k)^T \tilde{y}^{k+1} \\ y^{k+1} &= \frac{\tilde{y}^{k+q}}{\left\|\tilde{y}^{k+q}\right\|_2} \end{aligned}$$

end

3.6.3 Inverse Vektoriteration

Nachdem mit der Vektoriteration eine einfach Methode zur Berechnung des betragsmäßig größten Eigenwertes und des entsprechenden Eigenvektors vorgestellt wurde, wird hier noch eine Erweiterung betrachtet, mit der auch andere Eigenpaare approximiert werden können. Dazu nutzt man aus, dass λ_i genau dann Eigenwert von A ist, wenn $\lambda_i - \mu$ Eigenwert von $A - \mu I$ ist, für ein beliebiges $\mu \in \mathbb{R}$. Findet man nun eine Annäherung $\mu \approx \lambda_i$ eines Eigenwertes mit

$$|\lambda_i - \mu| < |\lambda_j - \mu| \quad \text{für alle} \quad i \neq j, \tag{3.6.12}$$

dann ist $(\lambda_i - \mu)^{-1}$ der betragsmäßig größte Eigenwert von $(A - \mu I)^{-1}$. Man wendet nun also die Vektoriteration an auf die inversen, um den Wert μ verschobenen Eigenwerte. Der Algorithmus ergibt sich dann wie folgt:

Algorithmus 2 : Inverse Vektoriteration

Wähle Startvektor y^0 mit $\|y^0\|_2 = 1$
for $k = 0, 1, 2, \ldots$ **do**

$$\begin{aligned} (A - \mu I)\tilde{y}^{k+1} &= y^k \\ \lambda^{(k)} &= \frac{1}{(y^k)^T \tilde{y}^{k+1}} + \mu \\ y^{k+1} &= \frac{\tilde{y}^{k+q}}{\left\|\tilde{y}^{k+q}\right\|_2} \end{aligned}$$

end

Die Konvergenzrate dieses Verfahrens hängt nun von dem Abstand des nächsten (nicht gesuchten) Eigenwertes vom Wert μ ab, also von der Güte des Schätzwertes für den gesuchten Eigenwert. Das bedeutet, man kann die Konvergenzgeschwindigkeit noch erhöhen, falls man den Wert für die Spektralverschiebung μ nach jedem Schritt an die neueste Approximation anpasst. Verwendet man jedoch eine QR-Zerlegung für die Lösung des Gleichungssystems in Schritt 1 der for-Schleife im Algorithmus, so steigt der Rechenaufwand bei dieser Methode erheblich an.

3.6.4 Krylov-Unterraum-Projektion

Obwohl man durch die Spektralverschiebungen die Möglichkeit hat, beliebige Eigenwerte der Matrix A zu approximieren, so ist man mit der Vektoriteration jedoch immer noch nur in der Lage, einen einzelnen Eigenwert mit zugehörigem Eigenvektor zu bestimmen. Eine alternative Methode zur Berechnung von Eigenwerten stellen die Methoden der Krylov-Unterraum-Projektion dar, die auf einer Sequenz von orthogonalen Eigenvektoren basieren, die für gewöhnlich durch Verktoriterationen erzeugt werden.
Um die Eigenwerte zu approximieren betrachtet man die Krylov-Unterräume

$$\mathcal{K}_k(A, v_1) := Span\left\{v_1, Av_1, A^2v_1, A^3v_1, \ldots, A^{k-1}v_1\right\}. \tag{3.6.13}$$

Auf diesen Unterräumen definiert man nun die *Ritz-Paare* basierend auf einer so genannten Galerkin-Bedingung. Ein Vektor $x \in \mathcal{K}_k(A, v_1)$ heißt *Ritz-Vektor* mit entsprechendem *Ritz-Wert* θ, wenn die Gelerkin-Bedingung

$$\langle w, Ax - x\theta \rangle = 0 \ , \text{ für alle } w \in \mathcal{K}_k(A, v_1) \tag{3.6.14}$$

erfüllt ist.

Lemma. *(x, θ) ist ein Ritz-Paar $\Leftrightarrow x = Wy$ mit $W^T AWy = y\theta$, für eine orthonormale Basis W von $\mathcal{K}_k$.*

Dies folgt direkt aus der Galerkin-Bedingung, da $0 = W^T(AWy - Wy\theta) = W^T AWy - y\theta$ (vgl. [15, Kap. 3]).

Man kann nun zeigen, dass $\mathcal{K}_k$ ein A-invarianter Unterraum ist, genau dann wenn $v_1 = Vy$, wobei $AV = VR$ mit $V^T V = I_k$ und R obere $k \times k$ Dreiecksmatrix. Nun sucht man eine geeignete orthogonale Basis $V = WQ$, mit deren Hilfe man sukzessiv die Basisvektoren erzeugen kann. Es ist möglich eine unitäre $k \times k$-Matrix Q mit standard Householder Transformationen zu erhalten, so dass $v_1 = Ve_1$ und $H := Q^T W^T AWQ = Q^T BQ$ eine obere Hessenberg-Matrix mit nicht-negativen Einträgen unterhalb der Diagonale. Mit dieser Basis gilt nun:

$$AV = VH + fe_k^T, \tag{3.6.15}$$

wobei $f := \gamma p_B(A)$ für ein Skalar γ und das charakteristische Polynom p_B von $B := W^T AW$. Aus der Galerkin-Bedingung folgt des weiteren noch, dass $V^T f = 0$ gilt.

Diese Beobachtungen zeigen, dass wenn es möglich ist einen Vektor v_1 als Linearkombination aus Eigenvektoren von A zu erhalten, gleichzeitig $f = 0$ gilt und V eine orthonormale Basis eines invarianten Unterraums zu A bildet. Die Ritz-Werte $\sigma(H) \subset \sigma(A)$ und die entsprechenden Ritz-Vektoren bilden dann Eigenpaare von A (siehe [15, Kap. 3]).

Definition: Die Arnoldi-Faktorisierung. *Sei $A \in \mathbb{C}^{n \times n}$, dann heißt*

$$AV_k = V_k H_k + f_k e_k^T \tag{3.6.16}$$

eine Arnoldi-Faktorisierung *von A, falls für $V_k \in \mathbb{C}^{n \times k}$ gilt, $V_k^T f_k = 0$ und $H_k \in \mathbb{C}^{k \times k}$ eine obere Hessenberg-Matrix ist mit nicht-negativen Einträgen unterhalb der Diagonale.*
Falls A hermit ist, ist H_k reell, symmetrisch und tridiagonal und die Beziehung wird Lanczos-Faktorisierung genannt (vgl. [15, Kap. 4]).

Mit Hilfe dieser Faktorisierung und den vorausgegangenen Betrachtungen erhält man nun die Ritz-Paare aus den Eigenpaaren der viel handlicheren Matrix H. Falls $H_k y = y\theta$ gilt, dann erfüllt der Vektor $x = V_k y$ für $\beta_k = \|f_k\|$ folgende Gleichung:

$$\|Ax - x\theta\| = \|(AV_k - V_k H_k)y\| = \left|\beta_k e_k^T y\right| \tag{3.6.17}$$

Algorithmus 3 : Arnoldi Faktorisierung (aus [15], Kap. 4)

Data : (A, v)
Setze $v_1 = v/\|v\|$, $w = Av_1$, $\alpha_1 = v_1^T w$
Setze $f_1 = w - v_1\alpha_1$, $V = (v_1)$, $H = (\alpha_1)$
for $j = 1, 2, \ldots, k-1$ **do**

$$\begin{array}{lcll} \beta_j & = & \|f_j\|, & v_{j+1} = f_j/\beta_j \\ V_{j+1} & = & (V_j, v_{j+1}), & \hat{H}_j = \begin{pmatrix} H_j \\ \beta_j e_j^T \end{pmatrix} \\ z & = & Av_{j+1} & \\ h & = & V_{j+1}^T z, & f_{j+1} = z - v_{j+1}h \\ H_{j+1} & = & (\hat{H}_j, h) & \end{array}$$

end

Die Spalten von V bilden, wenn man von exakter Arithmetik ausgeht, eine orthonormale Basis des Krylov-Unterraumes und die Matrix H stellt eine orthogonale Projektion von A auf diesen Unterraum dar. Wenn man mit endlicher Präzision arbeitet, kann man die Orthonormalität der Vektoren trotzdem durch eine Korrektur nach Schritt 4 der for-Schleife sicherstellen.

Die Informationen die man aus diesem Algorithmus erhält, hängen wesentlich von der Wahl des Startvektors ab. Dabei kann es vorkommen, dass die gesuchten Eigenpaare erst bei großem k auftreten. Dann kann aber die komplette Orthogonalität meist nicht mehr gewährleistet werden, was jedoch dazu führen kann, dass so genannte Spur-Eigenwerte berechnet werden, die keine Eigenwerte des ursprünglichen Problems mehr sind. Daher versucht man k auf einer moderaten Größe fest zu halten und dann den Startvektor anzupassen.

3.6.5 Neustarten der Arnoldi Methode

Bei Neustarten des Arnoldi-Prozesses beginnt man die Iteration mit einem Startvektor, der so berechnet wird, dass er näher am k-dimensionalen invarianten Unterraum von Interesse liegt. Das bedeutet, man versucht die ungewollten Anteile bei der Expansion der Eigenvektoren abzuschneiden. Der Algorithmus sieht

dann wie folgt aus.

Algorithmus 4 : Explicit Restarted Arnoldi Method (aus [15], Kap. 5)

Data : (A, v)
Setze $v_1 = v/\|v\|$
for $j = 1, 2, \ldots$ **do**

1. Berechne eine m-Schritt Arnoldi-Faktorisierung
   $Av_m = V_m H_m + f_m e_m^T$ mit $V_m e_1 = v_1$
2. Berechne $\sigma(H_m)$ und entsprechende Ritz-Vektoren und
   stoppe falls die gesuchten Eigenwerte gut angenähert sind
3. Konstruiere ein Polynom Ψ mit Hilfe von $\sigma(H_m)$
   um ungewollte Komponenten abzuschneiden
4. $v_1 = \Psi(A)v_1$, $v_1 = v_1/\|v_1\|$

end

Details zur Konstruktion der Polynome Ψ sollen hier nicht weiter betrachtet werden und auf andere Arbeiten verwiesen werden (vgl. [15]).
Der Hintergrund bei obiger Methode besteht darin, dass für einen Vektor v_1 als Linearkombination von exakt k Eigenvektoren von A der Arnoldi Algorithmus nach k Schritten mit V_k eine orthonormale Basis des A-invarianten Krylov-Unterraumes bildet. Die Ritz-Werte $\sigma(H_k)$ stellen dann die entsprechenden Eigenwerte von A dar. Der Startvektor v_1 wird dabei genau so angepasst, dass die Komponenten des Vektors in Richtung der „gewollten“ Eigenvektoren erhalten bleiben und die „ungewollten“ Komponenten abgeschnitten werden.

Die Implementierung dieser Methode in *Arpack* weist noch einige technische Rafinessen auf, um möglichst viele Eigenwertprobleme lösen zu können. Insbesondere die Struktur der zugrunde liegenden Matrizen wird dabei berücksichtigt und man hat somit ein mächtiges Softwarepaket für die Berechnung mehrerer Eigenwerte und Eigenvektoren von *large scale* Systemen zu Verfügung. Der Aufruf in der Modalanalyse wird in Kapitel 3.4 zur Implementierung der Methode beschrieben und gerade die Genauigkeit der Approximation hat einen großen Beitrag zur Güte der Simulation der Schwingungen.

Kapitel 4

Schwingende Saite

Im folgenden Kapitel soll nun als erstes Beispiel für die Berechnung von Schwingungen die Vibration einer auf beiden Seiten eingespannten Saite untersucht werden. Unter Saite versteht man ein vorgespanntes fadenförmiges, elastisches Kontinuum ohne Biegesteifigkeit (vgl. [14]). Ausgehend von analytischen Betrachtungen der grundlegenden Wellengleichung wird dann die Simulation mit Hilfe der Modalanalyse beschrieben und schließlich die Ergebnisse verglichen. Dabei lassen sich interessante Schlüsse über die Güte des Verfahrens bei unterschiedlichen Diskretisierungen und verschiedenen Lösungsansätzen ziehen.

4.1 Herleitung der Bewegungsgleichung

Die Bewegungsgleichung einer schwingenden Saite oder eines dünnen vorgespannten Seils kann man sich aus dem Newtonschen Grundgesetz der Dynamik angewendet auf die Querbewegung eines Saitenstücks der Länge Δx herleiten. Unter den Voraussetzungen für das physikalische Modell aus 2.1.1 treten neben einer geringen Queraulenkung $u(x,t)$ im Vergleich zur Länge der Saite ebenso nur kleine Neigungswinkel $\alpha(x,t)$ auf. Daher kann man für die Ableitung $\partial u/\partial x = u' = \tan\alpha \approx \alpha$ folgern. Die Beziehung $F = ma = m\ddot{x}$ und das Kräftegleichgewicht in z-Richtung führen dann bei einer Spannkraft $S(x)$ und der Masse pro Länge $\mu(x) = \rho A(x)$ mit $\sin\alpha \approx \alpha$ zu

$$\mu(x)\Delta x\frac{\partial^2 u}{\partial t^2} = S(x+\Delta x)\ \alpha(x+\Delta x,t) - S(x)\ \alpha(x,t). \tag{4.1.1}$$

Durch Einsetzen von $\alpha(x,t) \approx u'(x,t)$ und die Division durch Δx kann man den Grenzübergang $\Delta x \to 0$ durchführen und erhält

$$\mu(x)\ddot{u}(x,t) = [S(x)u'(x,t)]'. \tag{4.1.2}$$

Da die Spannkraft als sehr groß angenommen wird (vgl. 2.1.1), kann diese auch bei Aufbringen einer Streckenlast als konstant angenommen werden. Damit ergibt

sich bei konstantem Querschnitt A mit

$$c^2 = \frac{S}{\mu} = \frac{S}{\rho A} \tag{4.1.3}$$

die Bewegungsgleichung

$$\ddot{u}(x,t) = c^2 u''(x,t). \tag{4.1.4}$$

Diese Gleichung dient nun als Grundlage für alle Bewegungen der schwingenden Saite und soll zunächst analytisch gelöst werden.

4.2 Analytische Lösung

Die homogene Bewegungsgleichung einer schwingenden Saite wurde nun über das Newtonsche Grundgesetz hergeleitet und in einem System ohne Streckenlast, was gleichbedeutend mit einer konstanten Spannkraft ist, erhält man eine eindimensionale Wellengleichung

$$c^2 \, \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}. \tag{4.2.1}$$

Um diese Differentialgleichung 2. Ordnung eindeutig Lösen zu können, muss man noch Randbedingungen an den Enden und Anfangsbedingungen für die Auslenkung festlegen. Das Koordinatensystem des Kontinuums wird dabei der Einfachheit halber mit der x-Achse entlang der Saite ohne Querauslenkung gewählt, was für die Enden $x = 0$ und $x = L$ für die Länge L der Saite zu folgenden Bedingungen führt:

$$u(0,t) = u(L,t) = 0. \tag{4.2.2}$$

Um den Zustand der Saite und Anregungen zum Zeitpunkt $t = 0$ festzulegen, werden noch die Funktionen $A(x)$ für die Querauslenkung u und $B(x)$ für die Ableitung nach der Zeit $\dot{u}$ definiert.

$$u(x,0) = A(x)\,; \qquad \frac{\partial u(x,0)}{\partial t} = B(x). \tag{4.2.3}$$

Nun kann man die Bewegungsgleichung für die schwingende Saite lösen, indem man sich eines Separationsansatzes bedient. Dazu spaltet man die Funktion $u(x,t)$ auf in ein Produkt zweier separater Funktionen f und g, die entweder nur vom Ort x oder der Zeit t abhängig sind. Damit trennt man die Ableitungen der Differentialgleichung auf jeweils eine der Funktionen auf.

$$u(x,t) = f(x)g(t), \tag{4.2.4}$$

$$f(x)\ \ddot{g}(t) = c^2\ f''(x)\ g(t). \tag{4.2.5}$$

Durch Umstellen von f auf die linke und g auf die rechte Seite der Gleichung sind beide Seiten entweder von x oder von t abhängig. Dies ist jedoch nur möglich,

wenn beide einem konstanten, von Ort und Zeit unabhängigem Wert $-\omega^2$ entsprechen.

$$\frac{\ddot{g}(t)}{g(t)} = c^2 \, \frac{f''(x)}{f(x)} = -\omega^2. \tag{4.2.6}$$

Die Definition eines negativen Wertes ω^2 hat dabei physikalische Gründe, da bei positiver Querauslenkung immer eine rückführende Kraft in Richtung der Ruhelage entsteht. Nun spaltet man diese Gleichung in zwei Teilgleichungen auf, deren allgemeine Lösung bekannt ist.

$$\ddot{g}(t) + \omega^2 g(t) = 0, \tag{4.2.7}$$

$$f''(x) + \left(\frac{\omega}{c}\right)^2 f(x) = 0. \tag{4.2.8}$$

Da die Lösung für den Wert $\omega = 0$ nicht relevant ist, soll im Folgenden allgemein $\omega \neq 0$ vorausgesetzt werden. Die allgemeinen Lösungen für diese Gleichungen lauten dann

$$f(x) = a \cos \frac{\omega}{c} x + b \sin \frac{\omega}{c} x, \tag{4.2.9}$$

$$g(t) = A \cos \omega t + B \sin \omega t. \tag{4.2.10}$$

Die Konstanten a, b können nun aus den Randbedingungen 4.2.2 bestimmt werden. Aus der Bedingung $u(0,t) = f(0) = 0$ ergibt sich direkt $a = 0$. Für b erhält man dann aus $u(t,L) = f(L) = 0$ die Gleichung

$$b \sin \frac{\omega}{c} L = 0. \tag{4.2.11}$$

Außer der trivialen Lösung mit $b = 0$ ergibt sich daraus die charakteristische Gleichung oder *Frequenzgleichung* zur Bestimmung der Eigenfrequenzen ω.

$$\sin \frac{\omega}{c} L = 0. \tag{4.2.12}$$

Bekanntlich besitzt die Sinusfunktion unendlich viele Nullstellen, was dazu führt, dass bei derartigen Schwingungen auch unendlich viele Eigenkreisfrequenzen auftreten. Diese ergeben sich mit $\frac{\omega}{c} L = k\pi$ zu

$$\omega_k = k\pi c/L, \qquad k = 1, 2, \ldots \; . \tag{4.2.13}$$

Damit ergibt sich zu jeder Eigenkreisfrequenz ω_k eine Eigenfunktion $f_k(x)$,

$$f_k(x) = b_k \sin \frac{\omega_k}{c} x = b_k \sin k\pi \frac{x}{L}. \tag{4.2.14}$$

Diese charakteristischen Eigenfunktionen werden in späteren Simulationen durch die Eigenvektoren der Systemmatrizen approximiert. Genau wie bei den Eigenvektoren kann man auch hier die Funktionen $f_k(x)$ nur bis auf eine multiplikative

Konstante bestimmen. Um das Gleichungssystem zu vereinfachen wird diese Konstante über die Orthogonalitätseigenschaft der Eigenfunktionen bestimmt.

$$\int_0^L f_i(x) f_j(x) dx = \begin{cases} L_i^* & \text{für } i = j, \\ 0 & \text{für } i \neq j. \end{cases} \tag{4.2.15}$$

Der genaue Wert von L_i^* wird vom Eigenwertlöser selbst bestimmt, um die Orthonormalität bezüglich der Systemmatizen zu gewährleisten und die Bewegungsgleichung in eine einheitliche Form zu bringen (vgl. Kapitel 3.1 und 3.6.1). Für die Berechnungen hier soll der Wert $b_k = 1,\ k = 1, 2, \ldots$ angenommen werden, wobei sich eine Orthogonalitätskonstante $L_k^* = L/2$ ergibt. Dieser Wert kann so gewählt werden, dass er für alle Eigenfunktionen konstant bleibt.
Die zu einer einzelnen Eigenkreisfrequenz ω_k gehörende Einzellösung wird als Eigenschwingung $u_k(x,t)$ bezeichnet. Diese ergibt sich durch Einsetzen der Eigenkreisfrequenz in die Teillösungen und durch Multiplikation der zuvor separierten Funktionen.

$$u_k(x,t) = f_k(x) g_k(t) = \sin k\pi \frac{x}{L} \left(A_k \cos \omega_k\ t + B_k \sin \omega_k\ t \right). \tag{4.2.16}$$

Genau wie bei der Überlagerung der Moden in der Modalanalyse (Kapitel 3.1) erhält man hier die allgemeine Lösung durch Aufsummieren der Eigenschwingungen, wobei nun unendlich viele Summanden auftreten.

$$\begin{aligned} u(x,t) &= \sum_{k=1}^{\infty} u_k(x,t) \\ &= \sum_{k=1}^{\infty} \sin k\pi \frac{x}{L} \left(A_k \cos \omega_k t + B_k \sin \omega_k t \right) \\ &= \sum_{k=1}^{\infty} \sin(k\pi \frac{x}{L}) \cdot C_k \cos(\omega_k t - \varphi_k). \end{aligned} \tag{4.2.17}$$

Letztendlich müssen noch die Konstanten für die Anfangsauslenkungen A_k und Anregungen B_k für jede Eigenschwingung aus den Anfangsbedingungen $u(x, t = 0) = u_0(x)$ und $\dot{u}(x, t = 0) = \dot{u}_0(x)$ bestimmt werden. Aus (4.2.17) erhält man für $t = 0$

$$u_0(x) = \sum_{k=1}^{\infty} A_k \sin k\pi \frac{x}{L}, \quad \dot{u}_0(x) = \sum_{k=1}^{\infty} \omega_k B_k \sin k\pi \frac{x}{L}. \tag{4.2.18}$$

Hier nutzt man nun die Orthogonalität der Eigenformen und multipliziert die Gleichungen mit $f_j(x) = \sin j\pi \frac{x}{L}$. Durch Integration von $x = 0$ bis $x = L$ fallen alle Therme für $j \neq k$ aus der Summe weg und es bleibt nur die Konstante $L_k^* =$

$L/2$ für $j = k$ übrig. Damit ergeben sich die Faktoren für die Anfangsbedingungen zu

$$A_k = \frac{2}{L}\int_0^L u_0(x)\sin k\pi\frac{x}{L}dx, \quad k = 1, 2, \dots , \tag{4.2.19}$$

$$B_k = \frac{2}{\omega_k L}\int_0^L \dot{u}_0(x)\sin k\pi\frac{x}{L}dx, \quad k = 1, 2, \dots . \tag{4.2.20}$$

Die Faktoren für die Amplituden C_k und die Phasenverschiebungen φ_k in (4.2.17) sind die gleichen wie bei der Modalanalyse und werden analog aus A_k und B_k berechnet.

$$C_k = \sqrt{{A_k}^2 + {B_k}^2}, \tag{4.2.21}$$

$$\varphi_k = \tan^{-1}\left(\frac{B_k}{A_k}\right). \tag{4.2.22}$$

Tritt eine Streckenlast F nach Gleichung 3.1.1 auf, so kann man diese für $t \geq 0$ in eine neue Ruhelage x^0 umrechnen.

$$x^{Gl} = x^0 = \frac{F}{\omega^2} \quad \text{(vgl. 3.2.7)}. \tag{4.2.23}$$

Der Gewichtsfaktor für die einzelnen Eigenfunktionen aus 4.2.17 berechnet sich dann aus

$$x_k = x_k^0 + C_k\cos(\omega_k t - \varphi_k). \tag{4.2.24}$$

Bei einer Impulsanregung muss man diese wie eine Geschwindigkeitsänderung der Struktur in der Ruhelage betrachten. Aus der transformierten Impulskraft r_k berechnet man sich den Wert für die Geschwindigkeit v_k^0 durch die Division mit der entsprechenden Eigenfrequenz. Dieser Wert kann dann in 4.2.24 direkt als Amplitude C_k für die Eigenschwingungen genutzt werden.

4.3 Herleitung der Schwachen Formulierung

Gegeben sei die Wellengleichung:

$$\frac{\partial^2 u}{\partial t^2} = \nabla[c^2\nabla u], \quad x \in \Omega, t > 0, \tag{4.3.1}$$

$$u(x, 0) = f(x), \quad x \in \Omega, \tag{4.3.2}$$

$$\frac{\partial}{\partial t}u(x, 0) = 0, \quad x \in \Omega, \tag{4.3.3}$$

$$u(x, t) = 0, \quad x \in \partial\Omega, t \geq 0. \tag{4.3.4}$$

Durch Multiplikation mit den Basisfunktionen und Integration der Gleichung (4.3.1) kann man nun die schwache Formulierung erhalten:

$$\int_\Omega \frac{\partial^2 \hat{u}}{\partial t^2} N_i \, d\Omega = \int_\Omega \nabla[c^2 \nabla \hat{u}] N_i \, d\Omega. \tag{4.3.5}$$

Nun wird die rechte Seite partiell integriert um die Randbedingungen für die Ableitungen zu erfassen. Durch Einsetzen des FEM-Ansatzes für $\hat{u}$ bekommt man schließlich eine diskrete Bewegungsgleichung.

$$\text{(FEM)} \qquad u \approx \hat{u} = \sum_{j=1}^{M} u_j N_j, \tag{4.3.6}$$

$$\int_\Omega \frac{\partial^2 \hat{u}}{\partial t^2} N_i \, d\Omega = \hat{u}'(l) - \hat{u}'(0) - \int_\Omega c^2 \nabla \hat{u} \nabla N_i \, d\Omega, \tag{4.3.7}$$

$$\sum_{j=1}^{M} \int_\Omega N_i N_j \frac{\partial^2 u_j}{\partial t^2} \, d\Omega = -\sum_{j=1}^{M} \int_\Omega c^2 \nabla N_i \nabla N_j u_j \, d\Omega, \tag{4.3.8}$$

$$\sum_{j=1}^{M} M_{i,j} \, \ddot{u}_j = -\sum_{j=1}^{M} K_{i,j} \, u_j. \tag{4.3.9}$$

Zusammengefasst und in Matrix-Schreibweise ergibt das die im weiteren angenommene Form der Bewegungsgleichung im ungedämpften Fall:

$$M \, \ddot{u} + K \, u = 0. \tag{4.3.10}$$

Damit hat man eine schwache Formulierung für die Massenmatrix M und die Steifigkeitsmatrix K gefunden:

$$M = \int_\Omega N_i N_j \, d\Omega, \tag{4.3.11}$$

$$K = \int_\Omega c^2 \nabla N_i \nabla N_j \, d\Omega. \tag{4.3.12}$$

Durch hinzufügen einer äußeren Last F wird aus der homogenen Gleichung (4.3.10) schließlich noch die inhomogene:

$$M \, \ddot{u} + K \, u = F. \tag{4.3.13}$$

Die noch verbleibenden Randbedingungen (4.3.4) finden sich in äußeren Einträgen der Matrizen M und K wieder, was genauer bei der Konstruktion aus den Elementmatrizen klar wird (vgl. Kapitel 4.4).

4.4 Implementierung in Diffpack

Im Folgenden soll nun eine Simulation der Schwingungen durch die Implementierung in Diffpack und die Anwendung der Modalanalyse beschrieben werden. Zunächst wird die Klasse des dazu benötigten Simulators (vgl. Kapitel 2.4) Schritt für Schritt aufgebaut und die verwendeten Methoden beschrieben.

Für die Klasse des Simulators `Wave1` wurden die Header-Datei `Wave1.h` und die entsprechende C++-Datei `Wave1.cpp` angelegt. Alle verwendeten Variablen und Methoden sind zum besseren Verständnis auch im Quellcode dokumentiert und können bei Bedarf angepasst werden. Die Ausführungen in diesem Kapitel sollen einen Einblick in die Funktionsweise der Methoden geben und den Ablauf der Simulation verdeutlichen und bilden keine vollständige Dokumentation der Klasse. Der Aufruf der Funktionen ist im Kapitel zur Programmierung mit Diffpack beschrieben (vgl. Kapitel 2.4).

```
Wave1::adm( MenuSystem& menu )
```

Zur Administration der Systemvariablen wird die Funktion `adm` genutzt, die zunächst das `MenuSystem menu` mit der benutzerdefinierten Simulation `SimCase` durch den Befehl `attach(menu)` verknüpft. Im Anschluss werden die Funktionen `define`, `menu.prompt` und `scan` aufgerufen, um die globalen Variablen zu definieren und aus dem *Inputfile* einzulesen.

```
Wave1::define( MenuSystem& menu, int level )
```

Dem Menü werden nun durch die Methode `addItem` Werte hinzugefügt. Für die schwingende Saite ist hier im Wesentlichen das *Gridfile* mit der Anzahl und dem Typ der verwendeten Elemente definiert. Zudem werden die Werte für die Spannkraft S und die Dichte ρ, sowie für den Querschnitt A der Saite aus Gleichung 4.1.2 hier anlegt, um diese bei Bedarf individuell angeben zu können. Die Standardwerte sind in Tabelle 2.1 aufgeführt.
Hier werden nun auch alle anderen benötigten `defineStatic`-Methoden anderer Klassen aufgerufen. In diesem Fall sind das die Klassen `SaveSimRes` zum Setzten der Parameter für das Speichern der Resultate, `LinEqAdmFE` für Änderungen im Löser des linearen Gleichungssystems und `FEM` um die Finite Elemente Methode selbst anzupassen. Schließlich wird hier auch die im Kapitel der Modalanalyse (vgl. Kapitel 3.4) beschriebene Methode `defineStatic` zum Festlegen aller Parameter für die Simulation der Schwingungen aufgerufen.

```
SaveSimRes::defineStatic (menu , level +1);
LinEqAdmFE::defineStatic (menu , level +1);
```

```
FEM::defineStatic (menu , level+1);

ModalAnalysis::defineStatic(menu, level);
```

Das `level` wird dabei um einen Zähler erhöht, um ein Untermenü für jedes aufgerufene Menü zu schaffen. Die Klasse `ModalAnalysis` stellt dieses Untermenü auch ohne das Erhöhen des Levels zur Verfügung. Details zur Verwendung dieser Ebenen werden bei der Ausführung der Simulation beschrieben.

`Wave1::scan()`

Alle in der Funktion `define` angelegten Parameter werden nun aus einem *Inputfile* oder über direkte Eingaben im Menü eingelesen. Die für die Wellengeschwindigkeit ausschlaggebenden Parameter nach Gleichung 4.1.3 sind

- tension force (Spannkraft S)
- cross section (Querschnitt A)
- rho (Dichte ρ)

Nicht gesetzte Variablen erhalten die Standardwerte, die in der Funktion `addItem` angelegt wurden. Nun werden alle global definierten `Handle` durch die Methode `rebind` und die globalen Variablen mit den eingelesenen Werten initialisiert. Insbesondere wird hier die entsprechende `scan`-Funktion für die Modalanalyse angesprochen:

```
ModalAnalysis::scan( menu, dof ) ;
```

Zur späteren Beurteilung der Ergebnisse der Simulation werden hier auch die Werte zur Berechnung einer analytischen Lösung gesetzt, welche durch eine eigene Klasse `El1AnalSol` repräsentiert wird. Die Initialisierung der Anfangsbedingungen wird dann in den folgenden Methoden geschehen.

`Wave1::setIC()`

Um die Anfangsauslenkung der Saite setzen zu können, wird wie bei der analytischen Lösung eine eigene Klasse `Wave1IC` genutzt. Diese wird abgeleitet von der Diffpack-Klasse `FieldFunk` und besitzt die hier verwendete Funktion `valuePt(const Ptv(dpreal)&x, dpreal t)`, um jedem Punktvektor `x` des Gitters einen Wert zu Zeit `t` zuzuweisen. Der Aufruf in der Funktion `setIC` beschränkt sich dann auf `u_init->fill( *uinit_func, 0.0)`, um das Anfangs-Feld `u_init` zur Zeit $t = 0$ mit den Werten der Funktion zu belegen. Die Werte für diese Anfangsauslenkung können anhand der Gleichung 4.2.19 für A_k bestimmt werden oder unter Berücksichtigung der Randwerte beliebig mit Werten belegt werden.

`Wave1::fillEssBC()`

Alle essenziellen Randbedingungen für die Saite (siehe Gleichung 4.2.2) werden im *Gridfile* beschrieben und müssen noch auf das Objekt `dof` übertragen werden, das die Freiheitsgrade des Systems repräsentiert. Dies geschieht durch `fillEssBC`:

```
dof->initEssBC () ;
const int nno = grid->getNoNodes() ;

for (int i = 1; i <= nno; i++)
{
   // is node i subjected to boundary indicator 1?
   if (grid->boNode( i , 1 )
   {
      dof->fillEssBC( i , 0.0 ) ;
   }

   if( grid->boNode( i , 2 ) )
   {
      vSource( i ) = dSource ;
   }
   if( grid->boNode( i , 3 ) )
   {
      vImpulse( i ) = dImpulse ;
   }
}

dof->vec2field( vSource , *source ) ;
dof->vec2field( vImpulse , *impulse ) ;
```

Dabei wird für jeden Knoten des Gitters abgefragt, ob der Knoten auf dem Rand liegt `grid->boNode(i,1)`. Da hier nur ein Freiheitsgrad pro Knoten notwendig ist, gibt es auch nur den Indikator `1` für die Randbedingung. Für alle so gekennzeichneten Knoten wird nun die Auslenkung durch `dof->fillEssBC` auf `0` gesetzt. Die Last F auf die Saite in Gleichung 4.2.23 wird durch den Indikator `2` angezeigt und der entsprechende Eintrag im Vektor `vSource` auf den im Input-File angegebenen Wert `dSource` gesetzt. Nachdem die Impulskraft analog festgelegt wurde, die dem Wert B aus Gleichung 4.2.20 entspricht, werden die beiden Vektoren noch in Felder geschrieben, die in der Modalanalyse verwendet werden können.

`Wave1::integrands( ElmMatVec &elmat, const FiniteElement &fe )`

Die eigentliche Formulierung der Problemstellung geschieht nun in der `integrands`-Funktion. Hier werden für jedes Element die Systemmatrizen und der

Lösungsvektor aufgrund der schwachen Formulierung aus Kapitel 4.3 bestimmt. Die Referenz auf das Objekt `FiniteElement fe` beinhaltet dabei die Informationen über die Anzahl der Basisfunktionen des aktuellen Finiten Elements, die Dimension, die Jacobi-Determinante und die Werte der Basisfunktionen sowie deren Ableitungen. Um nun die Matrix des Objekts `elmat` mit Werten zu füllen, wird zunächst unterschieden, ob die Steifigkeitsmatrix oder die Massenmatrix berechnet wird. Dazu wird abgefragt, ob der Integrationstyp `integrationType` gleich `STIFFNESSMATRIX` oder `MASSMATRIX` ist. Zunächst der 1. Fall:

```
if (integrationType==STIFFNESSMATRIX)
{
   for (i = 1; i <= nbf ; i++)
   {
      for (j = 1; j <= nbf ; j++)
      {
         nabla2 = m_dC2Wave * fe.dN(i,1) * fe.dN(j,1) ;
         elmat.A(i,j) += nabla2*detJxW;
      }
   }
}
```

Wie man der schwachen Formulierung der Bewegungsgleichung der schwingenden Saite aus Gleichung 4.3.12 entnehmen kann, wird die Elementmatrix hier aus dem Produkt der ersten Ableitungen `fe.dN` der Basisfunktionen gebildet. Der Faktor `m_dC2Wave` entspricht der Wellengeschwindigkeit c^2 und lässt sich aus der Masse pro Element und der Spannkraft der Saite bestimmen (vgl. Gleichung 4.1.3). Die Transformation auf globale Koordinaten wird, wie im Kapitel 2.3.2 beschrieben, durch die Multiplikation mit der Jacobi-Determinante `detJxW` durchgeführt (vgl. Gleichung 2.3.13). Zur numerischen Integration werden die berechneten Werte dann bei jedem Aufruf von `integrands` aufsummiert (siehe Gleichung 2.3.9).
Analog wird auch die Massenmatrix berechnet:

```
else if (integrationType==MASSMATRIX)
{
   for (i = 1; i <= nbf ; i++)
   {
      for (j = 1; j <= nbf ; j++)
      {
         elmat.A(i,j) += fe.N(i)*fe.N(j)*detJxW ;
      }
   }
}
```

Hier wird die Matrix direkt aus den Werten der Basisfunktionen `fe.N` und der Transformation durch die Jacobi-Determinante berechnet, wie in der schwachen

Formulierung beschrieben (vgl. 4.3.11). Somit werden die Matrizen bei dem entsprechenden Wert von `integrationType` durch das Aufsummieren aller Elementmatrizen gebildet. Die Lösung des Problems bzw. die Modalanalyse wird dann in der Funktion `solveProblem` durchgeführt.

`Wave1::solveProblem()`

Da nun alle Methoden zu Initialisierung und Berechnung der Parameter zur Verfügung stehen, kann man in der Funktion `solveProblem` alle notwendigen Funktionen aufrufen und das Problem lösen. Dazu werden zunächst die oben beschriebenen Methoden `fillEssBC` und `setIC` aufgerufen, um die Anfangs- und Randbedingungen zu setzen. Dann wird mit der Steifigkeitsmatrix der erste Teil des Systems erzeugt:

```
integrationType = STIFFNESSMATRIX ;
makeSystem( *dof , *lineq ) ;
```

Die Diffpack-interne Funktion `makeSystem` führt dazu alle notwendigen Schritte aus, um mit Hilfe der `integrands`-Funktion die Matrix im Objekt `lineq` vom Typ `LinEqAdmFE` zu speichern. Um diese schließlich in eine separate Matrix `K` zu kopieren, muss man durch `makeItSimilar` die Parameter anpassen und die Werte aus `lineq->A` übertragen.

```
lineq->A().makeItSimilar( K ) ;
*K = lineq->A() ;
```

Die gleichen Schritte sind nun notwendig, um die Massenmatrix zu erzeugen. Da beide Matrizen die gleichen Parameter besitzen, kann man auch zuerst `makeItSimilar` aufrufen und der Funktion `makeSystem` gleich die Referenz auf die angepasste Matrix `M` übergeben.

```
integrationType = MASSMATRIX;
lineq->A().makeItSimilar( M );
makeSystem(*dof,M.getRef());
```

Nun hat man mit `K` und `M` die beiden Systemmatrizen zur Verfügung und kann die Modalanalyse starten. Je nach Bedarf setzt man zunächst die Felder für Anfangsauslenkung `u_init`, für die Last `source` und den Impuls `impulse`. Danach kann man die charakteristischen Eigenfrequenzen und Eigenfunktionen mit Hilfe des Eigenwertlösers berechnen.

```
setInitialField( *u_init ) ;
setSource( *source ) ;
setImpulse( *impulse ) ;

calcEV( *K, *M ) ;
```

Im Anschluss muss die Funktion `solveProblem` der Modalanalyse selbst aufgerufen werden, um mit Hilfe der gespeicherten Eigenformen die Schwingung zu berechnen.

```
ModalAnalysis::solveProblem( ) ;
```

Alles Weitere geschieht nun in der Klasse `ModalAnalysis` und wird in den Dateien mit den Resultaten `.simres` und `.field` gespeichert. Das Feld mit den Verschiebungen ist über die Variable `D` vom Typ `Handle(FieldsFE)` verfügbar und kann für weitere Berechnungen genutzt werden. Des weiteren gibt es noch Methoden, um einzelne Parameter der Modalanalyse auch außerhalb der Abgeleiteten Klasse zu erhalten, wie zum Beispiel `GetEigenValues`, welche einen Vektor mit allen berechneten Eigenwerten der Matrizen liefert.
Zum Abschluss wird in der Funktion `solveProblem` durch `calcAnalSol` noch die Berechnung der analytischen Lösung auf der Basis der Ergebnisse aus 4.2 gestartet, um die Genauigkeit der Simulierten Schwingung zu überprüfen.

```
Wave1::calcAnalSol( void )
```

In einfacheren Beispielen, wie hier der Bewegung einer schwingenden Saite, ist es möglich auch eine analytische Lösung der Differenzialgleichung zu berechnen. Der Lösungsweg dazu ist in Kapitel 4.2 beschrieben. Wie in den Ausführungen zur Methode `scan` angedeutet, wurde hier eine eigene Klasse angelegt, um die analytischen Eigenfrequenzen und Eigenvektoren zu speichern. Diese werden nun in einer ebenfalls vorgegebenen Anzahl wie in der Modalanalyse überlagert und die Ergebnisse verglichen. Bei Bedarf kann man hier die Abweichungen sowohl der approximierten Eigenformen und Frequenzen, als auch den Fehler in der Schwingungssimulation ausgeben. Zur graphischen Veranschaulichung wurde auch die Bewegung eines einzelnen Knotens im Vergleich zur Modalanalyse geplottet, wie in Abbildung 4.1 zu sehen ist.

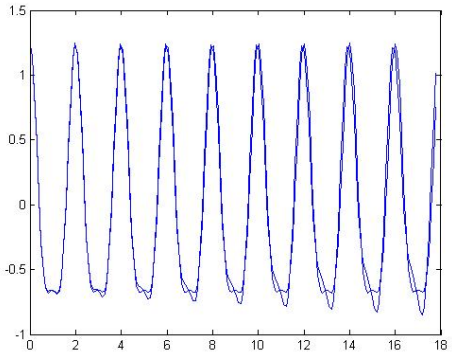

Abbildung 4.1: Schwingung eines Knotens, Simulation & analytische Lösung

4.5 Berechnung der Schwingungen

4.5.1 Programmaufruf

Die Berechnung eines konkreten Beispiels erfolgt in *Diffpack* üblicherweise durch den Aufruf der erstellten Applikation in einer Konsole oder einem anderen Programm. Dabei können die Eingabeparameter entweder nach dem Aufruf gesetzt werden, oder einfacher zuvor in einem *InputFile* gespeichert werden. Dort werden dann die Variablen sowohl für die Saite, also auch das Gitter, als auch für die Modalanalyse und den Eigenwertlöser festgelegt. Ein einfaches Beispiel könnte wie folgt aussehen:

```
set gridfile = String1.grid

set impulse = -0.01
set analytical Modes = 15

sub EigenSolver
   set EigenSolver cut_bound = true
ok

sub ModalAnalysis
   set ModalAnalysis noModes = 5
   set ModalAnalysis dampFactor = 0.1
   set ModalAnalysis time parameters = dt=0.05 t in [0,10]
ok
ok
```

Hier ist die Struktur des Diffpack-Menüs mit den verschiedenen Ebenen zu erkennen. Für sie Saite werden alle Parameter über das Schlüsselwort `set` gesetzt,

die Modalanalyse und der Eigenwertlöser besitzen jeweils ein Untermenü welches über `sub` erreicht wird. Befindet man sich dann in der untergeordneten Ebene, muss man noch das Prefix der entsprechenden Klasse hinzufügen, um die Werte eindeutig zu beschreiben. Das Untermenü wird dann mit `ok` wieder geschlossen. In dem aufgerufenen `gridfile` befindet sich die Definition der Saite, bzw. der Knoten und Elemente, die verwendet werden sollen. Es ist auch möglich verschiedene Materialien für die Elemente zu verwenden, was hier jedoch nicht betrachtet wurde. Nachdem festgelegt wurde, wie viele Dimensionen und Knoten, bzw. Elemente die Struktur besitzt, werden die Koordinaten und Randbedingungen für alle Knoten gesetzt. In diesem Beispiel gehören zu den so genannten `boundary indicators` zunächst der Indikator, ob ein Knoten auf dem Rand liegt. Weitere Indikatoren werden für Knoten festgelegt, auf die eine angegebene Last `source` oder der Impuls `impulse` wirkt. Zuletzt müssen noch für jedes Element der Typ, hier `ElmB2n1D`, der Indikator für das Material und die zugehörigen Knoten angegeben werden. Ein Beispiel für ein solches Gitter befindet sich auf der beigelegten CD.

Nachdem alle Parameter durch das *InputFile* und die räumliche Diskretisierung der Struktur durch das `gridfile` gegeben sind, wird die Schwingungssimulation berechnet. Dabei werden ein Reihe von Ausgabedateien erzeugt:

- Diffpack Dateien
  - SIMULATION.files
    Auflistung und Beschreibung der erzeugten Dateien.
  - SIMULATION.dp
    log_file mit Programm-Name, CPU-Zeit, etc.
  - .SIMULATION.field
    Ausgabedatei für alle erzeugten Felder.
  - .SIMULATION.grid
    Information über das Gitter und die Elemente.
  - .SIMULATION.simres
    Übersicht über alle Felder mit Zeitschritten.
- Eigenwertlöser
  - .SIMULATION_EV_i
    Knotenwerte der einzelnen Eigenvektoren.
  - EV.txt
    Übersicht über alle Eigenwerte und Eigenvektoren.
  - (.SIMULATION.simres)
    Speicherung des Feldes der Eigenvektoren.

- Modalanalyse
  - .Node_plot_1
    Verschiebungen eines einzelnen Knotens für alle Zeitschritte.
  - .SIMULATION_1
    Analytische Verschiebungen des Knotens für alle Zeitschritte.
  - (.SIMULATION.simres)
    Speicherung des Feldes der Knotenverschiebungen.

Der Standardname „SIMULATION“ kann dabei durch das Setzen des `casename` verändert werden. Die Ergebnisse der Simulation befinden sich im Wesentlichen in der Datei mit der Endung `field`. Alle in der Berechnung verwendeten Felder, wie zum Beispiel `u_init` für die Anfangsauslenkung oder `D` für die Knotenverschiebungen, können hier gespeichert werden.
Gerade bei Schwingungen sind graphische Darstellungen und Videos sehr hilfreich, um die Ergebnisse zu analysieren. In *Diffpack* wird dazu oft das Format *vtk* genutzt, das von zahlreichen Programmen wie zum Beispiel *ParaView* visualisiert werden kann. Zur Umwandlung des Gitters und der Verschiebungen wird hier das Diffpack-Skript `simres2vtk` benutzt, das für beliebige Felder eine entsprechende vtk-Datei erzeugt. Der Aufruf sieht dann wie folgt aus:

```
simres2vtk -s -n D_1 -f SIMULATION -a -A
```

Die Beschreibung der Parameter kann den Ausführungen und der Hilfe im Skript entnommen werden. Durch das Anhängen von `-A` an den Aufruf werden alle Felder mit der Bezeichnung `D_1` berücksichtigt und die Verschiebungen zu jedem Zeitschritt können dargestellt werden.

4.5.2 Beispiel: Last auf Saitenmitte

Im ersten Beispiel wird die Querschwingung einer Saite mit 30 Elementen bei Aufbringen einer Last auf den mittleren Knoten betrachtet.

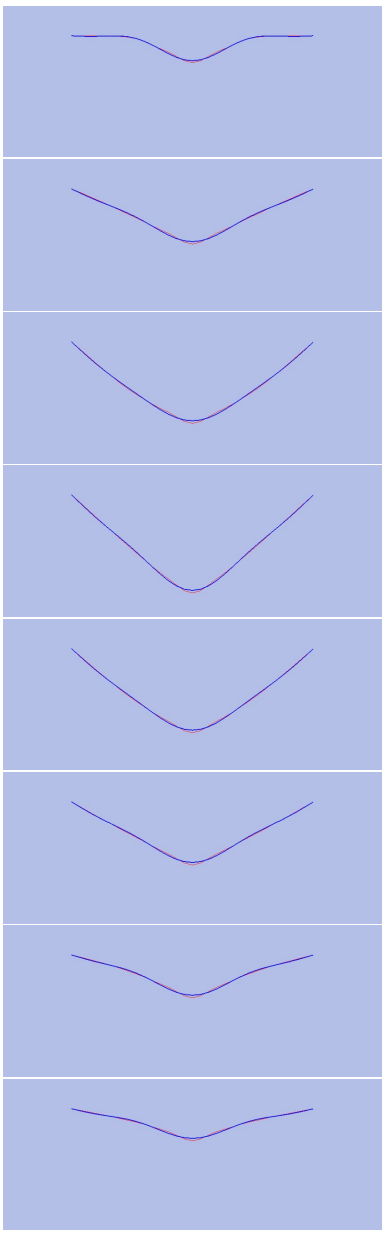

Abbildung 4.2: Schwingende Saite

Durch die Kraft auf den mittleren Knoten, die durch die Last ab dem Zeitpunkt $t = 0$ verursacht wird, beginnt die Saite um einen neue Gleichgewichtslage x_0 (vgl. 4.2.23) zu schwingen. Die Simulation der Knotenverschiebungen wird durch die Überlagerung der ersten 5 Eigenformen, die in Abbildung 4.4 dargestellt sind, berechnet. Die Gewichtsfaktoren für die Gleichgewichtslage ergeben sich nach Gleichung 3.2.7 durch das Skalarprodukt des Lastvektors für die Knoten mit dem jeweiligen Eigenvektor und die Division durch den zugehörigen Eigenwert $\lambda = \omega^2$. Ein positiver Dämpfungsparameter $\xi < 1$ nach 3.2.5 führt zu einer Abschwächung der Schwingungsamplitude und der Konvergenz gegen die berechnete Gleichgewichtslage. Abbildung 4.2 zeigt die Auslenkung der Saite zu den Zeitpunkten $i/8 \cdot T = i/8 \cdot 2\pi/\omega_1$ für $i = 1, \ldots, 8$. Die Position bei 2/8 oder 6/8 der Schwingungsperiode entspricht annähernd der neuen Gleichgewichtslage. Ein Vergleich mit der analytischen Lösung wird durch die dünne Rote Linie in der Abbildung gegeben, welche eine Berechnung der Verschiebung durch Überlagerung der 15 ersten analytischen Eigenformen darstellt. Detaillierter ist der Unterschied in Abbildung 4.3 zu erkennen, in der die Gleichgewichtslage zu sehen ist. Die Besonderheit bei einer Last auf die Mitte der Saite besteht darin, dass nur ungerade Eigenfrequenzen der Schwingung angeregt werden, da alle geraden Eigenvektoren eine Nullstelle am mittleren Knoten besitzen und daher nicht zur Schwingung beitragen. Die Abweichungen von der analytischen Lösung ergeben sich hier zum einen durch die unterschiedliche Anzahl an verwendeten Eigenformen, zum anderen durch die Fehler in der Approximation der Eigenvektoren und Eigenwerte, die jedoch vergleichsweise gering sind. Insbesondere beim ersten Eigenpaar beträgt dieser Fehler für die Eigenfrequenz lediglich etwas weni-

ger als 0.05% und die $L2$-Norm der Eigenvektordifferenz liegt im Bereich $4 \cdot 10^{-6}$. Für die Güte der Ergebnisse ist die Approximation der Eigenvektoren und Eigenwerte durch das Softwarepaket *Arpack* besonders ausschlaggebend. Im obigen Beispiel wurde eine Diskretisierung der Saite durch 31 Konten gewählt, was zu einer ausreichend guten Annäherung der Eigenformen führt, wie in Abbildung (4.4) dargestellt.

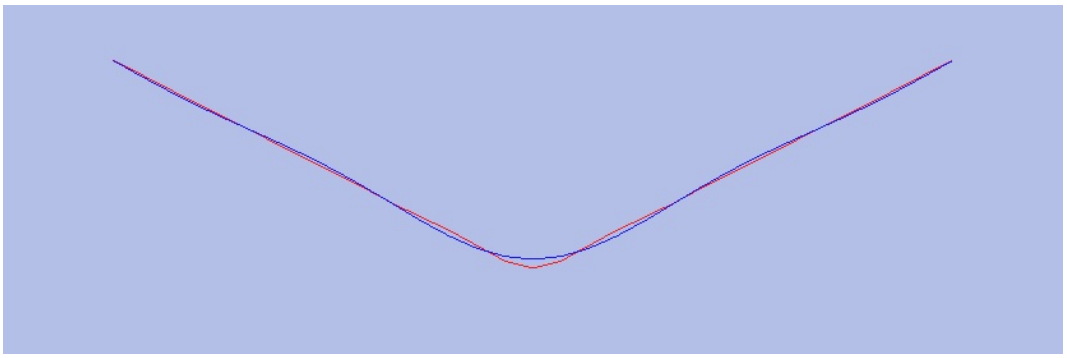

Abbildung 4.3: Saite Gleichgewichtslage

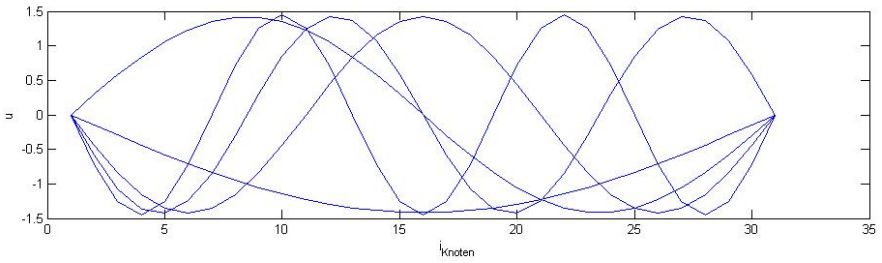

Abbildung 4.4: Eigenvektoren Saite

4.6 Analyse der Ergebnisse

Die Simulation der Schwingungen einer Saite kann mit *Diffpack* und der implementierten Modalanalyse sehr präzise berechnet werden. Dazu trägt auch die vorteilhafte Tridiagonalgestalt der Systemmatrizen bei, die durch die Elemente mit jeweils einem Freiheitsgrad für die beiden Knoten entsteht.
Diese Gestalt wirkt sich auch bei der Eigenwertberechnung durch *Arpack* aus und ermöglicht die schnelle Berechnung von sehr guten Approximationen der

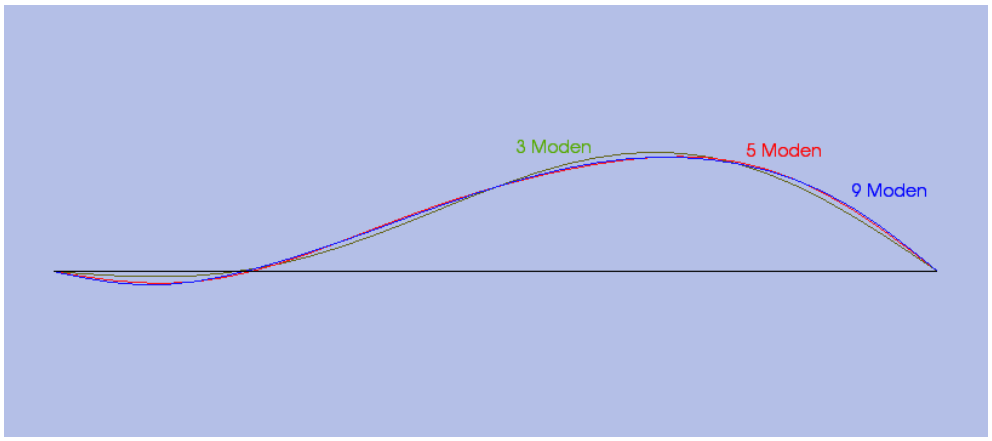

Abbildung 4.5: Vergleich Anzahl der Moden

Eigenformen und Eigenfrequenzen. Eine höhere Genauigkeit der Schwingungssimulation kann bei Bedarf durch ein feineres Gitter und die Verwendung höherer Eigenformen erreicht werden. Jedoch haben die Beispiele gezeigt, dass durch die Überlagerung von 5 Eigenvektoren bereits gute Approximationen der Knotenverschiebungen berechnet werden können (vgl. Abbildung 4.5). Verwendet man noch weniger Eigenformen, so zeigen sich zu Beginn der Schwingung etwas größere Abweichungen, welche jedoch durch die modale Dämpfung mit zunehmender Zeit verschwinden. Dies hängt damit zusammen, dass höhere Eigenfrequenzen stärker gedämpft werden und ab einem bestimmten Zeitpunkt nur noch die erste Eigenschwingung zu beobachten ist.
Betrachtet man die Anregung der Schwingung durch einen Impuls zum Zeitpunkt $t = 0$, so treten zunächst auch wesentlich höhere Eigenfrequenzen auf. Daher ist für eine gute Simulation auch die Berechnung von mehreren Eigenformen notwendig. Ist der spätere Verlauf der Schwingung von größerer Bedeutung, kann dies jedoch ignoriert werden. Zudem verändert sich die Gleichgewichtslage der Saite durch den Impuls nicht und nach Abklingen der Schwingung befindet sich die Saite wieder in der Null-Position. Die Berechnung der Eigenwerte und Eigenvektoren ist jedoch unabhängig von der Anregung und hängt nur von den Systemmatrizen K und M ab.
Durch Setzen einer Anfangsauslenkung kann man auch Abweichungen der Position von der Ruhelage zum Zeitpunkt $t = 0$ festlegen. Dies könnte zum Beispiel durch Aufhebung einer Last oder Ähnliches verursacht werden. Die Simulation zeigt dann das Ausschwingen der Saite bis zum Übergang in die Gleichgewichtslage bei einem genügend großen Zeitintervall. Durch diese Funktion hat man auch die Möglichkeit, mehrere Schwingungsbereiche festzulegen, indem man die Modalanalyse mit der jeweiligen Endauslenkung und einer neuen Anregung nochmals startet.

Kapitel 5

Eingespannt-freier Balken

Die zweite Anwendung der Modalanalyse betrachtet die Schwingung eines eingespannten Balkens, der eine Biegesteifigkeit besitzt. Die Eigenschaften einer solchen Struktur wurden in Kapitel 2.1.2 beschrieben. Durch Veränderung der Einspannung der Balkenenden können hier vielzählige unterschiedliche Schwingungsmuster beobachtet werden. Zunächst soll jedoch die Bewegungsgleichung der Struktur hergeleitete werden.

5.1 Herleitung der Bewegungsgleichung

Eine Möglichkeit die Bewegung eines Balkens zu erklären, ergibt sich aus dem *Hamiltonschen Integralprinzip* für die Energieen (vgl. Kapitel 2.1.3).

$$\delta \int_{t_1}^{t_2} (E_k - E_p)\, dt = 0 \text{ oder } \int_{t_1}^{t_2} (E_k - E_p)\, dt = \text{stationär}. \tag{5.1.1}$$

Hier wird das Gleichgewicht zwischen kinetischer Energie E_k und potentieller Energie E_p beschrieben, welches dem Energieerhaltungssatz folgt. Die einzelnen Energieanteile sind mit den Definitionen aus Kapitel 2.1.2 zur Masse $\mu(x) = \rho A(x)$ und der Steifigkeit $EI(x)$ definiert als

$$E_k = \frac{1}{2} \int_0^L \mu(x) \dot{w}^2(x,t)\, dx, \tag{5.1.2}$$

$$E_p = \frac{1}{2} \int_0^L EI(x) w''^2(x,t)\, dx. \tag{5.1.3}$$

Da der Flächenträgheitsmoment $I(x)$ in 5.1.3 nach Gleichung 2.1.4 nur anhängig vom Querschnitt A des Balkens ist, der Aufgrund der Annahmen in Kapitel 2.1.2

konstant ist, wird auch I konstant.
Mit Hilfe der δ-Variation und partieller Integration nach t kann man die kinetische Energie aus 5.1.2 umformen in:

$$\begin{aligned}\delta \int_{t_1}^{t_2} \frac{1}{2}\mu(x)\dot{w}^2 \, dt &= \int_{t_1}^{t_2} \mu(x)\dot{w}\delta\dot{w} \, dt \\ &= \mu(x)\dot{w}\delta w\Big|_{t_1}^{t_2} - \int_{t_1}^{t_2} \mu(x)\ddot{w}\delta w \, dt \end{aligned} \tag{5.1.4}$$

Eine analoge Umformung, allerdings mit zweimaliger partieller Integration nach x, kann man mit der potentiellen Energie in 5.1.3 durchführen und erhält:

$$\begin{aligned}\delta \int_{0}^{L} \frac{1}{2}EI \; w''^2 \, dx &= \int_{0}^{L} EI \; w''\delta w'' \, dx \\ &= EI \; w''\delta w'\Big|_{0}^{L} - \left[EI \; w''\right]' \delta w\Big|_{0}^{L} \\ &\quad + \int_{0}^{L} \left[EI \; w''\right]'' \delta w \, dx \end{aligned} \tag{5.1.5}$$

Da in Gleichung 5.1.4 die Variation δw für die festen Zeitpunkte t_1 und t_2 verschwindet, ergibt sich durch Einsetzen von E_k und E_p in 5.1.2 das Hamiltonsche Prinzip zu:

$$\int_{t_1}^{t_2} \left\{ \int_{0}^{L} \left[\mu\ddot{w} + (EIw'')''\right] \delta w \, dx - (EIw'')'\delta w\Big|_{0}^{L} + EIw''\delta w'\Big|_{0}^{L} \right\} \, dt = 0 \tag{5.1.6}$$

Die beliebige Wahl der Zeitpunkte t_1 und t_2 erfordert, dass die geschweifte Klammer Null sein muss und aus der Variationsrechnung folgt, dass die erste eckige Klammer und die verbleibende Summe auch jeweils für sich verschwinden müssen.

$$\mu(x)\ddot{w}(x,t) + \left[EI \; w''(x,t)\right]'' = 0, \tag{5.1.7}$$

$$\left[EI \; w''(x,t)\right]' \delta w\Big|_{0}^{L} - EI \; w''(x,t)\delta w'\Big|_{0}^{L} = 0. \tag{5.1.8}$$

Nun hat man durch 5.1.7 eine Differentialgleichung 4. Ordnung für die Bewegung des Balkens gegeben. Die zweite entstandene Gleichung 5.1.8 erklärt sich aus den physikalischen Definitionen für den Biegemoment $M_b(x,t) = -EI \; w''(x,t)$ und die Querkraft $Q(x,t) = M_b'(x,t)$ (vgl. Kapitel 2.1.2). Nun kann man die Gleichung

so interpretieren, dass die Summe der an den Rändern des Balkens geleisteten Arbeit verschwinden muss.
Zur eindeutigen Lösung der Bewegungsgleichung muss man nun noch die üblichen Randbedingungen definieren, die nun auch Aussagen über den Biegemoment an den Balkenenden enthalten können. Für einen auf nur einer Seite fest eingespannten Balken mit freiem Ende auf der anderen Seite gilt dann zum Beispiel:

$$w(0,t) = 0 \ , \ w(0,t)' = 0 \ , \ M_b(L,t) = 0 \ , \ M_b(L,t)' = 0. \tag{5.1.9}$$

Für diese, aber auch alle anderen betrachteten Randwerte, zum Beispiel die der gelenkigen Lagerung, verschwinden die Ausdrücke aus Gleichung 5.1.8 jeweils für sich. Es gibt jedoch auch Systeme, bei denen nur die Summe an sich verschwindet und man bezeichnet diese dann als *nicht-lokale Randbedingungen*, die hier nicht weiter betrachtet werden sollen.

5.2 Lösung der Balkengleichung

Auch für die Bewegungsgleichung des Biegebalkens kann man eine analytische Lösung angeben. Man betrachtet dazu folgende Umformung der im vorherigen Abschnitt hergeleitete Gleichung 5.1.7:

$$\ddot{w}(x,t) = -\frac{EI}{\mu} w''''(x,t). \tag{5.2.1}$$

Ähnlich wie bei der Lösung bei der schwingenden Saite kann man hier den Separationsansatz nutzen, um die zeitlichen und räumlichen Ableitungen in 5.2.1 getrennt zu betrachten. Der Ansatz sieht dann wie folgt aus:

$$w(x,t) = \hat{w}(x) f(t). \tag{5.2.2}$$

Durch Einsetzen von 5.2.2 in 5.2.1 erhält man eine Aufteilung in zwei voneinander getrennte gewöhnliche Differentialgleichungen. Definiert man zudem

$$\frac{\ddot{f}}{f} = -\frac{EI}{\mu} \frac{\hat{w}''''}{\hat{w}} := -\omega^2 \tag{5.2.3}$$

und führt κ als Abkürzung ein, ergibt sich folgendes System:

$$\ddot{f}(t) + \omega^2 f(t) = 0, \tag{5.2.4}$$

$$\hat{w}''''(x) - \kappa^4 \hat{w}(x) = 0, \ \kappa^4 = \omega^2 \frac{\mu}{EI}, \ \mu = \rho A. \tag{5.2.5}$$

Man kann nun für beide Gleichungen eine allgemeine Lösung angeben, wobei 5.2.5 nun durch die 4. räumliche Ableitung auch einen Anteil des Sinus Hyperbolicus und des Kosinus Hyperbolicus enthält.

$$f(t) = A\cos(\omega t) + B\sin(\omega t), \tag{5.2.6}$$

$$\hat{w}(x) = a_1 \sin \kappa x + a_2 \cos \kappa x + a_3 \sinh \kappa x + a_4 \cosh \kappa x. \tag{5.2.7}$$

Die Linearfaktoren $a_i, i = 1, \ldots, 4$ muss man nun aus den Randbedingungen ermitteln. Hier soll der Fall des einseitig fest eingespannten Balkens betrachtet werden. Bei Einspannung an $x = 0$ und dem freien Ende bei $x = L$ lauten die Randbedingungen für die Gleichung

$$\hat{w}(0) = 0, \qquad \hat{w}'(0) = 0, \qquad \hat{w}''(L) = 0, \qquad \hat{w}'''(L) = 0. \tag{5.2.8}$$

Durch Einsetzen dieser Bedingungen in die Gleichung 5.2.7 erhält man nun ein homogenes Gleichungssystem für die Linearfaktoren. Will man andere Lösungen als die triviale $a_i = 0,\ i = 1, \ldots, 4$ finden, so muss die Determinante der Systemmatrix verschwinden (siehe [14], Kap. 7). Dies führt mit $\lambda = \kappa L$ zu der Frequenzgleichung

$$1 + \cos\lambda \cosh\lambda = 0. \tag{5.2.9}$$

Daraus kann man nun die Eigenfrequenzen $\omega_k = \lambda_k^2 \sqrt{\frac{EI}{\rho A L^4}}, k = 1, \ldots, \infty$ des Biegebalkens bestimmen. Für $k \geq 4$ stimmen die Werte für λ_k mit einer Abweichung von weniger als 0.01% mit den Nullstellen des Kosinus überein, die ersten 3 Werte werden angegeben.

$$\begin{aligned} \lambda_1 &= 1.8751 \\ \lambda_2 &= 4.6941 \\ \lambda_3 &= 7.8548 \\ \lambda_k &\approx (2k-1)\frac{\pi}{2}, \text{ für } k \geq 4. \end{aligned} \tag{5.2.10}$$

Mit diesen Werten kann man nun die Frequenzen ω_k der Hauptschwingungen des betrachteten Balkens aus den Eigenwerten λ_k berechnen. Die Gleichung für $k \geq 4$ lautet dementsprechend

$$\omega_k = \lambda_k^2 \sqrt{\frac{EI}{\rho A L^4}} \approx \left(\frac{(2k-1)}{2}\right)^2 \pi^2 \sqrt{\frac{EI}{\rho A L^4}}. \tag{5.2.11}$$

Da die Eigenvektoren des Balkens und damit auch die Linearfaktoren a_i aus 5.2.7 nur bis auf eine multiplikative Konstante eindeutig sind, kann ein Wert frei gewählt werden. Daher wird hier der Wert von a_4 zu 1 gewählt. Die restlichen Faktoren ergeben sich dann folgendermaßen:

$$\begin{aligned} \text{Setze} \quad & a_4 := 1 \Rightarrow\ a_2 = -1 \\ \text{mit} \quad & a_1 = -a_3 \Rightarrow \\ & a_3 = -\frac{\sinh\lambda - \sin\lambda}{\cosh\lambda + \cos\lambda} := -\sigma. \end{aligned} \tag{5.2.12}$$

Durch die Einführung der Abkürzung σ erhält man nun eine übersichtliche Form der Eigenfunktionen eines einseitig fest eingespannten Balkens:

$$\hat{w}_k(x) = \cosh(\lambda_k \frac{x}{L}) - \cos(\lambda_k \frac{x}{L}) - \sigma_k \sinh(\lambda_k \frac{x}{L}) + \sigma_k \sin(\lambda_k \frac{x}{L}). \tag{5.2.13}$$

Nun kann man die Bewegungsgleichung 5.2.1 lösen, indem man die Separation aus 5.2.4 und 5.2.5 rückgängig macht und anschließend alle *Hauptschwingungen* $w_k(x,t)$ überlagert.

$$\begin{aligned} w_k(x,t) &= \hat{w}_k(x) f_k(t) = \hat{w}_k(x)(A_k \cos \omega_k t + B_k \sin \omega_k t), && (5.2.14) \\ w(x,t) &= \sum_{k=1}^{\infty} w_k(x,t) \\ &= \sum_{k=1}^{\infty} \hat{w}_k(x)(A_k \cos \omega_k t + B_k \sin \omega_k t) \\ &= \sum_{k=1}^{\infty} \hat{w}_k(x)(C_k \cos \omega_k t - \varphi_k). && (5.2.15) \end{aligned}$$

Die Konstanten A_k und B_k bzw. C_k und φ_k müssen noch an die Anfangsbedingungen angepasst werden. Dies geschieht analog wie bei der Lösung der Bewegungsgleichung der schwingenden Saite mit Hilfe der Orthogonalitätsbeziehung (vgl. 4.2.19 und 4.2.21) und wird daher nicht nochmals betrachtet.
Eine äußere Kraft F in Form einer Last auf den Balken kann wiederum durch eine neue Gleichgewichtslage des Balkens wie in Gleichung 4.2.23 ergänzt werden. Unter Verwendung von Gleichung 4.2.24 für die Berechnung der Gewichtsfaktoren x_k kann dann die Schwingung dargestellt werden durch

$$w(x,t) = \sum_{k=1}^{\infty} x_k \cdot \hat{w}_k(x). \tag{5.2.16}$$

Durch die Wahl anderer Randbedingungen in 5.1.9 können natürlich auch andere Einspannungen des Balkens berechnet werden. Ausführungen dazu finden sich zum Beispiel in [14, chap. 7].

5.3 Herleitung der schwachen Formulierung

Beim einseitig fest eingespannten Balken erhält man folgende Bewegungsgleichung:

$$\begin{aligned} \ddot{w}(x,t) &= -\frac{EI}{\mu} w''''(x,t), && x \in \Omega, t > 0, && (5.3.1) \\ w(x,0) &= f(x) && x \in \Omega, && (5.3.2) \\ w(x,t) &= 0, && x \in \partial\Omega, t \geq 0, && (5.3.3) \\ w'(x,t) &= 0, && x \in \partial\Omega, t \geq 0, && (5.3.4) \\ w''(x,t) &= 0, && x \in \partial\Omega, t \geq 0. && (5.3.5) \end{aligned}$$

Am freien Ende verschwinden zudem die Werte für den Biegemoment M_b und die Querkraf Q. Die linke Seite der Gleichung 5.3.1 kann bis auf einen konstanten Faktor μ wie bei der schwingenden Saite umgeformt werden. Da jedoch in den Nebenbedingungen auch zweiten Ableitungen auftreten können, sind auf der rechten Seite jetzt zwei partielle Ableitungen notwendig, um diese Bedingung in die Gleichung zu integrieren.

$$\begin{aligned}
\int_\Omega \mu \frac{\partial^2 \hat{u}}{\partial t^2} N_i \, d\Omega &= -\int_\Omega EI \frac{\partial^4 \hat{u}}{\partial x^4} N_i \, d\Omega, \qquad (5.3.6)\\
&= -EI\hat{u}'(L) + EI\hat{u}'(0) + \int_\Omega EI \frac{\partial^3 \hat{u}}{\partial x^3} \frac{\partial N_i}{\partial x} \, d\Omega,\\
&= -EI\hat{u}'(L) + EI\hat{u}'(0)\\
&\quad + \left(EI\hat{u}''(L) - EI\hat{u}''(0) - \int_\Omega EI \frac{\partial^2 \hat{u}}{\partial x^2} \frac{\partial^2 N_i}{\partial x^2} \, d\Omega \right).
\end{aligned}$$

Man erhält also eine Steifigkeitsmatrix K, die auf den zweiten Ableitungen der Basisfunktionen basiert. Dies spielt gerade bei der Wahl der Elemente eine wichtige Rolle. Die Bewegungsgleichung bei Einsetzen der Randbedingungen (5.3.4) und (5.3.5) mit den entsprechenden Matrizen lautet jetzt:

$$M \ \ddot{w} + K \ w = F, \qquad (5.3.7)$$

$$M = \int_\Omega \mu N_i N_j \, d\Omega, \qquad (5.3.8)$$

$$K = \int_\Omega EI \nabla^2 N_i \nabla^2 N_j \, d\Omega. \qquad (5.3.9)$$

F kann hier als äußere Kraft interpretiert werden, die zum Beispiel durch Aufbringen einer Last auf den Balken eine neue Gleichgewichtslage des Balkens verursacht. Mit Hilfe dieser schwachen Fomulierung der Bewegungsgleichung ist man nun in der Lage, die Balkengleichung mit *Diffpack* zu berechnen.

5.4 Implementierung in Diffpack

Grundsätzlich geschieht die Implementierung für den schwingenden Biegebalken nach den gleichen Regeln wie die im Abschnitt 4.4 beschriebenen Ausführungen zur schwingenden Saite. Eine wesentliche Veränderung stellt jedoch die Einführung eines zweiten Freiheitsgrades für die Ableitung der Knotenverschiebungen nach dem Ort dar (siehe Kapitel 2.3.2). Daher werden in diesem Kapitel

hauptsächlich die Schritte der Implementierung vorgestellt, die notwendig sind, um die Berechnungen für ein solches System durchzuführen.

`define( MenuSystem& menu, int level )`

Die Administration der Simulation erfolgt wie üblich in der Funktion `adm`, die nach dem Anlegen eines Menüs die Methode `define` aufruft. Hier werden alle Variablen definiert, die als Eingabeparameter übergeben werden können. Zwar wird der Balken nur durch eindimensionale Elemente repräsentiert, jedoch werden hier dreidimensionale Abmessungen angelegt, die durch das Flächenträgheitsmoment I berücksichtigt werden. Zudem sind noch Werte wie die Dichte ρ das Balkens, die hier als einheitlich über die Länge des Balkens angenommen wird, und das Elastizitätsmodul E notwendig, um die Knotenverschiebungen zu berechnen. Die Liste der Parameter lautet dann folgendermaßen:

- gridfile
- width
- height
- length
- rho
- E_modul
- source
- impulse

Die Länge des Balkens kann natürlich auch direkt über das Gridfile selbst gegeben sein. Die reellen Werte `source` und `impulse` bezeichnen die auf einen einzelnen Knoten wirkende Last bzw. die Impulskraft auf diesen.

`scan( )`

Das Einlesen der angegebenen Eingabeparameter erfolgt in der Funktion `scan`. Zudem werden hier die benötigten Werte für den Flächenträgheitsmoment I und die Massenbelegung in der Massenmatrix M, sowie die Biegesteifigkeit, die in die Steifigkeitsmatrix K eingehen, berechnet. Nun müssen die Freiheitsgrade und alle Felder für die Anregung der Schwingung angelegt werden. Wie bereits erwähnt, werden hier zwei Freiheitsgrade benötigt, einer für die Knotenverschiebungen und ein zweiter für die Ortsableitungen. Der Grund dafür lässt sich mit Hilfe der schwachen Formulierung der Balkengleichung aus Abschnitt 5.3 erklären.

Diese enthält die zweite Ortsableitung der Basisfunktionen, welche daher an den Knoten stetige erste Ableitungen besitzen müssen. Dies wurde durch die hermiten Polynome gewährleistet, die die Basisfunktionen für die verwendeten Elemente vom Typ `ElmBH2n1D` bilden. Da die beiden Freiheitsgrade dadurch jedoch in enger Beziehung stehen und nicht als unahhängige Werte betrachtet werden dürfen, werden die Werte hier in einem neu angelegten `FieldHFE` gespeichert, das für jeden Knoten 2 Komponenten enthält. Es werden folglich drei solcher Felder `u_init`, `source` und `impulse` angelegt, um die Anregung der Schwingung an die Dimension des diskreten Problems anzupassen. Die in den Simulationen getesteten Lasten und Impulskräfte betreffen jedoch nur den Freiheitsgrad für die Knotenauslenkung, obwohl ein Wert für die Ableitung auch denkbar wäre.

`fillEssBC( )`

Die Randbedingungen für den Biegebalken können natürlich ebenfalls Werte für die Ableitung an den Randknoten besitzen. Im Falle einer festen Einspannung an einem Ende wird z.B. sowohl die Auslenkung als auch die Ableitung stets Null sein. Dieser zweite Wert für die Ortsableitung wird hier durch den zweiten Randindikator festgelegt und wie der erste über das Objekt `grid` vom Typ `GridFE` abgefragt. Die 2 im Aufruf der Funktionen steht in der folgenden *if*-Abfrage zum einen für diesen Indikator, zum anderen für den zweiten Freiheitsgrad, der dann auf Null gesetzt wird.

```
if( grid->boNode( i , 2 ) )
{
   dof->fillEssBC( i , 2 , 0.0 ) ;
}
```

Neben den Randbedingungen werden die Werte für die Anfangsanregung im Gitter festgelegt und die entsprechenden Felder nun mit den Werten versehen. Zunächst wird jedoch ein Vektor mit der Dimension der gesamten Anzahl der Freiheitsgrade angelegt, der dann mit Hilfe des Objektes `DegFreeFE *dof` in das Feld übertragen wird.

```
if( grid->boNode( i , 3 ) )
{
   iTotalIdx = i * iDofPerNode - iDofPerNode + 1 ;
   vSource( iTotalIdx ) = dSource ;
}

dof->vec2field( vSource , *source ) ;
```

In dem hier dargestellten Fall wird der zuvor festgelegte Lastwert `dSource` für alle indizierten Knoten in den Vektor `vSource` geschrieben. Der Index für den

entsprechenden Freiheitsgrad wird dafür über die Anzahl der Freiheitsgrade pro Knoten `iDofPerNode` ermittelt. Letztlich wird der gesamte Vektor mit der Funktion `vec2field` in das `FieldHFE source` übertragen. Das gleiche geschieht auch mit den Werten für die Impulskraft.

`calcElmMatVec( int e, ElmMatVec& elmat, FiniteElement& fe )`

Für die Berechnungen der Schwingungen des Biegebalkens muss zwar kein lineares Gleichungssystem gelöst werden, trotzdem ist die Funktion `calcElmMatVec`, die aus der Klasse `FEM` überschrieben wird, hier von großer Bedeutung. Bevor nämlich die ursprüngliche Methode von `FEM` aufgerufen wird, um die Elementmatrizen zu erzeugen, muss die Anzahl der benötigten Ableitungen festgelegt werden. Dies geschieht über

```
fe.evalDerivatives( 2 ) ;
```

Nun steht für das Finite Element auch die zweite Ortsableitung zur Verfügung, welche in der Funktion `integrands` entsprechend der schwachen Formulierung der Balkengleichung nach 5.3.9 zur Anwendung kommt. Um die Berechnung der Elementmatrizen zu vervollständigen, muss schließlich noch `FEM::calcElmMatVec` ausgeführt werden.

`integrands( ElmMatVec& elmat, const FiniteElement& fe )`

Ähnlich wie bei der Formulierung der schwingenden Saite besteht die Funktion `integrands` aus zwei Teilen, in denen jeweils der Integrand für eine der Elementmatrizen aufgestellt wird. Wie bereits aus der schwachen Formulierung der Bewegungsgleichung hervor geht, unterscheidet sich die Massenmatrix für den Balken nur um einen konstanten Term von der der schwingenden Saite. Diese Massenbelegung μ wird hier als `m_dMainMass` bezeichnet, die sich aus der Dichte und dem Balkenquerschnitt $\mu = \rho A = konst.$ berechnet. Die Aufsummierung der Basisfunktionen zur Elementmatrix sieht dann wie folgt aus:

```
elmat.A(i,j) += m_dMainMass*fe.N(i)*fe.N(j)*detJxW;
```

Der wesentliche Unterschied zu einem schwingenden Kontinuum zeigt sich nun in der Steifigkeitsmatrix, die nun aus den zweiten Ableitungen der Basisfunktionen erzeugt wird. Zudem gibt es auch hier einen konstanten Faktor `epsilon`, der durch Multiplikation des Elastizitätsmodul E und des Flächenträgheitsmoment I gebildet wird.

```
grad2N = epsilon*fe.d2N(i,1,1)*fe.d2N(j,1,1) ;
elmat.A(i, j)+= grad2N * detJxW ;
```

Die zweite Ableitung im Objekt `fe` wird über die Funktion `d2N(i,dir1,dir2)` abgefragt, wobei im eindimensionalen Modell nur eine Richtungsableitung in Frage kommt. Welche Matrix durch die `integrands`-Funktion berechnet werden soll, wird wiederum durch den `integrationType` festgelegt, der in der Methode `solveProblem` gesetzt werden muss.

`solveProblem( )`

Die Berechnung der Modalanalyse wird in der Funktion `solveProblem` gestartet. Dazu müssen wiederum durch den Aufruf `makeSystem( *dof, *lineq )` die Systemmatrizen für die Freiheitsgrade erzeugt werden, nachdem die essentiellen Randbedingungen durch `fillEssBC` gesetzt worden sind. Jeweils für den Fall `integrationType=STIFFNESSMATRIX` und `MASSMATRIX` kann die im Object `lineq` enthaltene Matrix dann gespeichert und gegebenenfalls in eine Datei geschrieben werden. Die Methoden der Klasse `ModalAnalysis` werden dann wie folgt ausgeführt:

```
setInitialField( *u_init ) ;
setSource( *source ) ;
setImpulse( *impulse ) ;
calcEV( *K, *M ) ;
ModalAnalysis::solveProblem() ;
```

Dadurch, dass die Simulator-Klasse von der Klasse `ModalAnalysis` abgeleitet wurde, erbt sie auch alle Methoden die als `public` oder `protected` gekennzeichnet sind. Wenn die Schwingungssimulation erfolgreich war, können die Ergebnisse, die im `FieldsFE D` gespeichert werden, in das Objekt `database` geschrieben werden.

`resultReport( )`

Die Speicherung der Ergebnisse der Simulation wird mit `resultRaport` durchgeführt. Üblicherweise verwendet man den Befehl `dump`, um das berechnete Feld in das *Field-File* zu schreiben. Dies ist auch für jeden Zeitschritt der Simulation möglich.

```
database->dump( (*D)(1) , tip.getPtr() ) ;
```

Da hier das Feld aus 2 Komponenten besteht, eine für die Knotenverschiebungen und die andere für die Ableitungen, wird hier nur das erste Feld in die Ergebnis-Datei geschrieben. Dies geschieht für jeden Zeitpunkt, der im Objekt `tip` betrachtet wird. Es entsteht eine nummerierte Ausgabe im *Field-File* mit dem Namen „D_1“, die den Zeitschritt, eine Beschreibung des Feldes und die Knotenwerte enthält.

5.5 Berechnung der Schwingungen

Bei einem schwingenden Balken kommen durch die Einbeziehung der Ortsableitung in die Randbedingungen verschiedene Systeme in Frage. So kann der Balken nur auf einer Seite frei schwingen, oder auf beiden Seiten sowohl gelenkig gelagert sein oder fest eingespannt. Zunächst soll der Fall, der in der Herleitung betrachtet wurde, also eine feste Einspannung auf einer Seite bei freiem zweiten Ende berechnet werden.

5.5.1 Beispiel: Einseitig fest-eingespannter Balken

Die Bewegungsgleichung, insbesondere die Eigenschwingungen und Eigenfrequenzen des einseitig fest eingespannten Balkens wurden in Kapitel 5.2 bereits analytisch gelöst. Hier wird nun die numerische Lösung der Gleichung bei Ansetzen einer Impulskraft zum Zeitpunkt $t = 0$ betrachtet. Für die räumliche Diskretisierung wurde eine Unterteilung das Balkens in 16 eindimensionale Elemente von Typ `ElmBH2n1D` gewählt und eine Impuls auf den 13. Knoten aufgebracht. Die Eigenvektoren des Balkens sind in Abbildung 5.1 dargestellt.

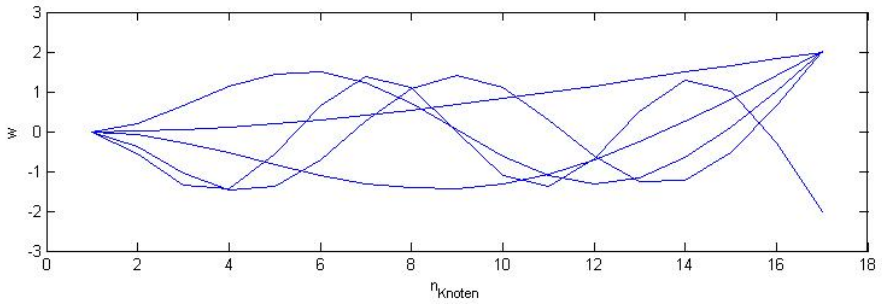

Abbildung 5.1: Eigenvektoren Balken 1

Man erkennt, dass die höheren Eigenschwingungen, bei denen mehrere lokale Maxima auftreten, durch die Diskretisierung mit 17 Knoten zunehmend unglatt werden. Diese Abweichung von den analytischen Eigenschwingungen macht sich auch durch einen zunehmenden Fehler in den Eigenfrequenzen bemerkbar. Daher entsteht in der Simulation bei schlechter Diskretisierung des Balkens auch eine geringe Phasenverschiebung. Allerdings kann die Feinheit des Gitters nicht beliebig vergrößert werden, da die Systemmatrizen dann pro Knoten 2 Freiheitsgrade mehr berücksichtigen müssen und schnell anwachsen. Dies verursacht auch beim Eigenwertlöser oft Probleme und es können in manchen Fällen nur weniger Eigenwerte als angegeben oder gar keine gefunden werden. Dabei kann es hilfreich sein, mehr Eigenwerte als zunächst benötigt, berechnen zu lassen, um ein Ergebnis zu erhalten. Im Falle eines Balkens mit 21 Elemente bricht der Löser

beispielsweise für 5 Eigenwerte die Berechnung bei einer intern festgelegten maximalen Iterationszahl ab. Bereits bei 6 zu berechnenden Werten erhält man dann jedoch die zuvor gesuchten 5 Eigenwerte und ab einer Anzahl von 7 geforderten Werten können all diese berechnet werden. Gerade bei koplexeren Systemen, wie dem in Kapitel 6 betrachteten in zwei Dimensionen, sollten dann auf jeden Fall mehr als die hier verwendeten 5 Eigenpaare für die Überlagerung genutzt werden.

Ist die Berechnungszeit der Simulation eher von geringerem Interesse, so empfiehlt sich bei Systemen mit mehr Freiheitsgraden immer eine feinere Diskretisierung und die Überlagerung von 10 oder mehr Eigenformen. Zum einen lassen sich so Probleme mit mehrfachen Eigenwerten vermeiden, da diese häufig nur gemeinsam approximiert werden können (vgl. Kapitel 6). Außerdem erhält man auf diese Weise eine genauere Berechnung der Schwingungen, da die Eigenfrequenzen und die entsprechenden Eigenmoden näher an der analytischen Lösung liegen. Dies macht Abbildung 5.3 des 5. Eigenvektors des Balkens deutlich, bei dem im Gegensatz zum 2. Eigenvektor (Abbildung 5.2) bereits größere Abweichungen von der blau dargestellten analytischen Lösung der Eigenfunktion sichtbar sind.

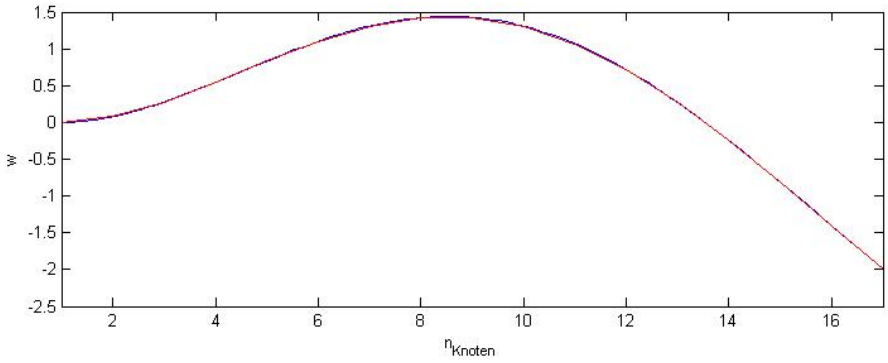

Abbildung 5.2: Eigenvektor 2, Balken1

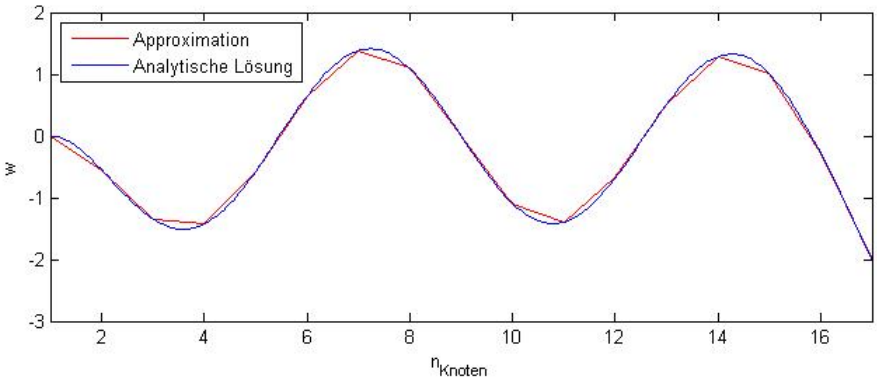

Abbildung 5.3: Eigenvektor 5, Balken1

Durch die Überlagerung der oben gezeigten Eigenschwingungen entsteht eine zeitabhängige Knotenverschiebung, welche für die ersten Zeitschritte in Abbildung 5.4 zu sehen ist.

Wie es bei Impulsanregungen oft der Fall ist, werden zunächst sehr viele Eigenfrequenzen des Systems angeregt. Die dafür zuständigen Gewichtsfaktoren r_k aus Gleichung 3.1.12 berechnen sich aus der Transformation des Impulsvektors mit der Modalmatrix. Die in diesem Beispiel verwendete Impulskraft von $1N/kg$ führt dann zu den Faktoren

- $r_1 = -1.046552$
- $r_2 = -2.147716 \cdot 10^{-1}$
- $r_3 = 9.251867 \cdot 10^{-1}$
- $r_4 = 9.888497 \cdot 10^{-1}$
- $r_5 = 4.073787 \cdot 10^{-1}$

für die 5 Moden.
Bei Abklingen der Oberschwingungen geht das System in eine Schwingung über, die vom ersten Eigenvektor geprägt ist. Dieses Verhalten ist auch schon in den letzten Bildern der Abbildung 5.4 zu erkennen. Durch den Dämpfungsfaktor $\xi = 0.1$ ist die gesamte Schwingung nach etwa 10 Schwingungsperioden abgeklungen und der Balken befindet sich wieder in der Ruhelage.
Betrachtet man das Verhalten bei Aufbringen eine Last auf den Balken, so verändert sich auch hier die Ruhelage und die Schwingung oszilliert um diesen neuen Wert. Die Wahl des Elastizitätsmodul E und der Dichte ρ, sowie der Flächenquerschnitt beeinflussen die Eigenfrequenzen nach Gleichung 5.2.11 und gehen in die Systemmatrizen ein, wie in Kapitel 5.4 beschrieben. Als Standard für die Berechnungen wurden die Tabelle 2.2 angegebenen Werte verwendet, die einem üblichen Stahlbalken entsprechen.

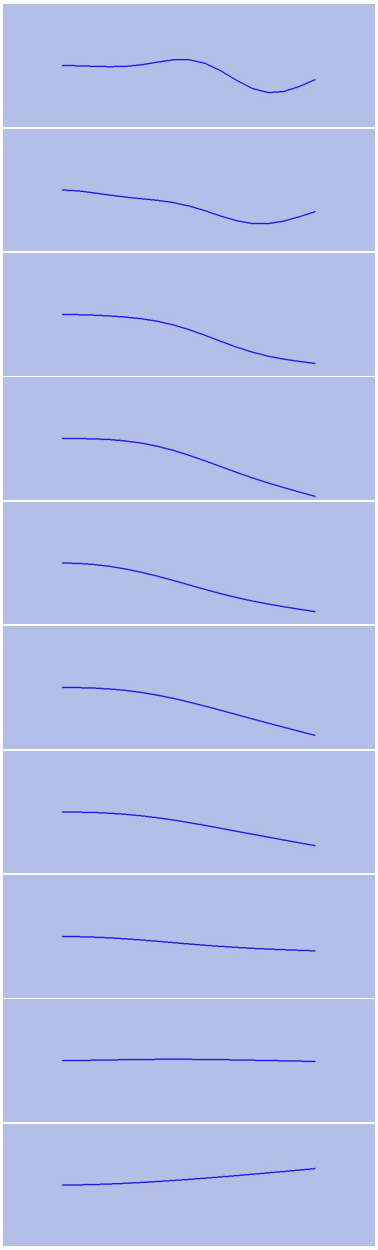

Abbildung 5.4: Schwingender Balken 1

5.5.2 Beispiel: Gelenkig gelagerter Balken

Ein anderes Beispiel für die Schwingung eines Biegebalkens ist der auf beiden Seiten gelenkig gelagerte Balken. Um das System an diese Randbedingungen anzupassen, ist lediglich eine kleine Veränderung im *Grid-File* notwendig. Bei einer gelenkigen Lagerung muss man folgende Randbedingungen beachten:

$$w(0,t) = 0 \ , \ w(L,t) = 0 \ , \ M_b(0,t) = 0 \ , \ M_b(L,t) = 0. \tag{5.5.1}$$

Da die Biegenmomente in der schwachen Formulierung der Bewegungsgleichung nicht angegeben werden müssen, bleibt der Randindikator 1 für die Knotenverschiebung an den Endknoten zu setzen. Dies führt beim Aufstellen der Steifigkeitsmatrix jedoch dazu, dass nur einer der zu den Endknoten gehörenden Freiheitsgrade auf 0 gesetzt wird, also in der Matrix der Wert auf der Diagonale gleich 1 und alle anderen Werte in der Zeile auf 0 gesetzt werden. Daher muss bei der Berechnung der Eigenwerte der Parameter `cut_bound` mit `false` belegt werden, um die Information über die Ableitung an den Enden nicht zu verlieren. In der Klasse `EigenSolver` werden diese abgeschnittenen Werte üblicherweise nach der Berechnung wieder mit der fehlenden 0 ergänzt. Nun werden auch die Werte für diese Knoten berechnet und weisen daher eine Knotenverschiebung ungleich 0 auf.

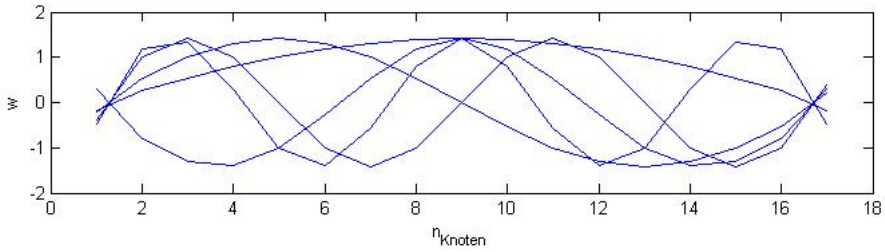

Abbildung 5.5: Eigenvektoren Balken 2

Alternativ könnte der Eigenwertlöser dementsprechend angepasst werden, dass die Freiheitsgrade pro Knoten einzeln untersucht werden und somit nur die Spalten und Zeilen des gesetzten Freiheitsgrades abgeschnitten werden. Durch die spätere Ergänzung mit dem Wert 0 würden dann auch die Verschiebungen an den Rändern verschwinden.
In Abbildung 5.5 kann man erkennen, dass die Eigenvektoren des Balkens bei gelenkiger Lagerung bis auf die angesprochenen Abweichungen und numerische Ungenauigkeiten denen der schwingenden Saite entsprechen. Dies kann man auch durch Einsetzen der Randbedingungen in die allgemeine analytische Lösung überprüfen. Die Ergebnisse der Simulation entsprechen daher bis auf einen durch die Biegesteifigkeit hervorgerufenen skalaren Faktor, denen der schwingenden Saite, bei gleicher Anregung.

Kapitel 6

Mehrdimensionale Balkennetze

Eine sehr interessante Erweiterung der Schwingungen des eindimensionalen Biegebalkens bietet sich durch die Kombination mehrerer solcher Balken, die im 2- oder 3-dimensionalen Raum liegen. Für diesen Zweck wurden dem Paket *Diffpack* neue Elemente hinzugefügt, die die Modellierung von Balkennetzen in höheren Dimensionen ermöglichen.

6.1 Implementierung in Diffpack

Das eindimensionale Balkenelement `ElmBH2n1D` wird durch zwei Freiheitsgrade für die Verschiebungen (translatorisch) und zwei rotatorische Freiheitsgrade beschrieben. Bei der Implementierung im Finite Elemente Modell erhält man somit pro Knoten einen Wert für die Verschiebung und einen für die Ortsableitung. Durch die Drehung eines solchen Elements in weitere Raumrichtungen kann man jedoch mit der gleichen Knotenanzahl auch mehrdimensionale Objekte modellieren (siehe Abbildung 6.1). Man betrachtet dazu den Vektor für die Freiheitsgrade bei 2 Raumrichtungen

$$\bar{v} = [\bar{v}_{u1}, \bar{v}_{w1}, \bar{v}_{\theta 1}, \bar{v}_{u2}, \bar{v}_{w2}, \bar{v}_{\theta 2}]^T. \tag{6.1.1}$$

Mit Hilfe einer 4×6 Transformationsmatrix, die den Zusammenhang zwischen lokalen und globalen Koordinaten herstellt, kann man den Vektor dieses System auf die eindimensionale Darstellung zurückführen.

$$T = \begin{bmatrix} -\sin\alpha & \cos\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.1.2}$$

Der Winkel α beschreibt hier den Winkel der globalen x-Achse zu der lokalen Längsachse des Elements. Mit Hilfe dieser Drehung kann man auf die hermiten Basisfunktionen des eindimensionalen Elements zurückgreifen und braucht

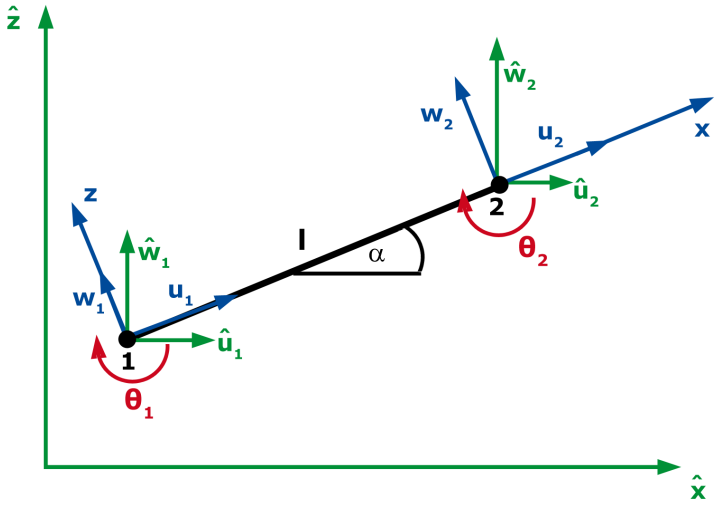

Abbildung 6.1: Balkenelement im 2D-Raum

keinen weiteren Freiheitsgrad zu berücksichtigen. Da bei den Berechnungen des Biegebalkens keine Längsverschiebungen betrachtet werden, werden hier auch die Erweiterungen für das dreidimensionale Element nicht ausgeführt. Somit erhält man eine Möglichkeit die Querschwingungen eines in einer Ebene liegenden Balkennetzwerkes zu berechnen. Die Anpassungen, die dazu notwendig sind, werden anhand des Beispiels 6.2 erläutert.

6.2 Beispiel: 5-Stern-Balken

Die Anwendungen für ein Balkennetzwerk aus linearen Balkenelementen sind vielfältig. Hier soll das Prinzip der Modalanalyse auf einen Stern aus 5 Balken mit einem gemeinsamen mittleren Knoten angewandt werden. Alle Balken sollen im gleichen Winkel zueinander stehen, das heißt der Winkel zwischen zwei benachbarten Balkenkomponenten soll jeweils 72° betragen. Entsprechend dieser Vorgabe sind die Knoten im *Grid-File* nun zu einer grundlegenden Komponente auf der x-Achse in der xy-Ebene ausgerichtet. Jeder Teil-Balken besteht aus 10 Elementen vom Typ `ElmBH2n2D` und wird somit über zwei 2-dimensionale Knoten pro Element festgelegt. Das ergibt eine gesamte Knotenanzahl von 51 aufgrund des gemeinsamen Knotens in der Mitte. Die Reihenfolge der Knoten im *Grid-File* ist natürlich frei wählbar und kann zu einer für den Eigenwertlöser schlecht konditionierten Matrix führen. Daher wurde für dieses etwas größere Beispiel die Funktion `renumberNodes( *grid )` verwendet, die in der Klasse `Puttonen` zu finden ist. Diese stellt einen Algorithmus zu Reduzierung der Bandbreite der Ma-

trix bereit, der von *J. Puttonen* entwickelt wurde (vgl. [7]).
Die Anregung der Schwingung wird nun durch eine Last auf die Knoten 5 und 6 der 2. und 3. Komponente erzeugt, also die zwei Balken, die sich gegen den Uhrzeigersinn nach der grundlegenden Komponente befinden. Somit stellt sich ein unsymmetrisches Gleichgewicht ein, was die Überlagerung mehrerer Eigenformen erfordert. Die Berechnungen haben gezeigt, dass mit den zuvor verwendeten 5 Moden keine Eigenvektorapproximation möglich war. Über den Grund erfährt man leider im Eigenwertlöser *Arpack* sehr wenig, nur die interne Iterationszahl wird überschritten. Für solche Strukturen ist es zudem Sinnvoll die Anzahl der zu berechnenden Eigenvektoren nach oben zu setzen. Hier wurden nun 10 Moden angegeben, für die auch der Eigenwertlöser ein Ergebnis liefern kann. Die approximierten Eigenwerte für dieses Beispiel lauten in aufsteigender Reihenfolge:

- $5.798723 \cdot 10^{-1}$
- 5.884792
- $1.108893 \cdot 10^{1}$
- $1.108893 \cdot 10^{1}$
- $1.108894 \cdot 10^{1}$
- $1.865878 \cdot 10^{1}$
- $5.886028 \cdot 10^{1}$
- $8.428124 \cdot 10^{1}$
- $8.428129 \cdot 10^{1}$
- $8.428135 \cdot 10^{1}$.

Man erkennt hier zunächst, dass nach den 2 kleinsten Eigenwerten 3 Werte berechnet werden, die sich nicht oder kaum unterscheiden. Ebenso weisen die letzten 3 Werte nur eine Abweichung von 10^{-4} bzw. 10^{-5} auf. Dies lässt sich so interpretieren, dass hier mehrfache Eigenwerte auftreten, die geringe numerische Ungenauigkeiten besitzen. Dies könnte auch die Probleme bei der Approximation von wenigen Eigenwerten in *Arpack* erklären. Hier wurde als Randbedingung eine feste Einspannung gewählt, das heißt, dass die Verschiebungen und die Ableitungen an den 5 Randknoten verschwinden. Somit kann das zu lösende Eigenwertproblem auch um die entsprechenden 10 Freiheitsgrade reduziert werden, jedoch besitzen die Systemmatrizen dann immer noch eine Dimension von 92×92. Es zeigt sich, dass die CPU-Zeit für die Berechnung nun $18.8s$ bei 150 Zeitschritten beträgt. Im Vergleich zu $6.5s$ bei einem eindimensionalen Modell ist das mehr als das Doppelte, was im wesentlichen durch die aufwänidigere Approximation der Eigenvektoren und die erhöhte Anzahl der verwendeten Moden hervorgerufen wird.

In Abbildung 6.2 ist die Verbiegung der Struktur kurz nach Aufbringen der Last zu sehen. Die Schwingung setzt sich dann zum Mittelpunkt des Balkennetzes fort und klingt langsam ab. Der Gleichgewichtszustand, der bei positiver Dämpfungskonstante erreicht wird, zeigt schließlich wie zu erwarten eine etwas größere Verschiebung an den belasteten Knoten.

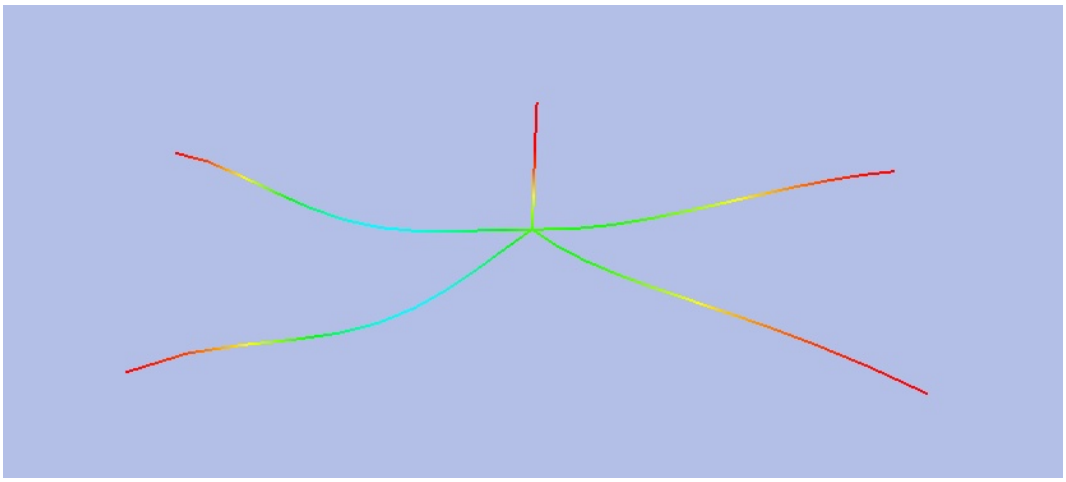

Abbildung 6.2: Balkennetz bei Belastung

Die Eigenvektoren einer solchen Struktur sind zu vergleichen mit einer dreidimensionalen Variante der Eigenformen einer einzelnen fest eingespannten Balkenkomponente. Jedoch sind die Auslenkungen der Komponenten nicht gleich groß. In diesem Beispiel zeigt sich etwa bei der ersten Eigenform eine größere Amplitude auf der im *Grid-File* zuerst angegebenen Balkenkomponente, die auf der y-Achse liegt. In Abbildung 6.3 ist der Verlauf von dieser ersten und der zweiten Balkenkomponente mit den unterschiedlichen Amplituden dargestellt.
Einen Ausgleich für diese unsymmetrische Verteilung erhält man durch die Eigenvektoren der mehrfachen Eigenwerte, wo die Verschiebungen der ersten Komponente numerisch gesehen verschwinden. Die verbleibenden Komponenten weisen paarweise alternierende Eigenformen auf oder die Verschiebung verschwindet bei zwei Elementen. Die Abbildung 6.4 zeigt den 5. Eigenvektor, bei dem nur die Komponenten 2 und 5 eine Auslenkung zeigen, die jeweils der 1. Eigenform eines fest eingespannten, eindimensionalen Balkens entsprechen. Daher verschwindet die Ortsableitung am mittleren Knoten bei einem Wert von $-1.677865 \cdot 10^{-14}$ numerisch gesehen, was jedoch die lineare Interpolation in der Grafik 6.4 nicht ganz zeigen kann.
Durch die zum Teil unsymmetrisch auftretenden Eigenvektoren lässt sich wiederum erklären, warum bei solchen Strukturen mehrere Eigenformen berechnet werden müssen. Zudem erklärt sich dadurch die Vielfachheit mancher Eigenwerte, die zu den paarweise auftretenden Eigenschwingungen der Komponenten gehören.

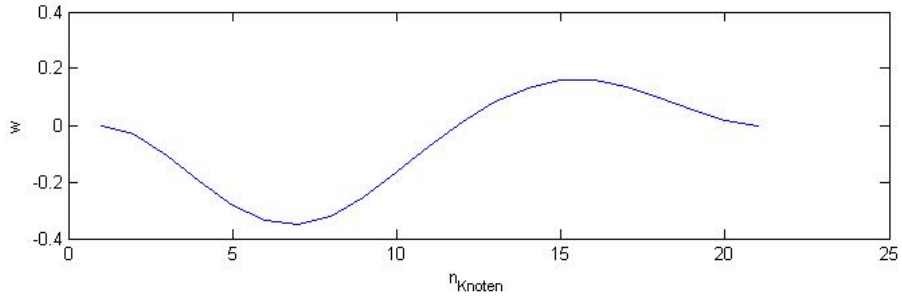

Abbildung 6.3: 1. Eigenvektor Balkenkomponenten 1/2

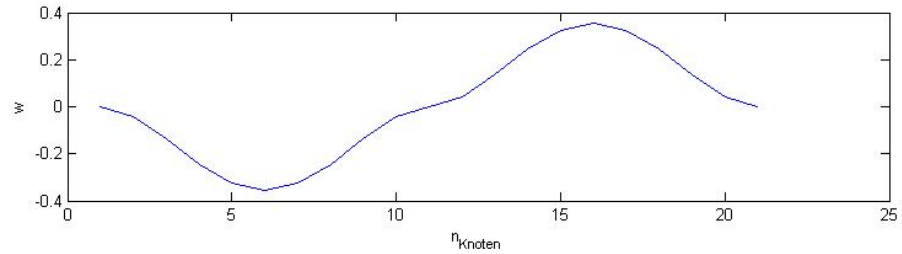

Abbildung 6.4: 5. Eigenvektor Balkenkomponenten 2/5

Auch für diese Struktur können durch Veränderung der Randbedingungen oder der Anregung zahlreich Variationen der Simulation durchgeführt werden. Eine gute Verbesserung der graphischen Darstellung könnte man mit einer geeigneten Software erreichen, die auf die Informationen der Ableitung an der Knoten eingeht. Diese Möglichkeit konnte bisher bei den Anpassungen des Skriptes `simres2vtk` nicht berücksichtigt werden, da hier zunächst die Erweiterung auf mehrere physikalische Raumdimensionen notwendig war. Gerade die Abweichung der Dimension der Elemente von der Dimension der gesamten Struktur ist auch in *Diffpack* noch in vielen Bereichen eine Neuerung, welche in Zukunft noch mehr Anwendungsbereiche ermöglichen soll.

Kapitel 7

Fehleranalyse

Wie alle numerischen Verfahren kann auch die Modalanalyse nur eine Approximation der Lösung der Bewegungsgleichungen liefern. Der dabei entstehende Fehler setzt sich jedoch aus verschiedenen Faktoren zusammen, die durch die verschiedenen Schritte der Berechnungsmethode bedingt sind. In diesem Kapitel soll daher betrachtet werden, welche Fehlerquellen auftreten und welchen Einfluss sie auf die berechnete Lösung haben.

7.1 Diskretisierungsfehler

Die Aufteilung des Lösungsgebietes Ω stellt bei einem Finite Elemente Verfahren immer den ersten Schritt zur Beschreibung des Systems dar. Dadurch wird die Lösung in einem endlich dimensionalen Unterraum des ursprünglich unendlichdimensionalen Lösungsraumes gesucht. Der durch diese Aufteilung entstehende Fehler wird als *Diskretisierungsfehler* bezeichnet und hängt von verschiedenen Faktoren ab. Zum einen spielt die Feinheit des zu Grunde gelegten Gitters eine entscheidende Rolle, also die Anzahl der Elemente, die in das Gebiet gelegt werden. Je mehr Elemente verwendet werden, desto genauer wird auch die Approximation. Allerdings steigt damit auch die Dimension des Problems und damit der Rechenaufwand, der notwendig ist. Der zweite wichtige Aspekt ist die Wahl der Elemente, insbesondere der Grad der Polynome, die diesen als Basisfunktionen zugrunde liegen. Hier zeigt sich, dass bereits quadratische Ansatzfunktionen deutlich bessere Ergebnisse liefern, als die linearen. Dazu benötigt man jedoch einen weiteren Knoten, um diese eindeutig zu definieren, was wiederum ein Anwachsen der Dimension des numerischen Problems bedeutet. Ein ähnlicher Fall findet sich bei den Hermiten Polynomen, die bei den Balkenelementen genutzt werden. Hier ist kein weiterer Knoten notwendig, jedoch wird die Anzahl der Basisfunktionen für die Berechnung der Elementmatrizen verdoppelt, da die Ableitung der Verschiebung berücksichtigt werden muss.

	5 Knoten	10 Knoten	15 Knoten	20 Knoten	30 Knoten
EV_1	$6.9 \cdot 10^{-16}$	$8.07 \cdot 10^{-7}$	$2.9 \cdot 10^{-6}$	$3.4 \cdot 10^{-6}$	$4.4 \cdot 10^{-6}$
EV_2	0.344	0.09	0.05	0.03	0.02
EV_3	-	0.25	0.13	0.08	0.04
EV_4	-	0.49	0.24	0.15	0.08
EV_5	-	0.81	0.4	0.25	0.13

Tabelle 7.1: Abweichungen (L_2-Norm) der Eigenvektoren

	5 Knoten	10 Knoten	15 Knoten	20 Knoten	30 Knoten
EW_1	2.58591	0.50844	0.209943	0.113954	0.0489053
EW_2	10.2658	2.04115	0.841228	0.456262	0.195706
EW_3	-	4.60731	1.89742	1.02814	0.440645
EW_4	-	8.16732	3.38141	1.83118	0.784097
EW_5	-	12.5059	5.2899	2.86655	1.22651

Tabelle 7.2: Abweichungen der Eigenwerte in %

Für die Modalanalyse ist es wichtig, dass die verwendeten Eigenformen durch die Diskretisierung gut dargestellt werden können. Die Eigenvektoren, die zu größeren Eigenfrequenzen gehören, zeigen dabei stets die größeren Schwankung in der Steigung, da sie mehrere lokale Extremwerte aufweisen. Man erhält somit eine ausreichend gute Diskretisierung und folglich auch gut genäherte Eigenformen, indem man die Feinheit des Gitters an den höchsten verwendeten Eigenwert und den zugehörigen Eigenvektor anpasst. Dies kann in der Simulation natürlich nur beobachtet werden, wenn die entsprechenden Eigenfrequenzen auch angeregt werden, da sonst diese Eigenformen für die Berechnung nebensächlich sind.
Tabelle 7.1 zeigt die Abweichungen der approximierten Eigenvektoren der Saite von den analytischen aus Kapitel 4.2 in der L_2-Norm. Bei einer Diskretisierung mit 5 Knoten kann der Eigenwertlöser nur 2 Werte berechnen, die des ersten Vektors stimmen jedoch schon sehr exakt. Die Verfeinerung des Gitters zeigt dann eine Verbesserung der Werte bis zum Faktor 2, nur der erste zeigt eine Verschlechterung. Dies liegt daran, dass die Eigenvektoren nur bis auf die Norm eindeutig bestimmt sind und daher auch nur die normierten Werte verglichen werden. Die 5 Knoten des ersten Vektors liegen jedoch bereits numerisch exakt auf der Sinuskurve des analytischen Eigenvektors. Da bei den späteren Vektoren mehrere Einträge zum Fehler beitragen wächst dieser in der Summe an, was auch mit der Maschinengenauigkeit zusammen hängt. Dieses Phänomen tritt bei den Eigenwerten nicht auf, was in Tabelle 7.2 zu erkennen ist.
Die Diskretisierung muss also immer in Abstimmung mit der gewünschten Genauigkeit und der damit zusammenhängenden Berechnungsgeschwindigkeit gewählt

werden. Die Ergebnisse der Modalanalyse sind dabei sowohl von den Eigenvektoren abhängig, die als Grundlage für die Modenüberlagerung genutzt werden, als auch von den Eigenwerten, aus denen die Eigenfrequenzen berechnet werden, die wiederum die Schwingungsperiode bestimmen. Man muss hier also den in der Numerik häufig auftretenden Kompromiss zwischen Genauigkeit der Ergebnisse und Geschwindigkeit der Berechnung oder der Lösbarkeit des Problems eingehen.

7.2 Fehler in der Modalanalyse

Da bisher beschrieben wurde, welche Auswirkungen die Wahl der Diskretisierung haben kann, werden nun die Folgen und Probleme in der Berechnung der Schwingungen betrachtet. Grundlage für eine genaue Überlagerung der Moden ist sicherlich eine gute Approximation der Eigenformen. Diese stellen insbesondere die Einträge der Modalmatrix dar, die als Transformationsmatrix auch für die Güte des modalen Systems verantwortlich ist. Somit kann die Berechnung der Anfangswerte für Amplitude und Phasenverschiebung auch nur so genau sein, wie es diese Transformation mit den approximierten Eigenvektoren zulässt. Bei der Last auf der Saitenmitte im ersten Beispiel 4.5.2 werden etwa analytisch betrachtet nur ungerade Eigenfrequenzen angeregt. Die Transformation des Lastvektors R sollte daher auch bei den geraden modalen Faktoren für die Last verschwinden, die für die neue Ruhelage verantwortlich sind. Hier zeigt sich ein direkter Bezug zum Wert des jeweiligen Eigenvektors am Knoten, an dem die Last angreift. Analytisch liegt hier eine Nullstelle der Eigenform, in der Approximation ergibt sich jedoch ein Fehler im Bereich von 10^{-7} bis 10^{-12} im Falle von 25 Knoten auf der Saite. Dieser Wert wird zur Berechnung des modalen Faktors noch durch das Quadrat der zugehörigen Frequenz ω^2 dividiert, was den Fehler noch auf Werte ab 10^{-10} verkleinert. Befindet sich kein Knoten in der Mitte der Saite, so ist die Genauigkeit von der Symmetrie der 2 benachbarten Punkte abhängig. Jedoch zeigen die Simulationen, dass die Fehler aufgrund dieser Ungenauigkeiten in der Orthogonalität nur geringe Auswirkungen auf das Ergebnis haben.
Ein bereits in der Einführung der Modalanalyse beschriebener Fehler entsteht dadurch, dass bei der Simulation nur einzelne Moden betrachtet werden. Üblicherweise werden jedoch auch andere Frequenzen angeregt, wenn auch nur gering. Das bedeutet, dass der so entstehende Fehler direkt von der Anregung der Schwingung abhängig ist. Bei Aufbringen einer Last nimmt die Anregung mit der Größe der Eigenfrequenz ab, da diese bei der Division durch ω^2 sogar quadratisch in die Berechnung mit eingeht. Dies ist bei einer Impulsanregung nicht der Fall. Daher treten beispielsweise bei Aufbringen einer Impulskraft auf den mittleren Knoten der Saite alle ungeraden Eigenfrequenzen auf. In Abbildung 7.1 wird die Simulation mit 5 Moden mit der Analytischen Lösung mit 15 Moden verglichen.
Auch wenn die Berechnung mit 15 Moden aufgrund der Anregung unendlich vieler Eigenfrequenzen keine gute Approximation darstellt, so lässt sich hier dennoch

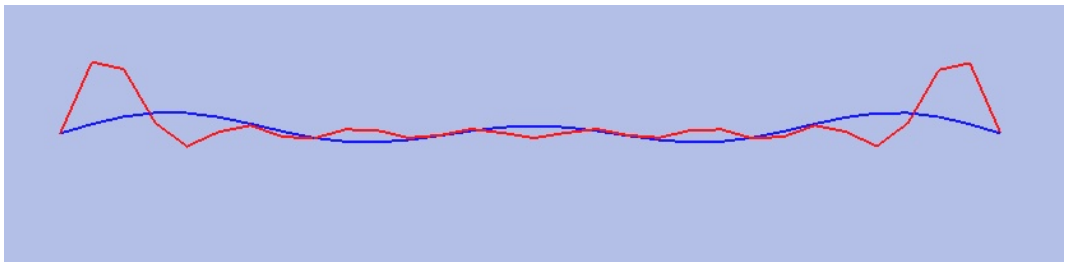

Abbildung 7.1: Saite bei Impulsanregung mit 5 und 15 Moden

der Unterschied zur Verwendung von nur 5 Moden deutlich erkennen. Der Vorteil einer solchen Schwingung ist es, dass die modale Dämpfung des Systems die Anteile der höheren Frequenzen sehr schnell verschwinden lässt. Daher ist bereits nach der ersten Schingungsperiode im Bezug auf die kleinste Eigenfrequenz ein Zustand erreicht, bei dem die Auswirkungen der Moden $k > 5$ sehr gering sind. Die 11. Eigenfrequenz geht beispielsweise zum Zeitpunkt $t = 0.8\ T$ maximal mit einem Faktor von $1.26432 \cdot 10^{-5}$ in die Berechnung mit ein. Der entsprechende Faktor für die erste Frequenz liegt mit $1.65875 \cdot 10^{-3}$ bereits wesentlich höher und wird auch geringer gedämpft. Diese Wirkung des Dämpfungsterms $e^{-\omega_k \xi t}$ bedeutet für die Modalanalyse, dass die Anzahl der verwendeten Moden gerade bei Impulsanregung an den betrachteten Zeitraum angepasst werden muss. Nur für die Auslenkungen direkt nach dem Impuls sind sehr viele Eigenformen zu betrachten, für das spätere Schwingungsverhalten genügt eine Simulation mit wenigen Moden.

7.3 Fehler der Zeitdiskretisierung

Die Simulation der Schwingung kommt letztendlich durch eine Überlagerung der approximierten Eigenformen mit entsprechenden Gewichtsfaktoren zustande, die für jeden Zeitschritt berechnet werden. Die Festlegung dieses Zeitschrittes Δt ist Bestandteil der Zeitdiskretisierung, die in Kapitel 3.3 diskutiert wurde. Die Lösung der entkoppelten Bewegungsgleichungen mit dieser Diskretisierung kann ebenfalls eine Fehlerquelle in der Schwingungssimulation darstellen. Dabei sind jedoch die zwei Methoden der Berechnung der Gewichtsfaktoren zu unterscheiden.

Beim trigonometrischen Ansatz wird die Lösung für die Mode k im wesentlichen mittels der Gleichung 3.3.4

$$x_k = x_k^0 + C_k \cdot e^{-\omega_k \xi t} \cos(\nu \omega_k t - \varphi_k).$$

berechnet. Da die Faktoren x^0, C, ω, ξ, ν und φ alle unabhängig von der Zeit t sind, wird im Grunde eine gedämpfte Kosinusschwingung ausgewertet. Der Rechenaufwand und die Genauigkeit der Approximation hängen somit direkt von

den Funktionen $\exp(x)$ und $\cos(x)$ ab. Diese können jedoch ohne zusätzlichen Zeitaufwand im Vergleich zur Methode der Finiten Differenzen ausgewertet werden. Die Genauigkeit entspricht der Maschinengenauigkeit in C++ und kann somit als numerisch exakt angesehen werden.

Der zweite Ansatz mit Finiten Differenzen basiert auf der Berechnung der Gewichtsfaktoren mit Hilfe der Gleichung 3.3.10

$$x^+ = \frac{2x - x^- + \Delta t D x^- - \Delta t^2 \omega^2 x + \Delta t^2 r}{1 + D\frac{\Delta t}{2}}.$$

Da die Werte x und x^- aus den vorhergehenden Zeitschritten zur Verfügung stehen und Δt^2, D, ω sowie r wiederum zeitunabhängig gewählt wurden, erscheint diese lineare Form der Berechnung zunächst etwas einfacher. Die Auswertung der Berechnungszeiten zeigt jedoch, dass die Berechnung der Faktoren auf diese Weise keinerlei Auswirkungen auf die Berechnungzeit der Schwingungen hat. Als ein Nachteil kann zum einen die bedingte Stabilität dieser Methode gesehen werden, da nur unter der Bedingung $\Delta t < 2/\omega_k$ die absolute Stabilität bei der Berechnung garantiert werden kann. Da jedoch die Frequenz aller berücksichtigten Moden zu Stabilität beiträgt und die Werte schnell anwachsen können, kann der Zeitschritt Δt sehr klein werden. Dies führt zu wesentlich längeren Berechnungen bei größeren Zeitintervallen. Neben dieser Einschränkung an den Zeitschritt Δt kann sich durch die Berechnung aus den Werten der vorherigen Zeitschritte x und x^- in 3.3.10 auch der Fehler in der Simulation fortpflanzen. Das Verfahren besitzt zwar eine Konsistenzordnung von zwei (vgl. [9]), jedoch ist damit auch ein Fehler in Abhängigkeit von den räumlichen Diskretisierung gegeben. Diese Ungenauigkeiten verdeutlichen den Nachteil dieser Methode im Vergleich zur Verwendung des trigonometrischen Ansatzes.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurde das Verfahren der Modalanalyse zur Berechnung von Schwingungen vorgestellt und die Implementierung für das Softwarepaket Diffpack umgesetzt. Zu Beginn wurden die physikalischen Modelle aufgestellt, mit deren Hilfe man aus den physikalischen Grundgesetzen die Bewegungsgleichungen herleiten kann. Um diese zu Lösen wurde dann auf der Basis funktionalanalytischer Grundlagen die Methode der Finiten Elemente eingeführt und die Implementierung in Diffpack beschrieben. Daraus erhält man ein lineares System, das eine diskrete Beschreibung der Gleichung auf einzelnen Elementen liefert. Die zeitabhängige Lösung dieser Gleichung kann dann mit der Modalanalyse durchgeführt werden.
Die Modalanalyse beschreibt die Möglichkeit, eine Schwingung durch die Überlagerung der einzelnen Eigenschwingungen der Struktur zu berechnen. Dazu benötigt man die Eigenfrequenzen und Eigenformen des Systems, welche sich mit einem Eigenwertlöser approximieren lassen. Hierfür konnte die freie Software Arpack genutzt werden, die auf Basis der Arnoldi-Faktorisierung mehrere Eigenwerte und Eigenvektoren großer Systeme bestimmen kann. Durch die Kenntnis der niedrigsten Eigenfrequenzen kann man dann Gewichtsfaktoren für jede Mode oder Eigenform bestimmen, um diese in der Anzahl der approximierten Eigenpaare zu überlagern.
Für die Implementierung dieser Methode wurde eine neue Klasse zum Softwarepaket Diffpack hinzugefügt, die zunächst die Berechnung der Eigenpaare mit Arpack durchführt. Nachdem dann die individuell angegebene Anregung der Schwingung eingelesen und transformiert wurde, beginnt die Überlagerung der Moden für jeden Zeitschritt. Die Ausgabe kann im Anschluss mit Skripten, die Diffpack beigefügt sind, in visuell darstellbare Daten umgewandelt werden.
Anhand von 3 Beispielmodellen wurde die Funktionsweise des Programms überprüft und die Ergebnisse analysiert. Gerade der Vergleich mit den analytischen Lösung in 2 Fällen konnte die hohe Genauigkeit nachweisen, mit der die Methode arbeiten kann. Je nach Aufbau des Beispiels mussten jedoch Anpassungen in den Einstellungen des Eigenwertlösers vorgenommen werden. Gerade die Anzahl der

zu bestimmenden Moden musste oft verändert werden, um eine gute Lösung zu erhalten. Insgesamt konnte jedoch gezeigt werden, dass das Verfahren für eine Vielzahl von Schwingungssimulationen sehr gut geeignet ist und eine wertvolle Erweiterung für Diffpack darstellen kann.

Alle in dieser Arbeit betrachteten Modelle unterlagen gewissen Einschränkungen der möglichen Schwingungen. Bei einer Weiterführung könnten man jedoch auch Schwingungen betrachten, die nicht nur Bewegungen senkrecht zur Ausrichtung der Struktur zulassen. Man würde dann weitere Freiheitsgrade einführen, die über die Betrachtung der Ableitung im Falle der Hermiten Polynome hinausgehen. Was die Randwerte betrifft, so sind noch Verbesserungen in der Schnittstelle zu Arpack möglich, wo die festen Werte auf dem Rand auch bei unsymmetrischen Matrizen nicht betrachtet werden müssen. Auch die Funktionsweise des Eigenwertlösers im Bezug auf Abbruchkriterien könnte intensiver untersucht werden, um in möglichst vielen Fällen eine Approximation der Werte zu erhalten. Gerade die Ergebnisse bei noch wesentlich größeren als den hier Betrachteten Beispielen wären auch für die praktische Anwendung interessant.
Somit sind noch einige Erweiterungen der Implementierung und Ausführung denkbar. Für viele Systeme in der Praxis kann jedoch auch die hier zur Verfügung gestellte Methode gute Ergebnisse liefern. Gerade die Vielseitigkeit von Diffpack als Lösungsverfahren von mathematischen Problemen kann somit genutzt und noch erweitert werden.

Anhang A

Inhalt der CD

Tabelle A.1: Allgemeines

DA_SPeetz_08.pdf	PDF-Version dieser Diplomarbeit
RTA_DA_SP.lic	Runtime-Lizenz für Diffpack
simres2vtk.exe simres2vtk.cmd	Diffpack-Skript zum Erzeugen von vtk-Dateien Ausführbare Datei für simres2vtk mit Parametern

Tabelle A.2: Quelldateien [/src]

EigenSolver.h EigenSolver.cpp	C++ Klasse zur Anbindung an den Eigenwertlöser Arpack
ModalAnalysis.h ModalAnalysis.cpp	Implemetierung der Modalanalyse
FieldHFE.h	Zusatzklasse für Felder mit hermiten Basisfunktionen
/Diffpack_Files	Angepasste Diffpack_Dateien (Quelle [10])

Tabelle A.3: Schwingende Saite [/String1]

String1.exe String1_aufruf.cmd	Applikation für die schwingende Saite Ausführbare Datei zum Starten der Applikation
String1.i String1.grid	Input-Datei mit Parametern für die Simulation Grid-File zur Definition des Modells
/src	Quelldateien für die Simulation
/Ergebnisse	Bilder und Videos der Simulation

Tabelle A.4: 1D Balken [/Beam1]

Beam1.exe Beam1_aufruf.cmd	Applikation für die schwingende Saite Ausführbare Datei zum Starten der Applikation
Beam1.i Beam1_21.grid	Input-Datei mit Parametern für die Simulation Grid-File zur Definition des Modells
/src	Quelldateien für die Simulation
/Ergebnisse	Bilder und Videos der Simulation

Tabelle A.5: 2D Balken-Netz [/Beam2D]

Beam2D.exe Beam2D_aufruf.cmd	Applikation für die schwingende Saite Ausführbare Datei zum Starten der Applikation
5Star50E.i 5Star_50E.grid	Input-Datei mit Parametern für die Simulation Grid-File zur Definition des Modells
/src	Quelldateien für die Simulation
/Ergebnisse	Bilder und Videos der Simulation

Abbildungsverzeichnis

Literaturverzeichnis

[1] *Elastizitätsmodul*, Website, 2008, Verfügbar online auf http://de.wikipedia.org/wiki/Elastizitäsmodul; besucht am 17.11.2008.

[2] *Flächenträgheitsmoment*, Website, 2008, Verfügbar online auf http://de.wikipedia.org/wiki/Flächenträgheitsmoment; besucht am 17.11.2008.

[3] Hans Wilhelm Alt, *Lineare funktionalanalysis*, Springer Verlag, Berlin [u.a.], 1992.

[4] Klaus-Juergen Bathe, *Finite-elemente-methoden*, Springer Verlag, Berlin [u.a.], 1986.

[5] Dietrich Braess, *Finite elemente*, Springer Verlag, Berlin [u.a.], 1992.

[6] Wolfgang Dahmen and Arnold Reusken, *Numerik für ingenieure und naturwissenschaftler*, Springer Verlag, Berlin, Heidelberg, 2006.

[7] InuTech GmbH, *Diffpack kernel and toolboxes documentation 4.0.00*, Website, 2003, Verfügbar online auf http://www.diffpack.com/diffpack/refmanuals/current/index.html.

[8] Holzmann, Meyer, and Schumpich, *Technische mechanik 3, festigkeitslehre*, B.G.Teubner Verlag, Stuttgart [u.a.], 2002.

[9] Arieh Iserles, *A first course in the numerical analysis of differential equations*, Cambridge University Press, Cambridge, 1996.

[10] Melanie Jahny, *Numerische lösung nichtlinearer balkengleichungen mit dem finite elemente software-programm Diffpack*, Diplomarbeit, Institut für Mathematik, Universität Augsburg, 2008.

[11] Martin Klein, Peter Kiehl, and Norbert Breutmann, *Einführung in die DIN-Normen*, B.G.Teubner Verlag, Stuttgart [u.a.], 2001.

[12] Hans Peter Langtangen, *Computational partial differtial equations*, Springer Verlag, Oslo, 2000.

[13] Michael Link, *Finite elemente in der statik und dynamik*, B. G. Teubner Verlag, Stuttgart, 2002.

[14] Kurt Magnus and Karl Popp, *Schwingungen*, B. G. Teubner Verlag, Wiesbaden, 2005.

[15] D. C. Sorensen, *Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations*, Technical report, Rice University, Department of Computational and Applied Mathematics, 1995.