Projektgruppe Praktische Mathematik

Projekt Flugzeugtragflügel

mit Erweiterung auf Fachwerke

Manuel Baumann
Matrikelnummer: 314394

Pavel Buran
Matrikelnummer: 314728

Christoph Menzel
Matrikelnummer: 315829

Frank Wagner
Matrikelnummer: 316058

Betreuer: Dr. Michael Karow

Wintersemester 2008/09
Abgabe: 23. April 2009

Inhaltsverzeichnis

1 Vorwort und Motivation

Bei unserem Projekt handelt es sich um eine interdisziplinäre Aufgabe zwischen Bereichen der höheren, technischen Mechanik und angewandten Mathematik. Zu Beginn des Projektes steht die mathematische und physikalische Modellierung der Aufgabenstellung. Wie in Kapitel 2 ausführlich hergeleitet wird, führt die Annahme eines einseitig eingespannten Balkens (Flugzeugrumpf) unter Streckenlast (Luftkräfte, Triebwerk) auf eine partielle Differentialgleichung, die nicht durch analytische Methoden gelöst werden kann. Man verwendet daher numerische Methoden, um das Problem auf die Lösung eines linearen Gleichungssystems zurückzuführen, welches in hoher Dimension vorliegt und daher effiziente Algorithmen benötigt.

In einem späteren Stadium des Projektes geht es um die Simulation von Schwingungen, die aus der dynamischen Belastung der Tragfläche resultieren. Dies ist auch aus Sicht der Anwendung eine interessante Problemstellung, da das Flattern der Tragflächen unter bestimmten Belastungen zu ernsten Resonanzproblemen führen kann.

Als Abschluss wurde ein Fachwerk aus Balken *der gleichen Mathematik* unter verschiedenen Belastungen simuliert.

2 Herleitung der Balken-Biege-Differentialgleichung für den statischen und dynamischen Fall

Wie im Vorwort bereits erwähnt, wird die Flugzeugtragfläche als einseitig eingespannter Balken unter einer Streckenlast q modelliert. Im Folgenden wird eine Bewegungsdifferentialgleichung für die Verformung $\omega(x,t)$ in z-Richtung gesucht. Dabei ist auch die Drehung $\psi(x,t)$ um die y-Achse zu beachten. Beide sind in Abbildung 1 (rot) eingeführt worden.

An einem differentiell kleinen Stück werden die *d'Alembertschen* Trägheitskräfte angetragen (grün) und bilanziert:

$$\sum F_{\uparrow} : dm \cdot \ddot{\omega} + Q - (Q + \frac{\partial Q}{\partial x}dx) - q \cdot dx \stackrel{!}{=} 0$$

$$\Leftrightarrow dm\ddot{\omega} = \frac{\partial Q}{\partial x}dx + qdx \tag{1}$$

$$\sum M^{(S)} : -(d\Theta_y \cdot \psi'') - M + (M + \frac{\partial M}{\partial x}dx) - Q \cdot dx \stackrel{!}{=} 0$$

$$\Leftrightarrow d\Theta_y \psi'' = \frac{\partial M}{\partial x}dx - Qdx \tag{2}$$

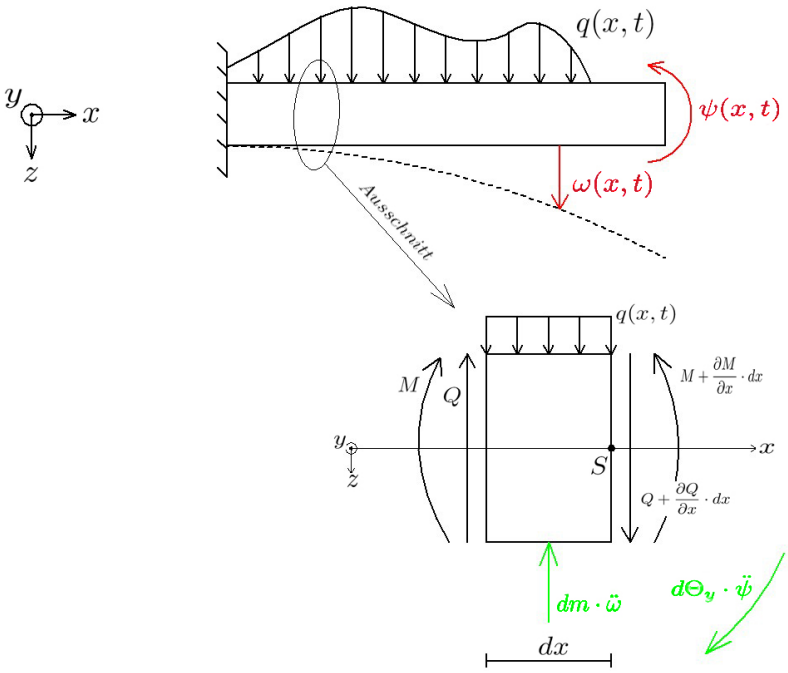


Abbildung 1: Schnittgrößen am differenziell kleinen Ausschnitt

In (1) und (2) können für das differentiell kleine Massenelement $dm = \rho \cdot A \cdot dx$ und das Massenträgheitsmoment $d\Theta_y = \rho \cdot I \cdot dx$ geschrieben werden, wobei ρ für die Dichte, A für die Querschnittsfläche und I für das Flächenträgheitsmoment stehen.

Vereinbart man im Folgenden als Kurzschreibweise $\frac{\partial(...)}{\partial x} = (...)'$ und $\frac{\partial(...)}{\partial t} = \dot{(...)}$, so ergeben sich die Gleichungen (1) und (2) zu:

$$\boxed{\rho A \ddot{\omega} = Q' + q} \tag{3}$$

$$\boxed{\rho I \ddot{\psi} = M' - Q} \tag{4}$$

Im Weiteren werden noch zwei Formeln aus der Geometrie der Spannungen im Balken hergeleitet. In Abbildung 2 werden die Schubspannung τ und die Normalspannung σ eingeführt.

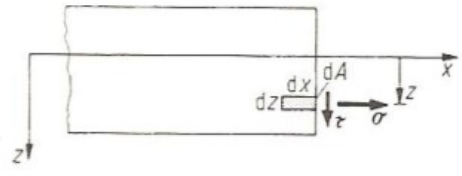


Abbildung 2: Schub- und Normalspannung am Balken

Nach Integration über die gesamte Querschnittsfläche ergibt sich:

$$M = \int_{(A)} z \cdot \underbrace{\sigma \cdot dA}_{dF_N} \tag{5}$$

und

$$Q = \int_{(A)} \tau \cdot dA \tag{6}$$

Setzt man im Folgenden kleine Verdrehungen ψ voraus, so kann, wie in Abbildung 3 angedeutet wird, linearisiert werden.

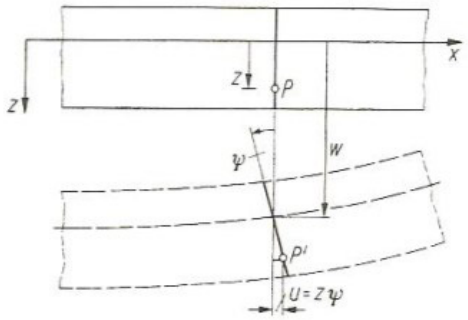


Abbildung 3: Linearisierung für kleine Winkel

Wegen der Taylor-Entwicklung des Sinus mit Abbruch nach dem ersten Glied folgt:

$$u = z \cdot \sin(\psi) \approx z \cdot \psi \tag{7}$$

Aus der Definition des E-Moduls[1] folgt:

$$\sigma = E \cdot \epsilon = E \cdot \frac{\partial u}{\partial x} \overset{(7)}{=} E \cdot \psi' z$$

Eingesetzt in (5) liefert:

$$M = \int z^2 \cdot E \cdot \psi' \cdot dA = E\psi' \cdot \underbrace{\int z^2 dA}_{I}$$

$$\Leftrightarrow \boxed{M = E \cdot \psi' \cdot I} \tag{8}$$

Ferner kann die Definition des G-Moduls in Verbindung mit der Definition für die Gleitung[2] genutzt werden:

$$\tau = G \cdot \gamma_{xz} = G \cdot (\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}) \overset{(7)}{=} G \cdot (w' + \psi)$$

[1] Eingeführt z.B. in Schnell, Gross, Hauger, Technische Mechanik 2, Kapitel 1.3
[2] Ebenda, Kapitel 3.1

Einsetzen in (6) liefert:

$$Q = \int_{(A)} G \cdot (w' + \psi) dA = G(w' + \psi) \cdot \int_A dA$$

$$\Leftrightarrow \boxed{Q = GA(w' + \psi)} \tag{9}$$

Setzt man für den schlanken Balken nun Schubstarrheit voraus ($G \cdot A \to \infty$) und vernachlässigt gleichzeitig die Rotationsträgheit ($\rho \cdot I \to 0$), so ergibt sich aus (3),(4),(8) und (9) folgendes Gleichungssystem:

$$\rho A \ddot{w} = Q' + q$$

$$M' = Q$$

$$EI\psi' = M$$

$$\omega' = -\psi$$

Die Lösung dieses Systems (simples Einsetzen!) führt zu einer partiellen Differentialgleichung, die im Weiteren mit Hilfe von numerischen Methoden gelöst werden soll. Es folgt die Bewegungsdifferentialgleichung für $\omega(x,t)$ in z-Richtung:

$$\boxed{(EI\omega'')'' + \rho A \ddot{w} = q(x,t)}$$

Bemerkung: Es sei an dieser Stelle angemerkt, dass diese Bewegungsdifferentialgleichung auch über die Energiemethoden der Mechanik aufgestellt werden kann. Im Modul *Technische Mechanik 3* lernt man, wie das mit Hilfe des Prinzips von Hamilton geht.

3 Modellierung der Biegelinie des Balkens

3.1 Formulierung des Randwertproblems

Wir suchen, wie zuvor hergeleitet, die Lösung der Differentialgleichung:

$$(EIw'')'' + \rho A \ddot{w} = q.$$

Da der Balken bei x=0 fest eingespannt ist und bei x=L keine Einzelkraft (bzw. Lagerreaktion) wirkt, gelten folgende Randbedingungen:

1.) $w(0) = 0$

2.) $w'(0) = 0$

3.) $M_b(L) = -EIw''(L) = 0$

4.) $Q(L) = -(EIw'')'(L) = 0$

M_b ist dabei das Biegemoment und Q die Querkraft im Balken. Wir werden später noch sehen, dass sich im Rahmen der numerischen Berechnung das Randwertproblem auf ein lineares Differentialgleichungssystem 2. Ordnung zurückführen lässt:

$$M\ddot{u} + Su = p,$$

wobei M Massenmatrix, S Steifigkeitsmatrix und p Lastvektor heißt. Die Randbedingungen fließen direkt in die Einträge von M und S ein! Die Bedeutung des Lösungsvektors u wird im folgenden Abschnitt erläutert.

3.2 Bedeutung des u-Vektors

Zunächst stellt sich die Frage nach der Darstellung der Biegelinie des Balkens. Theoretisch kann die Auslenkung durch eine unendliche Reihe von überlagerten Sinus- und Cosinusfunktionen beschrieben werden. Doch die Numerik tut sich schwer mit der Unendlichkeit, daher muss man sich etwas einfallen lassen, um die Biegelinie zu approximieren.

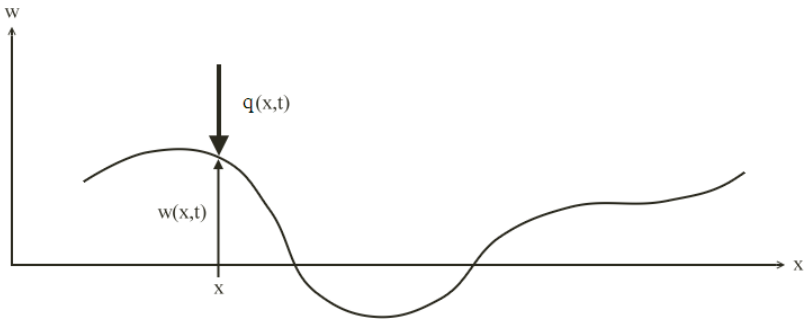


Abbildung 4: Reale Biegelinie

Für die numerische Berechnung wird der Balken in viele Teilintervalle zerlegt, für die jeweils eine eigene Lösungsnäherung der DGL berechnet und geplottet wird. Hat man $n-1$ Intervalle, so ergeben sich n Knotenpunkte x_1 bis x_n. Als Ansatz für die stückweisen Näherungsfunktionen werden Polynome 3. Grades gewählt. Dies hat verschiedene Vorteile:

- Polynome 3. Grades können intervallweise auch stark gekrümmte Funktionen approximieren und sind bei fester Intervalllänge genauer als Polynome 1. oder 2. Grades.
- Polynome 3. Grades sind bereits durch vier Parameter eindeutig bestimmt. Wir werden pro Polynom die Auslenkung und die Steigung an den Intervallgrenzen (Knoten) als Informationen verwenden.

- Polynome sind unendlich oft differenzierbar.

Die zur eindeutigen Festlegung der Polynome benötigten Informationen werden in einem Vektor u gespeichert und zwar derart, dass für alle Knoten nacheinander immer Auslenkung und Steigung abwechselnd vorkommen:

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{2k-1} \\ u_{2k} \\ \vdots \end{pmatrix} \begin{array}{l} \\ \\ \\ \longleftarrow \text{ Auslenkung am Knoten } x_k \\ \longleftarrow \text{ Steigung am Knoten } x_k \\ \\ \end{array}$$

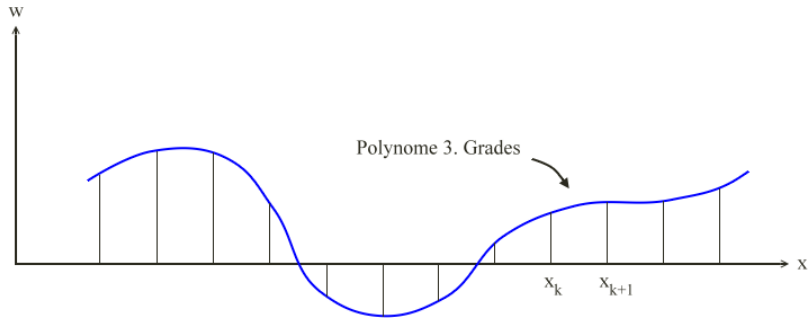
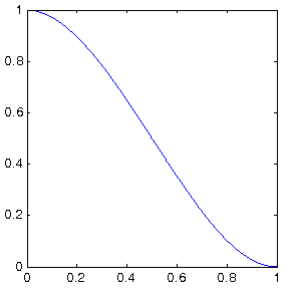


Abbildung 5: Approximierte Biegelinie

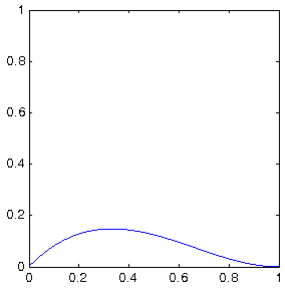
Damit die Biegelinie keine Sprünge oder Knicke enthält, gelten die Informationen an einem Knoten für beide an diesen Knoten grenzende Polynome!

3.3 Basispolynome

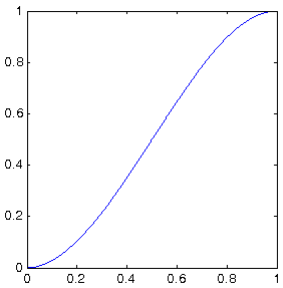
Man kann zeigen, dass φ_1 bis φ_4 eine Basis des Raums der Polynome vom Grad ≤ 3 bilden. Das bedeutet, dass all unsere stückweisen Polynome 3. Grades sich als Linearkombination folgender Basispolynome darstellen lassen (Die Auslenkung bzw. Steigung an den Intervallgrenzen ist immer 1 oder 0):



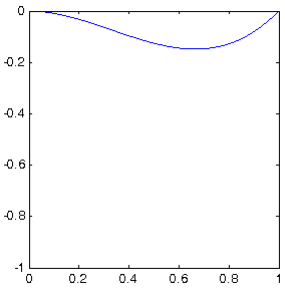
$\varphi_1(x) = 2x^3 - 3x^2 + 1$



$\varphi_2(x) = x^3 - 2x^2 + x$



$\varphi_3(x) = -2x^3 + 3x^2$



$\varphi_4(x) = x^3 - x^2$

Da die Basispolynome nur für Intervalle [0,1] gelten, muss man für die Darstellung der skalierten Polynome das Argument auf die Intervalllänge $l = \frac{L}{n-1}$ normieren:

- $\tilde{\varphi_1}(x) = 2(x/l)^3 - 3(x/l)^2 + 1$
- $\tilde{\varphi_2}(x) = (x/l)^3 - 2(x/l)^2 + (x/l)$
- $\tilde{\varphi_3}(x) = -2(x/l)^3 + 3(x/l)^2$
- $\tilde{\varphi_4}(x) = (x/l)^3 - (x/l)^2$

Wir werden später auf die Basispolynome zurückgreifen.

3.4 Schwache Formulierung des Randwertproblems

Wir wollen die Lösung stückweise durch Näherungsfunktionen $u(x,t)$ finden. Den Raum der Ansatzfunktionen hat man daher gegeben mit

$A = \{u : [0, L] \to \mathbb{R} \mid$ Auf jedem der Intervalle $[x_k, x_{k+1}]$ ist u Polynom höchstens 3. Grades und u' exisitert für alle $x \in [0, L]\}$,
wobei $0 = x_1 < x_2 < \ldots < x_n = L$.

Desweiteren definieren wir:

$D^1 := \{f : [a,b] \to \mathbb{R} \mid \text{f ist stetig, es gibt } a = x_1 < x_2 < \ldots < x_n = b, \text{ so dass f auf jedem Intervall } [x_k, x_{k+1}] \text{ differenzierbar ist.}\}$

$D^2 := \{f : [a,b] \to \mathbb{R} \mid f' \in D^1\}$

Satz $w : [0,L] \to \mathbb{R}$ ist Lösung des Randwertproblems (RWP), genau dann wenn für alle $\psi \in D^2$ mit $\psi(0) = \psi'(0) = 0$ gilt, dass

$$\int_0^L \rho A \ddot{w}\psi dx + \int_0^L (EIw'')\psi'' dx = \int_0^L q\psi dx.$$

Das ist die *schwache Formulierung* des Randwertproblems. Sie stellt eine *Testgleichung* zum Testen von Lösungen dar. ψ heißt *Testfunktion*. Wir werden diese Gleichung nutzen, um zunächst die Einträge der Steifigkeitssmatrix S zu bestimmen!

Beweis für eine Richtung Sei w Lösung des RWP, dann folgt für alle ψ wie oben definiert durch partielle Integration:

$$0 = \int_0^L \underbrace{\left((EIw'')'' + \rho A\ddot{w} - q\right)}_{=0} \psi dx$$

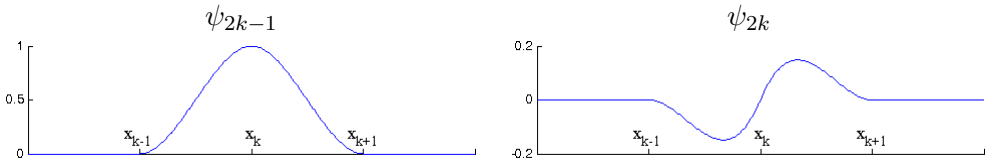
$$= \int_0^L \rho A\ddot{w}\psi dx + \left[(EIw'')'\psi\right]_0^L - \int_0^L (EIw'')'\psi' dx - \int_0^L q\psi dx$$

$$= \int_0^L \rho A\ddot{w}\psi dx + \left[(EIw'')'\psi\right]_0^L - \left[(EIw'')\psi'\right]_0^L + \int_0^L (EIw'')\psi'' dx - \int_0^L q\psi dx$$

Durch Einsetzen der Randbedingungen $\psi(0) = \psi'(0) = 0$ und $EIw''(L) = (EIw'')'(L) = 0$ werden zwei Terme zu Null und die Behauptung des Satzes ist sofort erfüllt!

3.5 Basisfunktionen

Alle Funktionen aus dem Ansatzraum A sind Linearkombinationen der folgenden Basisfunktionen:



Die Funktion ψ_{2k-1} besitzt in allen Knoten den Wert und die Steigung Null, nur an dem Knoten x_k besitzt diese Funktion den Wert 1.
Zudem besitzt die Funktion ψ_{2k} in allen Knoten den Wert und die Steigung Null, außer im Knoten x_k die Steigung 1.
Mit Hilfe dieser Vorüberlegung ist offensichtlich, dass sich die Biegelinie w in folgender Weise durch die Basisfunktionen approximieren lässt.

$$w = \sum_{k=1}^{n} (u_{2k-1}\psi_{2k-1} + u_{2k}\psi_{2k}) = \sum_{l=1}^{2n} u_l \psi_l \qquad (*)$$

wobei n die Anzahl der Knoten ist, u_{2k-1} den Wert und u_{2k} die Steigung im Knoten x_k angibt.

Verständnishinweis: Setzt man nacheinander alle Einträge im u-Vektor außer einem zu Null (Standardbasis), so entstehen die oben dargestellten Basisfunktionen. Daraus folgt sofort, dass jede mögliche Biegelinie eine Linearkombination dieser Funktionen ist, weil jeder u-Vektor eine Linearkombination seiner Standardbasis ist!

3.6 Herleitung des Differentialgleichungssystems

Diese ψ wählen wir als Testfunktionen in unserer schwachen Formulierung. Man könnte auch alle anderen Funktionen nehmen, aber dann kann das Integral nicht allgemein gelöst werden! Die Basisfunktionen selbst als Testfunktionen zu nehmen ist hinsichtlich unserer erwarteten Genauigkeit der Lösung hinreichend. Wir äußern folgende **Vermutung**: Wenn die obige Ansatzfunktion u die Bedingung $u_1 = u_2 = 0$ (Randbed.) und

$$\int_0^L \rho A \ddot{w} \psi_j dx + \int_0^L (EIw'')\psi_j'' dx = \int_0^L q\psi_j dx \qquad (**)$$

für alle $j = 1, \ldots, 2n$ erfüllt, dann ist u die beste Näherung im Ansatzraum!

Setzt man $(*)$ in $(**)$ ein, so erhält man

$$\sum_{l=1}^{2n} \underbrace{\left(\int_0^L \rho A \psi_l \psi_j dx \right)}_{M_{jl}} \ddot{u}_l + \sum_{l=1}^{2n} \underbrace{\left(\int_0^L EI \psi_l'' \psi_j'' dx \right)}_{S_{jl}} u_l = \underbrace{\int_0^L q\psi_j dx}_{p_j}$$

Für jedes j entsprechen die Summen einer Multiplikation der j-ten Zeile von einer Matrix mit unserem Vektor u bzw. $\ddot{u}$. Die Matrix M, die sich aus der ersten Summe ergibt nennen wir im Folgenden Massematrix und die Matrix S, die sich aus der zweiten Summe ergibt nennen wir im Folgenden Steifigkeitsmatrix. Folglich haben S und M j Zeilen und l Spalten.

Deshalb kann unsere partielle Differentialgleichung wie versprochen auch als gewöhnliches DGL-System zweiter Ordnung aufgefasst werden: $M\ddot{u} + Su = p$

$$\begin{bmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,2n} \\ M_{2,1} & M_{2,2} & & \vdots \\ \vdots & & \ddots & \\ M_{2n,1} & \cdots & & M_{2n,2n} \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \vdots \\ \ddot{u}_{2n} \end{bmatrix} + \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,2n} \\ S_{2,1} & S_{2,2} & & \vdots \\ \vdots & & \ddots & \\ S_{2n,1} & \cdots & & S_{2n,2n} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{2n} \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{2n} \end{bmatrix}$$

Der Lösungsvektor u dieses DGL-Systems beschreibt die Biegelinie des belasteten Balkens. Zur Lösung stehen verschiedene Algorithmen zur Verfügung, die in Kapitel 5 erläutert werden.

3.6.1 Berechnung der Einträge der Matrizen

Betrachten wir nun zunächst die Einträge der Steifigkeitsmatrix; die der Massematrix lassen sich dann analog bestimmen. Um die Einträge von S konkret zu berechnen, überlegen wir uns, wie die Integrale $\int_0^L EI\psi_l''\psi_j''dx$ aussehen.
Das Integral wird nur dann nicht Null, wenn das Produkt $\psi_j''\psi_l''$ nicht Null wird. Aber alle ψ (und damit auch ihre Ableitungen) sind nur auf zwei Intervallen von Null verschieden!

Das bedeutet, es gibt drei Fälle zu unterscheiden:

ψ_l und ψ_j sind beide von Null verschieden auf

1) keinem Intervall $\implies$ $\int(\ldots) = 0$

2) einem Intervall $\implies$ $\int(\ldots) \neq 0$

3) zwei Intervallen $\implies$ $\int(\ldots) \neq 0$

Die folgenden Bilder verdeutlichen die Beziehungen:

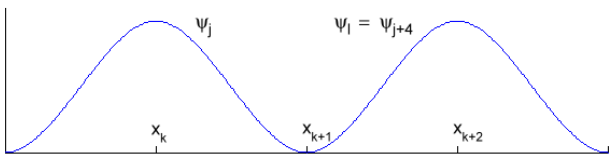


Abbildung 6: kein Intervall

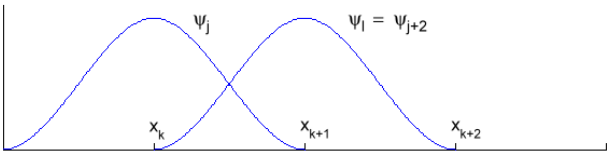


Abbildung 7: ein Intervall

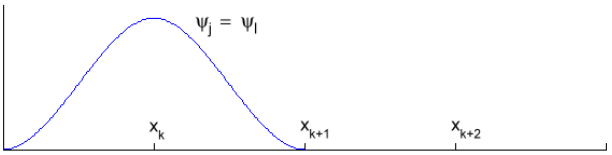
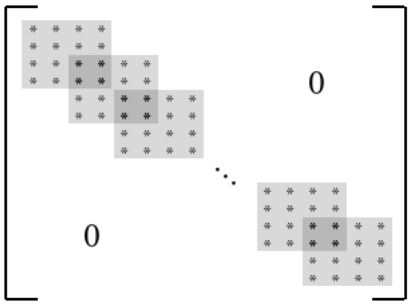


Abbildung 8: zwei Intervalle

Nun kommen wir wieder zurück auf die Basispolynome! Da die ψ's intervallweise zusammengesetzt sind aus φ's, berechnet man auch die Integrale intervallweise mit den entsprechenden φ's.
Wie in der Abbildung zu erkennen ist, gilt: Auf jedem Intervall gibt es lediglich 16 mögliche Integrale, denn es gibt nur vier Basispolynome, die in allen Varianten miteinander im Integral vorkommen können: $\int EI\varphi_j''\varphi_l''dx$ mit $j,l = 1,2,3,4$
Diese 16 Integrale bilden die sogenannte Elementsteifigkeitsmatrix S_E, die alle Varianten an Integralen enthält, die nur über *ein* Intervall gehen:

$$S_E = EI \begin{bmatrix} \int_0^l \varphi_1''\varphi_1'' & \int_0^l \varphi_1''\varphi_2'' & \int_0^l \varphi_1''\varphi_3'' & \int_0^l \varphi_1''\varphi_4'' \\ \int_0^l \varphi_2''\varphi_1'' & \int_0^l \varphi_2''\varphi_2'' & \int_0^l \varphi_2''\varphi_3'' & \int_0^l \varphi_2''\varphi_4'' \\ \int_0^l \varphi_3''\varphi_1'' & \int_0^l \varphi_3''\varphi_2'' & \int_0^l \varphi_3''\varphi_3'' & \int_0^l \varphi_3''\varphi_4'' \\ \int_0^l \varphi_4''\varphi_1'' & \int_0^l \varphi_4''\varphi_2'' & \int_0^l \varphi_4''\varphi_3'' & \int_0^l \varphi_4''\varphi_4'' \end{bmatrix} = \frac{EI}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & 12l & 4l^2 \\ -12 & -6l & 12 & 6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix}$$

Dabei ist $l = \frac{L}{n-1}$ die Länge der einzelnen Intervalle, daher verläuft die Integration jeweils über ein Intervall.
Die Systemsteifigkeitsmatrix erhält man nun durch mehrfache Überlagerung der Elementsteifigkeitsmatrizen in folgender Weise:



Dieser Sachverhalt leuchtet wahrscheinlich nicht unmittelbar ein, daher schauen wir uns die entstandene Matrix genauer an. Zum Verständnis empfiehlt es sich, parallel zu den folgenden Überlegungen, die einzelnen ψ Funktionen aufzuzeichnen und sich zu überlegen, welche φ Funktionen man miteinander multipliziert. Als Beispiel betrachten wir den Eintrag (3,4) auf der Nebendiagonalen:

$$\int_0^L \psi_3'' \psi_4'' dx = \int_0^l \varphi_3'' \varphi_4'' dx + \int_0^l \varphi_1'' \varphi_2'' dx$$

Welches genau dem entspricht, was man durch Überlagerung erhält. Zunächst erkennt man sofort, dass für $|j-l| \geq 4$ die Einträge in der Matrix Null sind. Die Einträge auf der Nebendiagonalen und der Hauptdiagonalen gehen über zwei Integrale. Dies wird erreicht, da die Einträge dort von zwei Elementsteifigkeitsmatrizen aufgrund der Überlagerung aufsummiert werden. Die Einträge weiter außen stimmen offensichtlich auch. Betrachtet man nun die Einträge für die Randpunkte x_0 und x_n, so stellt man fest, dass es hier zu keiner Überlagerung kommt. Dies ist so auch richtig, da wir auch hier nur über ein Intervall integrieren, weil wir den Gesamtintegrationsbereich (von 0 bis L) nicht verlassen dürfen.

Als Elementmassenmatrix erhält man analog:

$$M_E = \rho A \begin{bmatrix} \int_0^l \varphi_1''\varphi_1'' & \int_0^l \varphi_1''\varphi_2'' & \int_0^l \varphi_1''\varphi_3'' & \int_0^l \varphi_1''\varphi_4'' \\ \int_0^l \varphi_2''\varphi_1'' & \int_0^l \varphi_2''\varphi_2'' & \int_0^l \varphi_2''\varphi_3'' & \int_0^l \varphi_2''\varphi_4'' \\ \int_0^l \varphi_3''\varphi_1'' & \int_0^l \varphi_3''\varphi_2'' & \int_0^l \varphi_3''\varphi_3'' & \int_0^l \varphi_3''\varphi_4'' \\ \int_0^l \varphi_4''\varphi_1'' & \int_0^l \varphi_4''\varphi_2'' & \int_0^l \varphi_4''\varphi_3'' & \int_0^l \varphi_4''\varphi_4'' \end{bmatrix} = \frac{\rho A l}{420} \begin{bmatrix} 156 & 22l & 54 & -13l \\ 22l & 4l^2 & 13l & -3l^2 \\ 54 & 13l & 156 & -22l \\ -13l & -3l^2 & -22l & 4l^2 \end{bmatrix}$$

3.6.2 Einbauen der Randbedingungen

Unser Balken ist auf der linken Seite fest eingespannt. Das bedeutet, dass am Knoten x_1 keine Auslenkung und keine Steigung existieren darf. Demenstsprehend darf hier auch keine Beschleunigung existieren. Somit müssen die ersten beiden Einträge im u-Vektor Null sein. Man könnte u im Nachhinein modifizieren; eleganter ist jedoch folgendes:
Wir setzen alle Einträge in der ersten und zweiten Spalte und Zeile von S und M zu 0, außer den Diagonalelementen, diese werden zu 1 gesetzt. Außerdem werden die ersten beiden Einträge des Lastvektors zu 0 gesetzt. Der Grund ist leicht ersichtlich:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & & & \\ \vdots & & & \# & \\ 0 & 0 & & & \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \\ \vdots \\ u_{2n} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & & & \\ \vdots & & & \# & \\ 0 & 0 & & & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{2n} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ p_3 \\ \vdots \\ p_{2n} \end{bmatrix}$$

Die Werte von u_1 und u_2 ergeben sich sofort zu 0. Auf analoge Weise kann man auch andere Arten von Randbedingungen einbauen, worauf wir hier aber nicht eingehen wollen.

3.7 Lastvektor

Wir kennen nun S und M, aber über die Einträge von p haben wir noch keine Aussage gemacht. Wie wir oben bereits gesehen haben, sind die Einträge definiert durch

$$p_j = \int_0^L q\psi_j dx \qquad \text{mit } j = 1, \ldots, 2n$$

Da die ψ_j nur auf zwei Intervallen von Null verschieden sind, erhalten wir

$$p_{2k} = \int_{x_{k-1}}^{x_{k+1}} q\psi_{2k} dx \qquad p_{2k-1} = \int_{x_{k-1}}^{x_{k+1}} q\psi_{2k-1} dx.$$

Wir nehmen an, dass der Kraftverlauf p, der den Balken belastet, an den Knoten vorgegebene Werte hat, welche in einem Vektor q gespeichert sind, und zwischen den Knoten durch einen linearen Verlauf $q(x)$ approximiert wird:

Dann kann man das Integral für ψ_{2k-1} schreiben als:

$$P_{2k-1} = \int_{x_{k-1}}^{x_{k+1}} \psi_{2k-1}(x)q(x)dx$$

$$= \int_{x_{k-1}}^{x_k} \psi_{2k-1}(x)q(x)dx + \int_{x_k}^{x_{k+1}} \psi_{2k-1}(x)q(x)dx$$

$$= \int_{x_{k-1}}^{x_k} \varphi_3(x) \left(q_{k-1} + \frac{q_k - q_{k-1}}{l} x \right) dx + \int_{x_k}^{x_{k+1}} \varphi_1(x) \left(q_k + \frac{q_{k+1} - q_k}{l} x \right) dx$$

$$= \ldots = \frac{7}{20} q_{k-1} l + \frac{3}{20} q_k l + \frac{7}{20} q_{k+1} l$$

Und analog erhält man für ψ_{2k}:

$$P_{2k} = \int_{x_{k-1}}^{x_{k+1}} \psi_{2k}(x)q(x)dx$$

$$= \ldots = \int_{x_{k-1}}^{x_k} \varphi_4(x) \left(q_{k-1} + \frac{q_k - q_{k-1}}{l} x \right) dx + \int_{x_k}^{x_{k+1}} \varphi_2(x) \left(q_k + \frac{q_{k+1} - q_k}{l} x \right) dx$$

$$= \frac{1}{20} q_{k-1} l + - \frac{1}{20} q_{k+1} l$$

An den Rändern darf aus oben genannten Gründen jeweils nur über ein Intervall integriert werden und es ergibt sich:

$$p_1 = \frac{3}{20} q_1 l + \frac{7}{20} q_2 l$$

$$p_2 = -\frac{1}{30} q_1 l - \frac{1}{20} q_2 l$$

$$p_{n-1} = \frac{7}{20} q_{n-1} l + \frac{3}{20} q_n l$$

$$p_n = \frac{1}{20} q_{n-1} l + \frac{1}{30} q_n l$$

4 Statischer Fall

Im statischen Fall findet keine Schwingung statt, der Balken ist in Ruhe. Um diesen Zustand in der Realität zu erreichen, müssen die Kräfte theoretisch unendlich langsam aufgebracht werden. Da alle zeitlichen Ableitungen der Auslenkung immer Null sind ($\dot{u} = \ddot{u} = 0$), vereinfacht sich „unsere“ PDGL zu $(EI\omega'')'' = q$ und das zugehörige DGL-System $M\ddot{u} + Su = p$ vereinfacht sich zu einem linearen Gleichungssystem $Su = p$. Die Lösung erfolgt z.B. mit den im Anhang vorgestellten Algorithmen. Die unter einer Last entstehende Auslenkung kann als *statische Ruhelage* des Balkens während einer Schwingung unter derselben Last interpretiert werden. Dazu mehr im nächsten Abschnitt.

5 Dynamischer Fall

Die Lösung des Anfangswertproblems $M\ddot{u} + Su = p$, $u(0) = u_0$, $\dot{u}(0) = \dot{u}_0$ ist zum Beispiel mit dem Störmer-Verlet-Verfahren oder der Eigenwertmethode möglich. Beide Verfahren werden im Folgenden näher erläutert.

5.1 Anwendung des Störmer-Verlet-Verfahrens

Zunächst einmal formen wir das Differentiagleichungssystem um zu:

$$M\ddot{u} = \underbrace{p - Su}_{f}$$

Bei der numerischen Lösung eines Differentialgleichungssystems mit Hilfe des Störmer-Verlet-Verfahren ersetzt man die Funktionen $u(t)$,$\dot{u}(t)$ und $f(t)$ durch Folgen von Näherungswerten u_k, $\dot{u}_k$ und f_k, welche für bestimmte Zeitpunkte t_k berechnet werden.
Zudem definieren wir $h_k := t_{k+1} - t_k$ als Zeitschrittweite im k-ten Schritt. In unserem Fall blieb dieser Wert konstant.
Zunächst berechnet man eine Näherung für die Geschwindigkeit nach einem halben Zeitschritt:

$$\tilde{v} = v_k + \frac{h_k}{2} \underbrace{M^{-1} f_k}_{\ddot{u}_k}$$

Damit lässt sich nun die Auslenkung zum nächsten Zeitpunkt berechnen:

$$x_{k+1} = x_k + h_k \tilde{v}$$

Daraus wiederum die wirkende Kraft zum nächsten Zeitpunkt:

$$f_{k+1} = p - S x_{k+1}$$

Woraus sich die Geschwindigkeit zum nächsten Zeitpunkt berechnen lässt:

$$v_{k+1} = \tilde{v} + \frac{h_k}{2} \underbrace{M^{-1} f_{k+1}}_{\ddot{u}_{k+1}}$$

Nun kann man das Ganze wiederholen und so die Schwingung des Balkens über die gesamte Zeitdauer berechnen. Als kleine Anmerkung sei hier noch gesagt, dass man die Multiplikation mit M^{-1} nicht wirklich ausführt, sondern stattdessen die linearen Gleichungssysteme

$$M\tilde{v} = Mv_k + \frac{h_k}{2} f_k$$

$$Mv_{k+1} = M\tilde{v} + \frac{h_k}{2} f_{k+1}$$

numerisch löst; in unserem Fall durch das Cholesky-Verfahren (siehe Anhang).

5.2 Anwendung der Eigenwertmethode

Zunächst einmal betrachten wir die Schwingung des Balkens als eine Schwingung um die statische Ruhelage $u_{stat} = S^{-1}p$, daher:

$$u = \widetilde{u} + u_{stat}$$

Dies hat den enormen Vorteil, dass sich dabei die Inhomgenität p rauskürzen lässt und aus unserem inhomogenen Differentialgleichungssystem ein homogenes Differentialgleichungssystem wird:

$$M\ddot{\widetilde{u}} + S\widetilde{u} = 0;$$

Dieses lineare homogene Differentialgleichungssystem können wir nun einfach mit Hilfe des Exponentialansatz $\widetilde{u} = Cve^{i\omega t}$ lösen. Hierbei ist $C \in \mathbb{C}$ und $v \in \mathbb{R}^n$. Durch Einsetzen in das Differentialgleichungssystem ergibt sich:

$$-M\omega^2 Ce^{i\omega t}v + SCe^{i\omega t}v = 0$$

$$\Leftrightarrow (M^{-1}S - \omega^2 E_n)vCe^{i\omega t} = 0$$

$$\Leftrightarrow (M^{-1}S - \omega^2 E_n)v = 0$$

Dies ist nun gerade eine Eigenwertgleichung. Durch Berechnen der Eigenwerte und Eigenvektoren und unter Nutzung der Eulerrelation erhält man als allgemeine Lösung:

$$\widetilde{u} = \sum_{k=1}^{n} (C_{-k} e^{-\omega_k t} v_k + C_k e^{\omega_k t} v_k)$$

$$\widetilde{u} = \sum_{k=1}^{n} (A_k cos(\omega_k t) v_k + B_k sin(\omega_k t) v_k)$$

Somit ist die allgemeine Lösung unseres Anfangswertproblems:

$$u(t) = \sum_{k=1}^{n} (A_k cos(\omega_k t) v_k + B_k sin(\omega_k t) v_k) + u_{stat}$$

Zur fertigen Lösung fehlen nun nur noch die Koeffizienten, welche wir durch die Anfangswerte erhalten.
$u(0) = u_0$:

$$u_0 - u_{stat} = \sum_{k=1}^{n} (A_k v_k) = (v_1, \ldots, v_n) \cdot A$$

Daher sind die A_k in dem Lösungsvektor A des LGS enthalten.
$\dot{u}(0) = \dot{u}_0$:

$$\dot{u}_0 = \sum_{k=1}^{n} (B_k \omega_k v_k) = (v_1, \ldots, v_n) \cdot B$$

Der Lösungsvektor B des LGS enthält als Elemente die $B_k \omega_k$, daher erhält man durch Division mit ω_k jeweils die B_k. Damit haben wir das Anfangswertproblem also eindeutig gelöst.

5.3 Vergleich Eigenwertmethode mit Störmer-Verlet-Verfahren

Zunächst einmal stellten wir im direkten Vergleich zwischen Eigenwertmethode und Störmer-Verlet-Verfahren fest, dass sich die Ergebnisse von beiden kaum unterscheiden; beide Verfahren sind in ihrer Genauigkeit in etwa gleich gut.
Allerdings hat das Störmer-Verlet-Verfahren den starken Nachteil, dass es nicht besonders stabil ist:

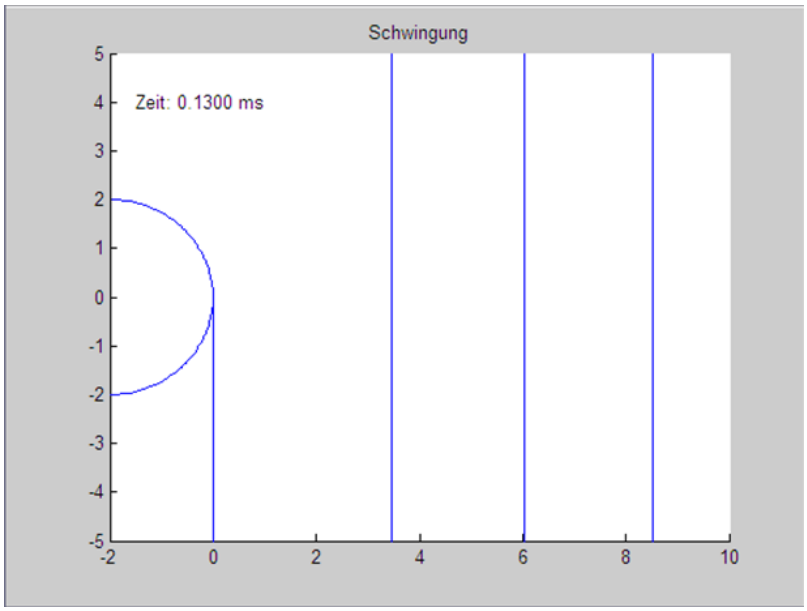


Abbildung 9: Fehler beim Störmer-Verlet-Verfahren

So sah die Lösung der Differentialgleichung aus, wenn die Parameter nicht richtig abgestimmt waren, insbesondere wenn man den Zeitschritt zu groß gewählt hat.
Dies ist der Grund, weshalb wir uns beim Fachwerk dazu entschieden haben, nur die Eigenwertmethode zu verwenden. Das Abstimmen der Parameter konnte zum Teil recht mühselig und zeitaufwendig werden und machte aus unserer Sicht wenig Sinn in unserem Fachwerkeditor.

6 Erweiterung des Projektes auf Fachwerke

6.1 Programmstruktur

Wie bereits angedeutet, konnten wir im Verlauf des Projektes unsere Simulation von der Berechnung eines einzelnen Balkens auf ein Fachwerk aus mehreren Balken *der gleichen Mathematik* erweitern.
Aus Sicht der Mechanik ist sofort klar, dass jetzt Übergangsbedingungen zwischen zwei Balken modelliert werden müssen und auch die verschiedenen Lagerungen berücksichtigt werden müssen.
Da der mathematische Hintergrund bereits ausführlich in Michael Karows „Notizen zum Fachwerkprojekt“[3] erklärt wird, wollen wir uns im Folgenden auf die Erläuterung unserer programmiertechnischen Lösung beschränken.

[3] Download unter http://www.math.tu-berlin.de/ppm/projekt_praktische_mathematik/materialien.php

Die Grundidee ist die Zwangsbedingungen, die in der Matrix C gespeichert sind, automatisch beim Erstellen des Fachwerkes im Hintergrund zu erzeugen. Dazu müssen allerdings Informationen über die Balken, die Lagerung und die Bindungen bekannt sein.
Zunächst erzeugt der Benutzer also einzelne Balken, indem er deren Koordinaten und Stoffeigenschaften angibt. Das Programm speichert diese Informationen zeilenweise in einer Matrix, die wir B genannt haben:

$$\begin{pmatrix} x_{an,1} & y_{an,1} & x_{en,1} & y_{en,1} & l_1 & \varphi_1 & EI_1 & EA_1 & \mu_1 & n_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{an,i} & y_{an,i} & x_{en,i} & y_{en,i} & l_i & \varphi_i & EI_i & EA_i & \mu_i & n_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{an,n} & y_{an,} & x_{en,n} & y_{en,n} & l_n & \varphi_n & EI_n & EA_n & \mu_n & n_n \end{pmatrix} \leftarrow \text{eine Zeile pro Balken}$$

Mit Erstellen der Balken, wurden auch immer gleichzeitig so genannte *Lastrohlinge* der entsprechenden Genauigkeit erzeugt. Diese können nun überschrieben werden, so dass das System äußeren Belastungen ausgesetzt werden kann.

Nun werden die Zwangsbedingungen in Form von Übergangsbedingungen und Lagerreaktionen eingegeben. Auch diese Informationen werden in Matrizen gespeichert. Die Matrix Z enthält die Information, welche Art der Bindung zwischen zwei Balken vorliegt:

$$\begin{pmatrix} Balken1_1 & Knoten1_1 & Balken2_1 & Knoten2_1 & Art_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Balken1_i & Knoten1_i & Balken2_i & Knoten2_i & Art_i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Balken1_n & Knoten1_n & Balken2_n & Knoten2_n & Art_n \end{pmatrix} \leftarrow \text{eine Zeile pro Bindung}$$

Analog wird in der Matrix A gespeichert, wo sich welche Form der Lagerung befindet:

$$\begin{pmatrix} Balken_1 & Knoten_1 & Art_1 & Winkel_1 \\ \vdots & \vdots & \vdots & \vdots \\ Balken_i & Knoten_i & Art_i & Winkel_i \\ \vdots & \vdots & \vdots & \vdots \\ Balken_n & Knoten_n & Art_n & Winkel_n \end{pmatrix} \leftarrow \text{eine Zeile pro Lagerung}$$

Aus diesen Informationen kann man mit mechanischen Methoden alle Zwangsbedingungen formulieren und diese spaltenweise in die Matrix C eintragen. Da auch die Stoffeigenschaften jedes Balkens bekannt sind, kann auch die Massen-

und Steifigkeitsmatrix des Fachwerkes aus m Balken sukzessiv aufgestellt werden:

$$M = \begin{pmatrix} M_L^{(1)} & & & & & \\ & M_T^{(1)} & & & & \\ & & M_L^{(2)} & & & \\ & & & M_T^{(2)} & & \\ & & & & \ddots & \\ & & & & & M_T^{(m)} \end{pmatrix}, S = \begin{pmatrix} S_L^{(1)} & & & & & \\ & S_T^{(1)} & & & & \\ & & S_L^{(2)} & & & \\ & & & S_T^{(2)} & & \\ & & & & \ddots & \\ & & & & & S_T^{(m)} \end{pmatrix}$$

Es entsteht wiederum ein LGS, ganz analog zum einzelnen Balken, welches mit denselben numerischen Methoden gelöst werden kann. Wir haben die Cholesky-Methode angewandt, sowie mit Hilfe der Potenzmethode die Eigenwerte für die analytische Lösung der Schwingungsdifferentialgleichung berechnet. Es sei darauf hingewiesen, dass bei komplexeren Fachwerken das Berechnen der Eigenwerte nach dem von uns programmierten Algorithmus sehr aufwändig ist. Es wird daher empfohlen, bei größeren Matrizen die Matlab-Routine an dieser Stelle zu verwenden.

6.2 Bedienungsanleitung

Dieser Abschnitt soll dem Benutzer erleichtern unser Programm anzuwenden und ein beliebiges Fachwerk mit Hilfe des Editors zu entwerfen.
Dabei ist immer folgende Reihenfolge einzuhalten:

- Start eines neuen Fachwerkes durch Eingabe **start_projekt**. Es werden alle zuvor beschriebenen Matrizen, die im Hintergrund die Informationen des Fachwerks speichern, auf Null gesetzt.
- **Balken erstellen:** Hierbei ist die Reihenfolge der zu übergebenden Variablen wichtig, und zwar: balken_erstellen($x_{an}, y_{an}, x_{en}, y_{en}, EI, EA, \mu, n$). Jeder Balken wird im Editor dargestellt, durchnummeriert sowie in Klammern die Anzahl der Knotenpunkte n angegeben.
- **Bindungen erstellen:** Zwischen zwei Balken müssen Bindungen gesetzt werden. Die Balken werden hierfür über ihre Nummer angesprochen sowie angegeben, ob die Bindung am Balkenanfang (Knoten=0) oder Balkenende (Knoten=1) ist. Ferner kann zwischen einer gelenkigen (Art=1) oder biegesteifen (Art=2) Bindung unterschieden werden. Es ist zu beachten, dass keine linear abhängigen Befehle eingegeben werden.
  Eingabe: bindung_erstellen(Balken1,Knoten1,Balken2,Knoten2,Art)
- **Lagerung erstellen:** Wie beim Erstellen der Bindungen kann über die Nummer des Balkens und die Knotenposition ein Lager an eine bestimmte

Stelle plaziert werden. Unterscheiden wird zwischen einer festen Einspannung (Art=1), einem Festlager (Art=2) und einem Loslager (Art=3). Ferner kann der Winkel der Lagerung eingestellt werden. Aufgerufen wird die Funktion wie folgt: lager_erstellen(Balken, Knoten, Art, Winkel). Lagerungen und Bindungen können über die entsprechenden Funktionen wieder gelöscht werden.

- Aufruf der Prozedur **C_erstellen**. Die Rand- und Übergangsbedingungen werden in der Matrix C gespeichert.
- Aufruf der Prozedur **Matrizen_erstellen**. Es werden die Massen- und Steifigkeitsmatrix des Fachwerks erstellt.
- Nach Erstellen des Fachwerks können **Anfangsauslenkungen** auf die einzelnen Balken aufgebracht werden. Es wird dann sofort die neue statische Ruhelage des Systems berechnet und geplottet. Beim Aufruf der Funktion Anfangslast_aendern(Balken, Vektor, Art) ist zu beachten, dass der Vektor die richtige Dimension, entsprechend der Anzahl der Knoten des Balkens, hat. Es kann zwischen transversalen (Art=1) und longitudinalen (Art=2) Auslenkungen unterschieden werden.
- Zusätzlich können durch die Funktion Last_aendern(Balken, Vektor, Art) dauerhafte **Einzel- oder Streckenlasten** während der Schwingung modelliert werden.
- Aufruf der Prozedur **simulation**.

A Lösungsmethoden für lineare Gleichungssysteme

Da wie oben erwähnt lineare Gleichungssysteme gelöst werden müssen, werden im Folgenden einige effiziente Lösungsalgorithmen vorgestellt.

A.1 LR-Zerlegung

A.1.1 Allgemeines zur LR-Zerlegung

Die LR-Zerlegung ist ein sehr intuitiver Lösungsalgorithmus, der stark auf dem Gaußalgorithmus aufbaut. Hierbei wird die Matirx A, welche unser Gleichungssystem darstellt, wie folgt zerlegt:

$$A = L \cdot R,$$

wobei es sich bei der Matrix L um eine linke untere Dreiecksmatrix handelt:

$$L = \begin{pmatrix} l_{11} & & & 0 \\ & l_{22} & & \\ & & \ddots & \\ * & & & l_{nn} \end{pmatrix}$$

und bei der Matrix R um eine rechte obere Dreiecksmatrix:

$$R = \begin{pmatrix} r_{11} & & & * \\ & r_{22} & & \\ & & \ddots & \\ 0 & & & r_{nn} \end{pmatrix}$$

Betrachtet man nun das Gleichungssystem mit der LR-Zerlegung:

$$A\vec{x} = \vec{b}$$

$$\Rightarrow (L \cdot R)\vec{x} = \vec{b}$$

so entstehen zwei elementare Gleichungssysteme:

$$R\vec{x} =: \vec{y}$$

$$L\vec{y} = \vec{b}$$

Die Lösung dieser elementaren Gleichungssysteme nennt man *Vorwärts-* bzw. *Rückwärtseinsetzen.*

A.1.2 Beispiel zur LR-Zerlegung

Am Besten versteht man einen Algorithmus an einem kleinen Beispiel. Hier wird folgendes gewählt:

$$A \cdot \vec{x} = \vec{b}$$

mit

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 8 \end{pmatrix}, \vec{b} = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}$$

Ziel ist die Bestimmung des Lösungsvektors $\vec{x}$.
Durch Umformen der Zeilen (Gaußschritte) wird die Matrix A auf rechte obere Dreiecksform gebracht und dies wird dann als Matrix R definiert. Bei der linken unteren Dreiecksmatrix L wird zunänchst mit einer Einheitsmatrix E gestartet und es werden die Rechenoperationen gespeichert.
Schritt 1: Das Element r_{21} wird zu Null gemacht mit -1/2 * I + II

$$A^* = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 5 & \frac{1}{2} \\ 1 & 1 & 8 \end{pmatrix}, L^* = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Schritt 2: Das Element r_{31} wird zu Null gemacht mit -1/4 * I + III (und Vorzeichenänderung)

$$A^* = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 5 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{31}{4} \end{pmatrix}, L^* = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{pmatrix}$$

Schritt 3: Das Element r_{32} wird zu Null gemacht mit -1/10 * II + III, was dann auch zum endgültigen Ergebnis führt:

$$R = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 5 & \frac{1}{2} \\ 0 & 0 & \frac{77}{10} \end{pmatrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{4} & \frac{1}{10} & 1 \end{pmatrix}$$

Die Teillösung $\vec{y}$ ergibt sich aus der schon erwähnten Formel $L \cdot \vec{y} = \vec{b}$ durch Vorwärtseinsetzen:

$$\vec{y} = \begin{pmatrix} 2 \\ 3 \\ \frac{1}{5} \end{pmatrix}$$

Mit dieser Teillösung kann nun das zweite elementare Gleichungssystem $R \cdot \vec{x} = \vec{y}$ durch Rückwärtseinsetzen gelöst werden:

$$\vec{x} = \frac{1}{77} \begin{pmatrix} 15 \\ 3 \\ \frac{1}{5} \end{pmatrix}$$

A.1.3 Algorithmus der LR-Zerlegung

Matlab-Quellqoute für die reine LR-Zerlegung:

```
function [L,R]=LR_zerlegung(A,n,b)
    R=A;
    L=eye(n);
    for j=1:n
        for i=j+1:n
            if R(j,j)==0
                'Kannst du durch Null teilen??'
                return;
            else
                L(i,j)= R(i,j)/R(j,j);
                R(i,:)=R(i,:)-R(j,:)*L(i,j);
            end
        end
    end
```

Matlab-Quellcode für das Vorwärtseinsetzen:

```
function x=vorwaerts(A,b)
[m,n]=size(A);
k=length(b);
for i=1:1:m
     x(i)=0;
```

```
end
if (m~=k)
     disp('Dimensionfehler !');
     return
end
for i=1:1:m
      c=0;
      for j=1:1:i-1
        c=c+x(j)*A(i,j);
     end
     x(i)=(b(i)-c)/A(i,i);
end
x=x';
```

Matlab-Quellcode für das Rückwärtseinsetzen:

```
function x=rueckwaerts(A,b)
[m,n]=size(A);
k=length(b);
for i=1:1:m
     x(i)=0;
end
if (m~=k)
     disp('Dimensionfehler !');
     return
end
for i=m:-1:1
     c=0;
     for j=1:1:m-i
         c=c+x(m-j+1)*A(i,m-j+1);
    end
    x(i)=(b(i)-c)/A(i,i);
end
x=x';
```

A.2 Cholesky-Verfahren

A.2.1 Allgemeines zum Cholesky-Verfahren

Als Grundidee ist wieder eine Matrixzerlegung in eine rechte obere Dreiecksmatrix und eine linke untere Dreiecksmatrix nötig, wobei die beiden Matrizen eine besondere Eigenschaft haben sollen, und zwar ist R gerade gleich der Transponierten von L. Es gilt demnach für die Zerlegung:

$$A = L \cdot L^T$$

Natürlich kann man sich vorstellen, dass diese spezielle Art der Matrixzerlegung nicht mit jeder beliebigen Matrix möglich ist. Sie funktioniert nur bei positiv definiten Matrizen, wobei die positive Definitheit meist dadurch ermittelt wird, dass man eine Cholesky-Zerlegung versucht und wenn diese erfolgreich ist, so ist auch die Matrix A positiv definit.
Es handelt sich hierbei um eine iterative Zerlegung, d.h. jeder Schritt ist exakt der Bildungsvorschrift zu entnehmen und für jede Matrix A identisch. Zur Verdeutlichung der Bildungsvorschrift wird folgend wieder ein Beispiel herangezogen.

A.2.2 Beispiel zum Cholesky-Verfahren

Beispielhaft soll zunächst eine 3x3-Matrix allgemein zerlegt werden:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \cdot \begin{pmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{pmatrix}$$

Wobei zu beachten ist, dass

$$l_{ij} = l_{ji} \qquad \forall i, j = 1, 2, 3$$

so dass sich hier folgende Vorschrift ergibt:

$$a_{11} = l_{11}^2 \Leftrightarrow l_{11} = \sqrt{a_{11}}$$

$$a_{12} = l_{11} \cdot l_{12} \Leftrightarrow l_{12} = \frac{a_{12}}{l_{11}}$$

$$a_{13} = l_{11} \cdot l_{13} \Leftrightarrow l_{13} = \frac{a_{13}}{l_{11}}$$

$$a_{22} = l_{12}^2 \cdot l_{22}^2 \Leftrightarrow l_{22} = \sqrt{a_{22} - l_{12}^2}$$

$$a_{23} = l_{12} \cdot l_{13} + l_{22} \cdot l_{23} \Leftrightarrow l_{23} = \frac{a_{23} - l_{12} \cdot l_{13}}{l_{22}}$$

$$a_{33} = l_{13}^2 \cdot l_{23}^2 \cdot l_{33}^2 \Leftrightarrow l_{33} = \sqrt{a_{33} - l_{13}^2 - l_{23}^2}$$

An dieser Stelle kann man auch sehr gut erkennen, was *positiv definit* bedeutet, denn hier dürfen die Werte unter der Würzel nicht negativ sein. Jedoch soll die ganze Theorie auch an einem konkreten Beispiel gezeigt werden:

$$A = \begin{pmatrix} 9 & 6 & -3 \\ 6 & 8 & 2 \\ -3 & 2 & 6 \end{pmatrix}$$

1. Schritt:

$$l_{11} = \sqrt{a_{11}} = \sqrt{9} = 3$$

2. Schritt:

$$l_{12} = \frac{a_{12}}{l_{11}} = \frac{6}{3} = 2$$

3. Schritt:

$$l_{13} = \frac{a_{13}}{l_{11}} = \frac{-3}{3} = -1$$

4. Schritt:

$$l_{22} = \sqrt{a_{22} - l_{12}^2} = \sqrt{8 - 2^2} = 2$$

5. Schritt:

$$l_{23} = \frac{a_{23} - l_{12} \cdot l_{13}}{l_{22}} = \frac{2 - 2 \cdot (-1)}{2} = 2$$

6. Schritt:

$$l_{33} = \sqrt{a_{33} - l_{13}^2 - l_{23}^2} = \sqrt{6 - (-1)^2 - 2^2} = 1$$

Daraus ergibt sich folgende linke untere Dreiecksmatrix L, bzw. deren Transponierte:

$$L = \begin{pmatrix} 3 & 0 & 0 \\ 2 & 2 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

$$L^T = \begin{pmatrix} 3 & 2 & -1 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

Nun kann man analog zur LR-Zerlegung vorgehen und durch Vorwärts- und Rückwärtseinsetzen die Lösung $\vec{x}$ ermitteln.

A.2.3 Algorithmus des Cholesky-Verfahrens

Matlab-Quellcode für die Cholesky-Zerlegung:

```
function L=cholesky(A)
[m,n]=size(A);
if (m~=n)
    disp('Die eingegebene Matrix ist nicht quadratisch !');
    return
end
L=zeros(m);
for i=1:1:m
    for j=1:1:i
        L(j,i)=A(j,i);
        if (i==j)
            for k=1:1:i-1
                L(i,i)=L(i,i)-L(k,i)*L(k,i);
```

```
            end
            L(i,i)=sqrt(L(i,i));
        else
            for k=1:1:j-1
                L(j,i)=L(j,i)-L(k,i)*L(k,j);
            end
            L(j,i)=L(j,i)/L(j,j);
        end
    end
end
L=L';
```

B Potenzmethode zur Berechnung von Eigenwerten und zugehörigen Eigenvektoren einer Matrix

B.1 Allgemeines zur Potenzmethode

Wie aus der Linearen Algebra bekannt ist, kann man zu jeder Matrix Eigenwerte und Eigenvektoren bestimmen. Da es sich in unserem Fall um spezielle Massen- und Steifigkeitsmatrizen handelt, haben auch die Eigenwerte und Eigenvektoren eine spezielle Eigenschaft: Es gibt zu jedem Eigenwert einen Eigenvektor (algebraische Vielfachheit = geometrische Vielfachheit). Hauptvektorlösungen entfallen hier.

Die Potenzmethode hat folgenden Eigenschaften/Charakteristika:

- Startvektor sollte „günstig“ gewählt sein
- Eigenwerte sind der Größe nach geordnet
- relativ gute Stabilität
- ungünstigere Konvergenz

An dieser Stelle sei noch auf andere Verfahren zur Bestimmung von Eigenwerten (-vektoren) hingewiesen:

a) Verfahren der konjugierten Gradienten

b) Verfahren der bikonjugierten Gradienten

Jedoch haben wir uns für die Potenzmethode entschieden, da sie ein sehr elegantes Verfahren ist.

B.2 Algorithmus und Gedankenbeispiel zur Potenzmethode

Da ein sinnvolles Beispiel der Potenzmethode den Rahmen sprengen würde, wollen wir *nur* den Algorithmus vorstellen und ein kleines gedankliches Beispiel durchgehen. Grundlegend für die Durchführbarkeit der Potenzmethode ist der folgende

Satz (Spektralsatz)
Seien M,$S \in \mathbb{R}^{n\times n}$ symmetrisch und sei M positiv definit. Dann gibt es eine Basis $(v_1,\ldots,v_1)$ vom $\mathbb{R}^n$, so dass

1) $Sv_k = \lambda_k \cdot Mv_k \qquad , \lambda_k \in \mathbb{R}$

2) $v_j^T \cdot M \cdot v_k = \begin{cases} 1 \text{ ,wenn } j = k \\ 0 \text{ ,wenn } j \neq k \end{cases}$

Insbesondere 2) wird später zur Orthonormierung verwendet.

Ausgangsposition:

- pos. definite Matrix A

$$A = M^{-1} \cdot S$$

  wobei M die Massenmatrix und S die Steifigkeitsmatrix darstellt

- quadratische Matrix A der Dimension $n \in N$

Algorithmus:

1. Wahl eines geeigneten Startvektors $\vec{x}$ (z.B. einfach per Zufallsgenerator)

2. Multiplikation des Startvektors $\vec{x}$ mit der Matrix A (überschreiben des Startvektors)

3. Orthogonalisierung des Startvektors $\vec{x}$ bzgl. schon ermittelter Eigenvektoren

4. M-Normierung des Startvektors $\vec{x}$ bzgl. der Massenmatrix M

5. Wiederholung der oben aufgeführten Punkte bis eine bestimmte Genauigkeit erreicht wurde oder die Schleife eine bestimmte Anzahl durchlaufen hat

Gedankenbeispiel

0. Ein Zufallsvektor $\vec{x}$ der Dimension n wird erstellt; des Weiteren wird die Matrix A gebildet durch: $A = M^{-1} \cdot S$, wobei M und S ebenfalls die Dimension n besitzen.

i. Es wird eine Schrittanzahl „schritte“, die die Anzahl der Wiederholungen beschreibt, festgelegt.

ii. Im ersten Schleifendurchlauf wird der Zufallsvektor $\vec{x}$ mit der Matrix A multipliziert und wieder in $\vec{x}$ gespeichert.

iii. Da es sich um die Bestimmung des ersten Eigenvektors handelt, ist es nicht nötig den Vektor $\vec{x}$ in einen Unterraum bzgl. der anderen Eigenvektoren zu *drücken*.

iv. Es wird eine so genannte M-Normierung durchgeführt, so dass der Vektor $\vec{x}$ nicht immer größere Werte annimmt, was zur Folge hätte, dass sich numerische Fehler verstärken.

v. Die Schritte ii. bis iv. werden „schritte"-mal wiederholt, so dass der Vektor $\vec{x}$ dem ersten Eigenvektor (zugehörig zum größten Eigenwert) entspricht.

vi. Wahl eines neuen Startvektors $\vec{x}$ durch einen Zufallsgenerator.

vii. Multiplikation des Startvektors $\vec{x}$ mit der Matrix A und Speichern im Vektor $\vec{x}$.

viii. Der Vektor $\vec{x}$ wird bzgl. des ersten Eigenvektors in einen orthogonalen Unterraum *gedrückt*.

ix. M-Normierung

x. „schritt"-fache Wiederholung der Punkte vii. bis ix. um den zweiten Eigenvektor zu bestimmen.

xi. usw.

Auf diese Weise kann man die Eigenvektoren bestimmen. Die dazugehörigen Eigenwerte sind dann einfach zu ermitteln, und zwar nach folgender Formel:

$$A \cdot \vec{v} = \lambda \cdot \vec{v}$$

$$\Rightarrow \lambda = \frac{\vec{v}^T A \vec{v}}{\vec{v}^T \vec{v}}$$

Quellcode für die Eigenvektoren

```
function [lambda,EV]=eigenvektor( M, S,schritte)
dim = size(M);
    EV = zeros(dim(1));
    for n=1:dim(1)
        x=rand(dim(1),1);
        A = inv(M)*S;
        for i=1:schritte
           x = A*x;
           for j=1:(n-1)
               x = x - EV(:,j)*(EV(:,j)'*M*x);
           end
           x = x/sqrt(x'*M*x);
        end
        EV(:,n) = x;
    end
    lambda = eigenwerte(M,S,EV,dim(1));
```

Quellcode für die Eigenwerte

```
function lambda = eigenwerte( M, S, EV, dim )
    for i=1:dim
       lambda(i) = EV(:,i)'*inv(M)*S*EV(:,i)/(EV(:,i)'*EV(:,i));
    end
```